Procesarea imaginilor sub Android

# Introducere

În cadrul lucrării de față se va dezvolta o aplicație ce va permite achiziționarea unei imagini prin intermediul camerei foto, vizualizarea și prelucrarea acestei imagini. Aplicația va permite achiziționarea imaginii în două moduri: cu ajutorul aplicației dedicate a sistemului de operare și prin controlul direct și personalizat al camerei.

# Scheletul aplicației

Spre deosebire de aplicațiile dezvoltate în cursul laboratoarelor precedente, aplicația de față va fi compusă din mai multe Activități. Aplicația de față va avea o Activitate corespunzătoare ecranului principal al aplicației, o activitate dedicată vizualizării imaginilor și o activitate dedicată capturării controlate a fotografiilor de la cameră. Fiecărei activități în parte îi corespunde o interfață (layout) proprie, definită într-un fișier xml. Pentru a scurta timpul de dezvoltare a aplicației, se va pleca de la o structură predefinită ce va conține fișierele xml corespunzătoare fiecărui layout și structura de bază a fiecărei activități. Acest proiect se poate downloada de la [www.recognizethis.org/androidlab3.rar](http://www.recognizethis.org/androidlab3.rar).

## Layout-uri

Layout-ul corespunzător activității principale este definit în fișierul *main.xml* și conține 4 butoane, câte unul pentru fiecare funcționalitate ce se dorește a fi realizată de aplicație. Se poate observa că spre deosebire de cazurile precedente, când valorile text ale butoanelor au fost declarate sub formă de string-uri, acum acestea au fost definite ca variabile în fișierul *string.xml*, fișier dedicat acestui scop. Acest mod de lucru cu valorile text în cadrul interfeței facilitează modificarea ulterioară a interfeței (Exemplu: cazul în care se dorește traducerea interfeței în altă limbă).

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<resources>

<string name=*"app\_name"*>MyFirstCamera</string>

<string name=*"btn\_takePicDefault"*>Take Picture (default)</string>

<string name=*"btn\_takePicCustom"*>Take Picture (custom)</string>

<string name=*"btn\_viewPic"*>View Picture</string>

<string name=*"btn\_editPic"*>Edit Picture</string>

<string name=*"btn\_viewBack"*>Back</string>

</resources>

Cod 1 : Fișierul strings.xml ce conține toate valorile text folosite în interfețe

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<LinearLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:orientation=*"vertical"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

>

<Button

android:id=*"@+id/btn\_takePicDefault"*

android:layout\_height=*"fill\_parent"*

android:layout\_width=*"fill\_parent"*

android:layout\_weight=*"1"*

android:text=*"@string/btn\_takePicDefault"*>

</Button>

<Button

android:id=*"@+id/btn\_takePicCustom"*

android:layout\_height=*"fill\_parent"*

android:layout\_width=*"fill\_parent"*

android:layout\_weight=*"1"*

android:text=*"@string/btn\_takePicCustom"*>

</Button>

<Button

android:id=*"@+id/btn\_viewPic"*

android:layout\_height=*"fill\_parent"*

android:layout\_width=*"fill\_parent"*

android:layout\_weight=*"1"*

android:text=*"@string/btn\_viewPic"*>

</Button>

<Button

android:id=*"@+id/btn\_editPic"*

android:layout\_height=*"fill\_parent"*

android:layout\_width=*"fill\_parent"*

android:layout\_weight=*"1"*

android:text=*"@string/btn\_editPic"*>

</Button>

</LinearLayout>

Cod 2 : Fișierul main.xml ce conține interfața corespunzătoare activității principale

În segmentul de cod de mai jos este prezentată interfața corespunzătoare activității ce permite vizualizarea fotografiilor. Se observă că pentru vizualizarea imaginilor s-a folosite widget-ul ImageView. Mai multe detalii la: <http://developer.android.com/reference/android/widget/ImageView.html>

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<LinearLayout

xmlns:android=*"http://schemas.android.com/apk/res/android"*

android:orientation=*"vertical"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*>

<ImageView

android:id=*"@+id/ImageView"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

android:layout\_weight=*"1"*>

</ImageView>

<Button

android:id=*"@+id/btn\_viewBack"*

android:layout\_width=*"fill\_parent"*

android:layout\_height=*"wrap\_content"*

android:textSize=*"30dp"*

android:gravity=*"center\_horizontal"*

android:text=*"@string/btn\_viewBack"*>

</Button>

</LinearLayout>

Cod 3 : Fișierul view.xml corespunzător activității de vizualizare a imaginilor

## Structura fișierelor .java (Activităților)

De asemenea, în fișierele *MyFirstCameraSkeletonActivity.java*, *viewActivity.java* și *captureCustom.java* au fost creeate obiectele prezente în interfață iar pentru butoane s-au creeat listenerele necesare.

