Salvando arquivos de imagem únicos

Nós já conseguimos salvar a foto tirada a partir do aplicativo, mas não conseguiamos fazer com que esta aparecesse no ImageView. Faremos isto adiante.

Observe que quando damos um startActivity, perderemos o controle de onde nosso código será executado.

@Override  
public void onClick(View v) {  
 Intent intentCamera = new Intent(MediaStore.ACTION\_IMAGE\_CAPTURE);  
 String caminhoFoto = getExternalFilesDir(null) + "/"+ System.currentTimeMillis() +".jpg";  
 File arquivoFoto = new File(caminhoFoto);  
 intentCamera.putExtra(MediaStore.EXTRA\_OUTPUT, Uri.fromFile(arquivoFoto));  
 startActivity(intentCamera);  
}

Se tentarmos adicionar // abrir a foto que a gente tirou logo abaixo deste trecho, antes de abrirmos o aplicativo da câmera já estaremos tentando abrir a foto.

Precisaremos encontrar uma maneira para que o sistema operacional nos avise que a foto já foi tirada e que poderemos utilizá-la. Queremos saber o resultado da Activity, então usaremos um startActivityForResult e um segundo parâmetro 567. Mais adiante, iremos explicar o número.

startActivityForResult(intentCamera, 567);

Mais abaixo, iremos sobrescrever o método onActivityResult, que será chamado quando o startActivityForResult estiver devolvendo o resultado. Nosso objetivo é abrir a foto tirada.

@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
 // abrir a foto que a gente tirou   
}

Vamos imaginar se tivéssemos um segundo botão, referente ao microfone, por exemplo. Iremos criar o código:

Button botaoMic = (Button) findViewById(R.id.formulario\_botao\_foto);  
botaoMic.setOnCLickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intentMic = new Intent(MediaStore.ACTION\_IMAGE\_CAPTURE);  
 String caminhoFoto = getExternalFilesDir(null) + "/foto.jpg";  
 File arquivoFoto = new File(caminhoFoto);  
 intentMic.putExtra(MediaStore.EXTRA\_OUTPUT, Uri.fromFile(arquivoFoto));  
 startActivityForResult(intentMic, 567);  
 }

No entanto, quando ele retornar, irá cair no onActivityResult, que no fim, irá abrir a foto. Com o botão do microfone, esperaríamos que fosse gravado o áudio. Precisaríamos encontrar uma forma de diferenciar os resultados. Para distingui-los iremos usar o número especificado como segundo parâmetro no startActivityForResult. Para o microfone, podemos usar o 678.

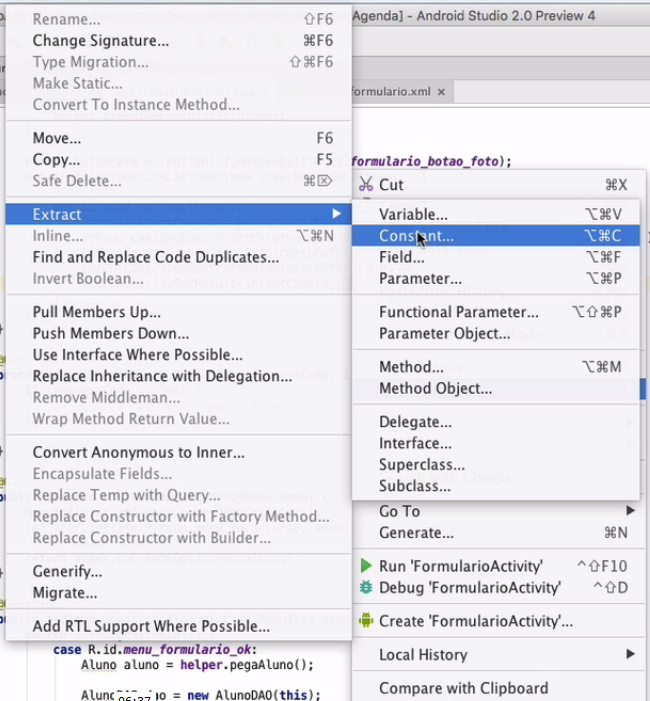
startActivityForResult(intentMic, 678);

Agora, poderemos adicionar ifs no onActivityResult:

@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data){  
 if (requestCode == 567) {  
 // abrir a foto que a gente tirou  
 } else if (requestCode == 678) {  
 // trata o som do microfone  
 }  
}

Mas o exemplo, foi só para compreendermos a utilidade do segundo parâmetro. Não pretendemos trabalhar com microfone no nosso aplicativo e iremos descartar o trecho relacionado a ele no código.

Agora que entendemos a utilidade, vamos transformar o segundo parâmetro do startActivityForResult em uma constante. Iremos clicar com o botão direito sobre o valor 567, iremos selecionar : Refector --> Extract --> Constant.