**package** com.condorovici.myfirstcameraskeleton;

**import** android.app.Activity;

**import** android.os.Bundle;

**import** android.view.View;

**import** android.widget.Button;

**public** **class** MyFirstCameraSkeletonActivity **extends** Activity {

Button takePicDefault;

Button takePicCustom;

Button viewPic;

Button editPic;

View.OnClickListener takePicDefaultListener = **new** View.OnClickListener() {

// **…**

};

View.OnClickListener takePicCustomListener = **new** View.OnClickListener() {

// **…**

}

};

View.OnClickListener viewPicListener = **new** View.OnClickListener() {

// **…**

}

};

View.OnClickListener editPicListener = **new** View.OnClickListener() {

// **…**

}

};

/\*\* Called when the activity is first created. \*/

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*main*);

takePicDefault = (Button)findViewById(R.id.*btn\_takePicDefault*);

takePicCustom = (Button)findViewById(R.id.*btn\_takePicCustom*);

viewPic = (Button)findViewById(R.id.*btn\_viewPic*);

editPic = (Button)findViewById(R.id.*btn\_editPic*);

takePicDefault.setOnClickListener(takePicDefaultListener);

takePicCustom.setOnClickListener(takePicCustomListener);

viewPic.setOnClickListener(viewPicListener);

editPic.setOnClickListener(editPicListener);

}

}

Cod 4 : Structura incipientă a fișierului MyFirstCameraSkeletonActivity.java

**package** com.condorovici.myfirstcameraskeleton;

**import** android.app.Activity;

**import** android.os.Bundle;

**import** android.view.View;

**import** android.widget.Button;

**import** android.widget.ImageView;

**public** **class** viewActivity **extends** Activity {

Button btnBack;

ImageView imageView;

/\*\* Called when the activity is first created. \*/

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*view*);

btnBack = (Button)findViewById(R.id.*btn\_viewBack*);

imageView = (ImageView)findViewById(R.id.*ImageView*);

btnBack.setOnClickListener(**new** View.OnClickListener() {

@Override

**public** **void** onClick(View v) {

// **TODO** Auto-generated method stub

}

});

}

}

Cod 5 : Structura fișierului viewActivity.java

# Capturarea imaginilor cu ajutorul aplicațiilor externe

Sistemul Android permite schimbul de servicii între diverse aplicații. Astfel, fiecare aplicație poate oferi o serie de operații pe care le poate realiza la cererea altei aplicații. De exemplu, aplicația de control a camerei implicită poate fi chemată și de alte aplicații pentru capturarea pozelor. De altfel, acesta este și modul prin care se va realiza captura fotografiei în acest caz: aplicația dezvoltată va ”cere ajutorul” aplicației implicite de control a camerei pentru capturarea unei poze, indicându-i locul în care dorește să fie salvată fotografia.

Pentru pornirea unei aplicații din cadrul aplicației curente se folosesc *Intent*-urile.Un Intent este o descriere a unei operații ce se dorește a fi realizată. Mai multe detalii despre Intent-uri se găsesc la: <http://developer.android.com/reference/android/content/Intent.html>. Principalele două atribute ale unui intent sunt:

* **Acțiunea**: acțiunea ce se dorește a fi realizată (Ex: ACTION\_VIEW, ACTION\_EDIT, ACTION\_IMAGE\_CAPTURE)
* **Date**: Datele cu care se dorește realizarea acțiunii (Ex: numele persoanei în baza de date cu contacte, numărul de telefon ce se dorește a fi apelat).

Pe lângă aceste atribute principale mai există și o serie de atribute scundare:

* Category: oferă informații suplimentare despre acțiunea ce se dorește a fi executată
* Type: specifică tipul de datelor intentului
* Component: se specifică numele unei clase ce va folosi Intentul
* Extras: este o colecție de tipul *Bundle* (<http://developer.android.com/reference/android/os/Bundle.html>) în care se depozitează orice informație suplimentară.

Tipul acțiunilor ce pot fi realizate cu ajutorul unui Intent este extrem de variat și destul de greu de cunoscut în totalitate, dar în această situație [www.google.com](http://www.google.com) este un prieten adevărat! (Ex: se dorește deschiderea unui browser la o anumită adresă. O căutare de genul ” android intent browser open” oferă numeroase rezultate utile).

Pentru cazul de față se dorește pornirea aplicației de captură a imaginilor și indicarea locației în care se dorește salvarea imaginii. Indicarea acțiunii se face chiar la construcția Intentului. Acțiunea ce descrie capturarea unei fotografii este dată de: android.provider.MediaStore.*ACTION\_IMAGE\_CAPTURE*

Pe lângă descrierea acțiunii se mai dorește indicarea fișierului în care se va salva aceasta. Această informație va fi adăugată în Intent sub forma unei operații de tipul Extra. Fiecare informație de tip Extra este caracterizată de un nume și o valoare. Pentru destinația fișierului în care se va salva imaginea, numelui variabilei este: android.provider.MediaStore.*EXTRA\_OUTPUT* iar valoarea este dată sub forma unui Uri. Un Uri (Uniform Resource Identifier) este o referință către o anumită resursă, iar în acest caz poate fi creeat dintr-un fișier.

Odată creeat Intentul ce conține descrierea acțiunii de realizat și date cu privire la locația în care se dorește salvarea pozei, se pornește aplicația ce va răspunde acestui Intent.

Dat fiind că Intentul nu solicită o anumită Aplicație ci doar descrie o acțiune, este necesară depistarea aplicațiilor ce pot realiza această acțiune. Pentru a se realiza acest lucru se folosesc filtrele de intente. Fiecare aplicație are declarat un set de Intente la care poate răspunde. Astfel, este posibil ca mai multe aplicații instalate pe terminal să poate răspundă la intentul de capturare a imaginii, caz în care utilizatorul va alege la momentul respectiv aplicația dorită.

Dat fiind faptul că se așteaptă și un rezultat în urma lansării intentului, pentru lansarea acestuia ”în eter” se folosește metoda *startActivityForResult(Intent, Result)*. Al doilea parametru al metodei este folosit pentru a se identifica tipul rezultatului întors de aplicația ce a răspuns la Intent.

Sper că lucrurile ce par extrem de încurcate și complicate se vor limpezi după următorul segment de cod:

Button editPic;

//Intentul de pornire a aplicatiei implicite de captura

Intent intentDefaultCamera;

//Locul in care se va salva imaginea: nume f

String originalPicPath = Environment.*getExternalStorageDirectory*().getAbsolutePath()+"/tempOriginal.jpg";

File originalFile = **new** File(originalPicPath);

Uri originalFileUri = Uri.*fromFile*(originalFile);

//Tipul rezultatelor

**final** **static** **int** *CAMERA\_REQUEST* = 0;

View.OnClickListener takePicDefaultListener = **new** View.OnClickListener() {

@Override

**public** **void** onClick(View v) {

// **TODO** Auto-generated method stub

// Creearea intentului de captura a imaginilor

intentDefaultCamera = **new** Intent(android.provider.MediaStore.*ACTION\_IMAGE\_CAPTURE*);

// Adaugarea locatiei in care se doreste salvarea imaginii

intentDefaultCamera.putExtra(android.provider.MediaStore.*EXTRA\_OUTPUT*, originalFileUri);

// Lansarea aplicatiei ce raspunde la intent

startActivityForResult(intentDefaultCamera, *CAMERA\_REQUEST*); }

};

Cod 6 : Lansarea aplicației de captură implicită

În acest moment, la apăsarea butonului corespunzător se lansează aplicația de captură a fotografiilor. În mod evident, capturarea unei fotografii nu ar avea nici un sens dacă Activitatea curentă nu ar fi anunțată când acest lucru s-a realizat. Orice activitate are o metodă ce este apelată imediat după ce un Intent a solicitat o acțiune: onActivityResult(int requestCode, int resultCode, Intent intent). Argumentele funcții sunt:

* requestCode : codul folosit la lansarea aplicației, utilizat acum pentru a identifica sursa rezultatului
* resultCode : un cod ce indică dacă rezultatul a fost obținut cu succes
* intent : Aplicația apelată poate întoarce și un Intent ce conține date suplimentare

In cazul de față, dacă rezultatul provine de la aplicația ce a răspuns intentului și este obținut cu succes, se va afișa un mesaj. Acesta se afișează sub forma unui *Toast* (un mesaj text ce dispare după o perioadă prestabilită).