Iremos chamar a constante de CODIGO\_CAMERA. Iremos adicioná-la também no onActivityResult.

Agora, o nosso código está assim:

public class FormularioActivity extends AppCompatActivity {  
  
 public static final int CODIGO\_CAMERA = 567;  
 private FormularioHelper helper;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_formulario);  
  
 helper = new FormularioHelper(this);  
  
 Intent intent = getIntent();  
 Aluno aluno = (Aluno) intent.getSerializableExtra("aluno");  
 if (aluno != null){  
 helpe.preencheFormulario(aluno);  
 }  
  
 Button botaoFoto = (Button) findViewById(R.id.formulario\_botao\_foto);  
 botaoFoto.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intentCamera = new Intent(MediaStore.ACTION\_IMAGE\_CAPTURE);  
 String caminhoFoto = getExternalFilesDir(null) + "/foto.jpg";  
 File arquivoFoto = new File(caminhoFoto);  
 intentCamera.putExtra(MediaStore.EXTRA\_OUTPUT, Uri.fromFile(arquivoFoto));  
 startActivityForResult(intentCamera, CODIGO\_CAMERA);  
 }  
  
 })  
 }  
  
 @Override  
 protect void onActivityResult(int requestCode, int resultCode, Intent data) {  
 if (requestCode == CODIGO\_CAMERA) {  
 // abrir a foto que a gente tirou  
 }  
  
 }  
\...

Para fazer com que a imagem abra na nossa tela, teremos que criar uma referência para o ImageView, adicionando findViewById. Dentro colocaremos o nome da nossa Activity.

findViewById(R.id.formulario\_foto);

Depois, iremos transformá-la em uma variável e chamá-la como foto.

ImageView foto = findViewById(R.id.formulario\_foto);

Vamos fazer o *cast* de findViewById, usando o comando ALT + ENTER.

Depois, vamos adicionar foto.setImageBitmap e momentaneamente usaremos o parâmetro bitmap. Criaremos a classe Bitmap, vamos usar a BitmapFactory e adicionar decodeFile que irá pedir o caminho (caminhoFoto).

@Override  
protect void onActivityResult(int requestCode, int resultCode, Intent data) {  
 if (requestCode == CODIGO\_CAMERA) {  
 ImageView foto = (ImageView) findViewById(R.id.formulario\_foto);  
 Bitmap bitmap = Bitmap Factory.decodeFile(caminhoFoto);  
 foto.setImageBitmap(bitmap);  
  
 }  
}

Porém, ainda não temos acesso à caminhoFoto, porque ela é uma variável local do método onClick. Teremos que transformá-la em um atributo. Vamos remover Stringe pediremos para ele criar um field na Activity.

Button botaoFoto = (Button) findViewById(R.id.formulario\_botao\_foto);  
botaoFoto.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intentCamera = new Intent(MediaStore.ACTION\_IMAGE\_CAPTURE);  
 caminhoFoto = getExternalFilesDir(null) + "/" + System.currentTimeMillis() + ".jpg";  
 File arquivoFoto = new File(caminhoFoto);  
 intentCamera.putExtra(MediaStore.EXTRA\_OUTPUT, Uri.fromFile(arquivoFoto));  
 startActivityForResult(intentCamera, CODIGO\_CAMERA);  
}  
  
})  
}

Criamos uma variável privada, do tipo *String*, representando o caminho da nossa foto.

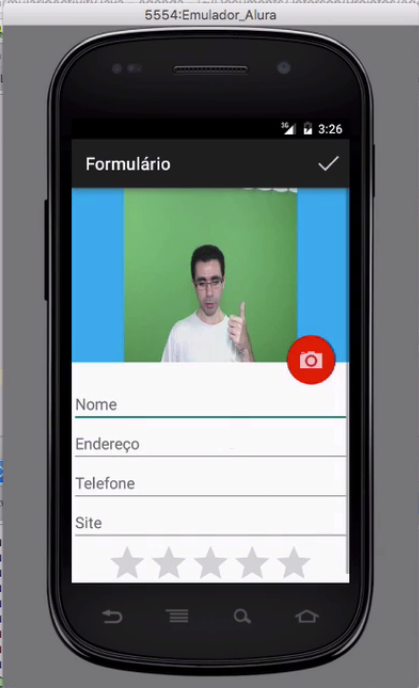
Porém, dependo da câmera usada, a imagem pode ter uma resolução muito grande e não ser suportada pelo ImageView. Teremos que reduzir o tamanho do Bitmap. Também iremos especificar as dimensões e adicionar true, para que seja usado um filtro depois da redução da imagem. Para dar diferenciação, chamaremos de bitmapReduzido.