**protected** **void** onActivityResult(**int** requestCode, **int** resultCode, Intent intent){

**super**.onActivityResult(requestCode, resultCode, intent);

// se verifica daca metoda a fost apelata datorita intoarcerii unui rezultat

// in urma intentului de captura a pozei

**if**(requestCode == *CAMERA\_REQUEST*)

// se verifica daca rezultatul s-a obtinut cu succes

**if**(resultCode == *RESULT\_OK*)

{

Toast alertBox;

alertBox = Toast.*makeText*(getApplicationContext(), "ImageCaptured", 1);

alertBox.show();

}

}

/\*\* Called when the activity is first created. \*/

Cod 7 : Acțiune la primirea rezultatului

În acest moment poza este capturată și păstrată în siguranță pe SD CARD.

**ACTIVITATE: Rulați aplicația dezvoltată până în acest moment și realizați o fotografie.**

# Vizualizarea imaginilor

Pentru vizualizarea fotografiei, Activitatea curentă va apela din nou la altă Activitate, dar de această dată la o activitate componentă a aplicației curente: *viewActivity*. Mecanismul de comunicare între activități este similar, realizându-se tot prin intermediul Intent-urilor, dar de această dată Intent-ul nu va mai primi o descriere a activității ce se dorește a fi realizată, ci îi va fi indicată direct Activitatea ce urmează a fi folosită. Specificarea clasei ce se dorește a fi utilizată se face prin metoda specifică clasei Intent: *setClass(context, class)*, unde *context* reprezintă contextul aplicației (de obicei se confundă cu package-ul curent: com.example...) iar *class* reprezintă clasa ce descrie activitatea apelată.

Pe lângă specificarea clasei, în Intentul folosit va mai fi adăugată sub formă de Extra, informația ce indică locația fotografiei de adăugat. De această dată această informație nu mai este un standard, ei atribuindu-se un nume, sub formă de string și o valoare.

Întrucât nu se dorește primirea unui rezultat de la activitatea de afișarea, pornirea acesteia se realizează cu metoda startActivity(intent).

View.OnClickListener viewPicListener = **new** View.OnClickListener() {

@Override

**public** **void** onClick(View v) {

// **TODO** Auto-generated method stub

Intent intentView = **new** Intent();

// Indicarea clasei ce va raspunde intentului

intentView.setClass(getApplicationContext(), viewActivity.**class**);

// Stocarea adresei pozei de afisat sub denumirea <<imageUri>>

intentView.putExtra("imageUri", originalFileUri);

// Pornirea activitatii

startActivity(intentView);

}

};

Cod 8 : Lansarea activatii de afisare a imaginii

Pentru ca o activitate a unei aplicații să fie vizibilă, aceasta trebuie să fie declarată în manifestul aplicației (manifest.xml).

<application android:icon=*"@drawable/icon"* android:label=*"@string/app\_name"*>

<activity android:name=*".MyFirstCamera2Activity"*

android:label=*"@string/app\_name"*>

<intent-filter>

<action android:name=*"android.intent.action.MAIN"* />

<category android:name=*"android.intent.category.LAUNCHER"* />

</intent-filter>

</activity>

<activity android:name=*".viewActivity"*>

</activity>

</application>

Cod 9 : Declararea activității de afișare a pozelor

Odată realizați acești pași, activitatea de afișare a imaginii poate fi pornită cu succes. Pentru a avea însă vreo utilitate, este necesară descrierea funcționalității acesteia. În cadrul acestei activități se dorește afișarea imaginii situate la locația trimisă prin intermediul Intent-ului. Pentru aceasta se obține Intent-ul ce a fost folosit pentru pornirea aplicației, se extrag toate datele de tip Extra iar apoi se extrage locația pozei din acesta și se creează un obiect de tipul Bitmap din datele situate la această locație.

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*view*);

btnBack = (Button)findViewById(R.id.*btn\_viewBack*);

imageView = (ImageView)findViewById(R.id.*ImageView*);

// Bitmap-ul ce va contine imaginea de afisat

Bitmap displayPic;

// Intentul ce a fost folosit pentru deschiderea activitatii

Intent callingIntent = **new** Intent();

// URI-ul corespunator fisierului de afisat

Uri imageFileUri;

// Obtinerea Intent-ului de afisat

callingIntent = getIntent();

// Extragerea datelor de tip Extras

Bundle extras = callingIntent.getExtras();

// Extragerea adresei fisierului de afisat

imageFileUri = (Uri)extras.get("imageUri");

Cod 10 : Obținerea adreasei fișierului de afișat (viewActivity.java)

Pentru obținerea obiectului Bitmap dintr-un fișier se folosește metoda *decodeFile* ce aparține clasei *BitmapFactory*. Aceasta primește ca argumente un string cu numele fișierului de decodat și un obiect de tipul BitmapFacotry.Options cu diverse opțiuni de decodare. In acest exemplu se vor folosi opțiunile implicite. Mai multe detalii la <http://developer.android.com/reference/android/graphics/BitmapFactory.html> .

// Extragerea adresei fisierului de afisat

imageFileUri = (Uri)extras.get("imageUri");

// Decodarea fisierului intr-un obiect de tipul Bitmap

displayPic = BitmapFactory.*decodeFile*(imageFileUri.getPath());

// Afisarea Bitmap-ului

imageView.setImageBitmap(displayPic);

btnBack.setOnClickListener(**new** View.OnClickListener() {

@Override

**public** **void** onClick(View v) {

// **TODO** Auto-generated method stub

finish();

}

});

Cod 11 : Decodarea și afișarea imaginii

Funcționalitatea singurului buton din această activitate este aceea de a o închide și a reveni la activitatea principală, funcționalitate realizată prin apelarea funcției *finish()*.

**ACTIVITATE: Vizualizați o fotografie realizată.**

# Modificarea imaginilor la nivel de pixeli

Așa cum am văzut în capitolul anterior, imaginile sunt reprezentate sub formă de obiecte *Bitmap* ce pot fi obținute dintr-un fișier ce conține imaginea. Pentru a se accesa pixelii unei imagini se utilizează metoda *getPixels* specifică clasei *Bitmap*: *getPixels(int[] pixels, int offset, int stride, int x, int y, int width, int height)*, unde parametrii reprezintă:

* Pixels: Un array de tipul int ce va conține pixelii
* Offset: poziția de la care se începe scrierea în pixels[]
* Stride: stride-ul imaginii (uzual este egală cu lățimea imaginii; detalii despre stride: <http://msdn.microsoft.com/en-us/library/windows/desktop/aa473780(v=vs.85).aspx)>
* X: Coordonata pe x a primului pixel citit din bitmap
* Y: Coordonata pe y a primului pixel citit din bitmap
* Width: numărul de pixeli citiți de pe fiecare rând
* Height: numărul de rânduri citite

În mod implicit un pixel este stocat sub forma unui întreg pe 32 de biți, din care primii 8 biți corespund canalului de Alpha, următorii 8 canalului de roșu, următorii 8 biți canalului de verde și ultimii 8 biți canalului de albastru:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pixels[1] | B31B30B29B28B27B26B25B24 | B23B22B21B20B19B18B17B16 | B15B14...B8 | B7B6B5B4B3B2B1B0 |

Pentru extragerea valorilor corespunzătoare fiecărui plan de culoare se vor folosi operații pe biți. Fie un pixel descris de pix = (255,127,96), reprezentat în formatul ARGB32. Pentru extragerea valorii corespunzătoare planului de verde se vor realiza operațiile green = (pixel>>8)&0xff:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Alpha | Red | Green | Blue |
| pixel | 255 | 255 | 127 | 96 |
| pixel | 11111111 | 11111111 | 01111111 | 01100000 |
| Pixel>>8 | 00000000 | 11111111 | 11111111 | 01111111 |
| 0xff | 00000000 | 00000000 | 00000000 | 11111111 |
| (Pixel>>8)&0xff | 00000000 | 00000000 | 00000000 | 01111111 |

În mod analog, valoarea corespunzătoare planelor de roșu și de albastru se obțin prin: red = (pixel>>16)&0xff și blue = pixel & 0xff.