Bitmap bitmapReduzido = Bitmap.createScaledBitmap(bitmap, 300, 300, true);

O trecho do código está assim:

@Override  
protect void onActivityResult(int requestCode, int resultCode, Intent data) {  
 if (requestCode == CODIGO\_CAMERA) {  
 ImageView foto = (ImageView) findViewById(R.id.formulario\_foto);  
 Bitmap bitmap = BitmapFactory.decodeFile(caminhoFoto);  
 Bitmap bitmapReduzido = Bitmap.createScaledBitmap(bitmap, 300, 300, true);  
 foto.setImageBitmap(bitmapReduzido);  
  
 }  
}

Vamos testar novamente o app e ver o que acontece.

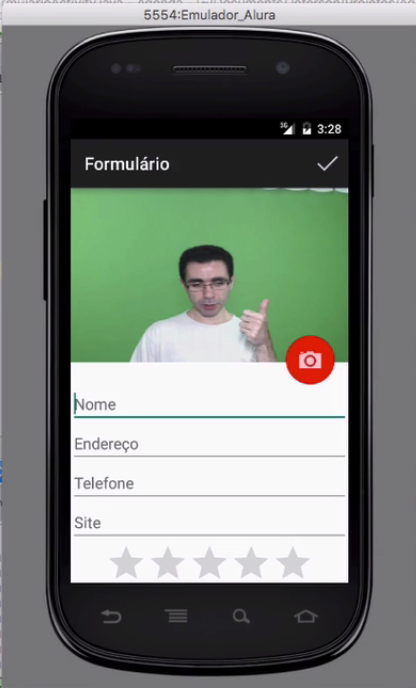


Quando tiramos uma foto, ela aparecerá no ImageView do contato. No entanto, queremos que a foto não fique com um aspecto achatado e também, desejamos aproveitar o espaço das bordas azuis ao redor da imagem. Faremos isto, adicionando a seguinte linha:

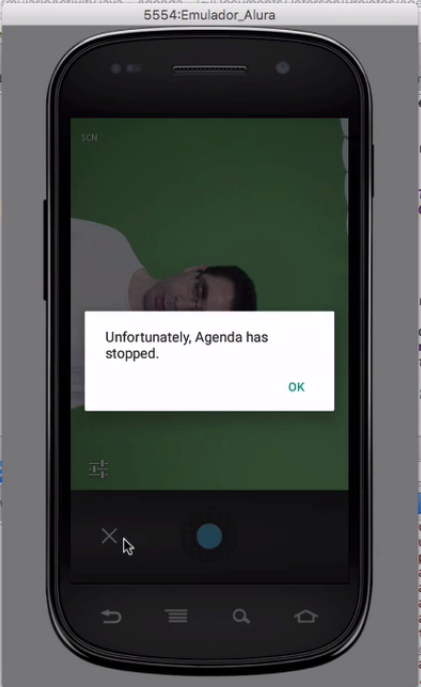
foto.setScaleType(ImageView.ScaleType.FIT\_XY);

Com o FIT\_XY, estamos especificando que a imagem deve se encaixar no espaço, ajustada tanto na largura como na altura disponível no ImageView.

Rodaremos de novo a aplicação e a imagem já terá as alterações.



Mas o que acontecerá se ao acessarmos a câmera, desistirmos de tirar a foto?



A app deixará de funcionar. Temos um problema... Nós cancelamos a foto e a Activityda câmera não criou o arquivo. Quando tentarmos abrir o bitmap, ele será nulo. Observe as seguintes linhas:

Bitmap bitmap = BitmapFactory.decodeFile(caminhoFoto);  
Bitmap bitmapReduzido = Bitmap.createScaledBitmap(bitmap, 300, 300, true);

Ao tentarmos reduzir o tamanho do bitmapnulo, teremos uma exceção (*NullPointerException*).

Para resolver o problema, usaremos o resultCode, que irá nos indicar se a ação de tirar foto foi completada. Adicionaremos um if:

if(resultCode == Activity.RESULT\_OK)

Com a nova linha, o trecho do nosso código ficará assim:

@Override  
protect void onActivityResult(int requestCode, int resultCode, Intent data) {  
 (resultCode == Activity.RESULT\_OK) {  
 if (requestCode == CODIGO\_CAMERA) {  
 ImageView foto = (ImageView) findViewById(R.id.formulario\_foto);  
 Bitmap bitmap = BitmapFactory.decodeFile(caminhoFoto);  
 Bitmap bitmapReduzido = Bitmap.createScaledBitmap(bitmap, 300, 300, true);  
 foto.setImageBitmap(bitmapReduzido);  
  
 }  
}  
  
}

Caso o arquivo tenha sido criado de fato, iremos executar a parte do código referente à câmera.

Após as alterações no código, quando fizermos um outro teste no app e cancelarmos novamente a câmera, teremos o comportamento esperado. Iremos retornar ao menu e o funcionamento continuará normal. Se tirarmos uma foto, ela será inserida no formulário.