Pentru împachetarea valorilor de roșu, verde și albastru înapoi se utilizează: pixel = 0xff000000 | (red<<16) | (green<<8) | blue.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | B31-B24 | B23-B16 | B15-B8 | B7-B0 |
| Red | 00000000 | 00000000 | 00000000 | 11111111 |
| Green | 00000000 | 00000000 | 00000000 | 01111111 |
| Blue | 00000000 | 00000000 | 00000000 | 01100000 |
| **0xff000000** | **11111111** | **00000000** | **00000000** | **00000000** |
| **Red<<16** | **00000000** | **11111111** | **00000000** | **00000000** |
| **Green<<8** | **00000000** | **00000000** | **01111111** | **00000000** |
| **Blue** | **00000000** | **00000000** | **00000000** | **01100000** |
| 0xff000000|R<<16|G<<8|B | 11111111 | **11111111** | **01111111** | **01100000** |

Pentru ca un Bitmap să poată fi modificat acesta trebuie să fie *mutable*. Un obiect de tipul Bitmap este immutable după ce a fost creat prin decodarea unui fișier, așadar nu poate fi modificat. Pentru a se putea realiza modificarea sa, se poate face o copie a acestia, copia fiind un obiect mutable ce se poate modifica. Această limitare poate fi depășită prin specificarea opțiunii *inmutable* la decodare, dar doar pentru versiuni de Android mai noi de 2.3.1. După realizarea copiei mutable a obiectului, obiectul original, immutable poate fi șters prin apelarea metodei recycle().

Imaginea obținută după modificare poate fi salvată într-un fișier prin apelarea funcției *compress(CompressFormat format, int Quality, OutputStream Stream)* ce primește ca parametri tipul fișierului imagine, calitatea, și streamul corespunzător fișierului în care se dorește salvarea imaginii.

Pentru salvarea imaginii pe card, este necesară declararea acestei permisiuni în manifestul aplicației. Pentru situația în care se salva pe cardul de memorie imaginea capturată de cameră nu era necesară declararea acestei permisii deoarece scrierea era făcută de aplicația camerei, aplicație ce avea declarată permisiunea respectivă. Permisiile se declară în fișierul *AndroidManifest.xml*.

<application android:icon=*"@drawable/icon"* android:label=*"@string/app\_name"*>

<activity android:name=*".MyFirstCamera2Activity"*

android:label=*"@string/app\_name"*>

<intent-filter>

<action android:name=*"android.intent.action.MAIN"* />

<category android:name=*"android.intent.category.LAUNCHER"* />

</intent-filter>

</activity>

<activity android:name=*".viewActivity"*>

</activity>

</application>

**<uses-permission android:name=*"android.permission.WRITE\_EXTERNAL\_STORAGE"* />**

</manifest>

Cod 12 : Declararea permisiei de scriere pe cardul de memorie externă

View.OnClickListener editPicListener = **new** View.OnClickListener() {

@Override

**public** **void** onClick(View v) {

// **TODO** Auto-generated method stub

Bitmap origBmp, imageBmpMutable;

// Decodarea imaginii din fisier

origBmp = BitmapFactory.*decodeFile*(originalFileUri.getPath());

// Crearea unei copii mutable a Bitmap-ului

imageBmpMutable = origBmp.copy(origBmp.getConfig(), **true**);

// Stergerea obiectului initial

origBmp.recycle();

// Dimensiunile imaginii

**int** width = imageBmpMutable.getWidth();

**int** height = imageBmpMutable.getHeight();

// Vectorul ce va contine valorile pixelilor

**int**[] pixArr = **new** **int**[width \* height];

// Stream-ul corespunzator fisierului de iesire

FileOutputStream outStream = **null**;

// Obtinerea pixelilor

imageBmpMutable.getPixels(pixArr, 0, width, 0, 0, width, height);

**int** R, G, B;

**int** index = 0;

// parcurgerea imaginii

**for**(**int** y = 0; y < height; y++)

{

**for**(**int** x = 0; x < width; x++)

{

// extragerea planelor de culoare

**int** r = (pixArr[index] >> 16) & 0xff;

**int** g = (pixArr[index] >> 8) & 0xff;

**int** b = pixArr[index] & 0xff;

R = r;G = 127;B = 127;

// reimpachetarea valorilor modificate

pixArr[index] = 0xff000000 | (R<<16) | (G<<8) | B;

index = index + 1;

}

}

// actualizarea Bitmap-ului cu noile valorile ale pixelilor

imageBmpMutable.setPixels(pixArr, 0, width, 0, 0, width, height);

// Salvarea fisierului

**try** {

outStream = **new** FileOutputStream(originalFile);

imageBmpMutable.compress(Bitmap.CompressFormat.*JPEG*, 100, outStream);

} **catch** (FileNotFoundException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

};

Cod 13 : Editarea imaginilor la nivel de pixel