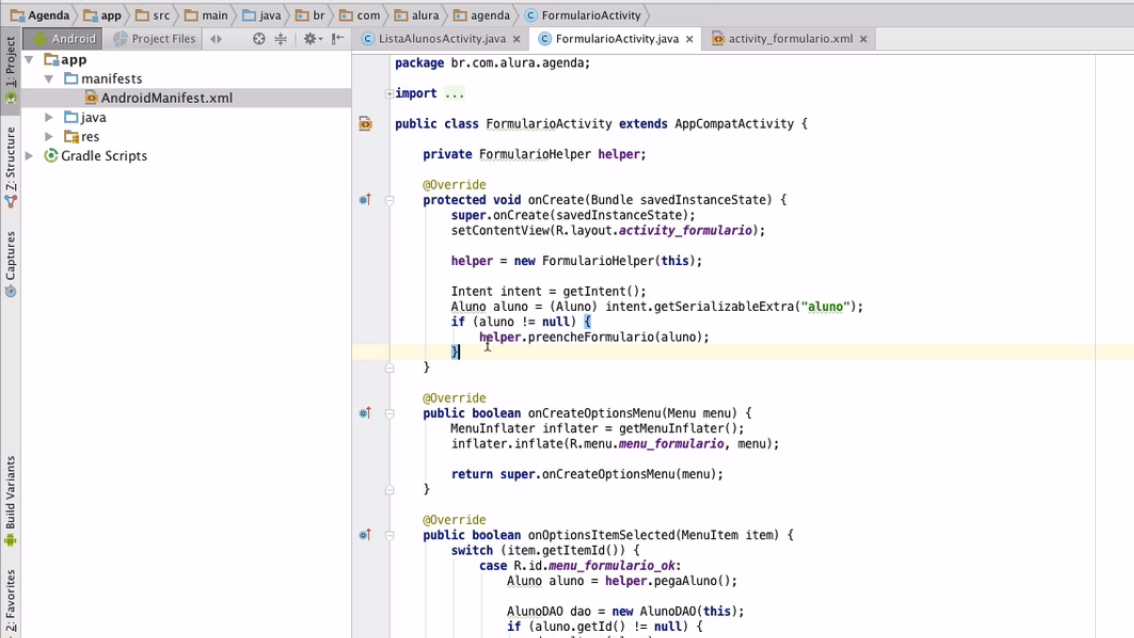
Botões funcionando

Vamos colocar um comportamento para o botão com a câmera. Nosso objetivo é que ao clicarmos nele, abra a câmera e seja possível tirar uma foto.

Iremos voltar ao arquivo FormularioActivity.java.



Criaremos um referência para o nosso botão abaixo do if. Adicionaremos uma findViewById, especificaremos o id com o nome que usamos anteriormente formulario\_botao\_foto.

View ViewById = findViewById(R.id.formulario\_botao\_foto);

Guardaremos isto em um variável, chamada botaoFoto. Iremos alterar View para Button e depois, vamos importá-lo.

Button botaoFoto = (Button) findViewById(R.id.formulario\_botao\_foto);

Para adicionar um comportamento, vamos adicionar o Listener simples. Dentro, vamos adicionar uma classe anônima e vamos instanciar a interface OnClickListenere o Android já dará qual método teremos que implementar.

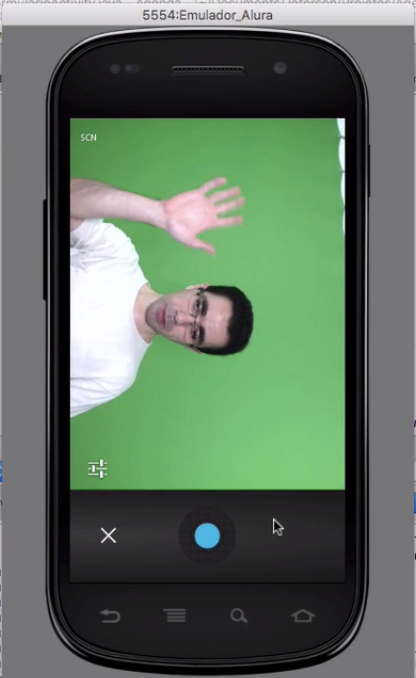
\...  
if (aluno != null) {  
 helper.preencheFormulario(aluno);  
}  
  
Button botaoFoto = (Button) findViewById(R.id.formulario\_botao\_foto);  
botaoFoto.setOnClickListener{new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 }  
 });

Nós queremos abrir o activity da câmera, adicionaremos uma Intent com uma ação que o Android entende: MediaStore. Dentro desta classe, iremos usar ACTION\_IMAGE\_CAPTURE.

Button botaoFoto = (Button) findViewById(R.id.formulario\_botao\_foto);  
botaoFoto.setOnClickListener{new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent intentCamera(MediaStore.ACTION\_IMAGE\_CAPTURE);  
 startActivity(intentCamera);  
 }  
 });

Adicionamos também o startActivity e passamos como parâmetro intentCamera.

Vamos testar o aplicativo. Ao clicarmos no botão da câmera, veremos que ele abrirá a câmera com sucesso.



Conseguimos tirar a foto, mas onde foi salvo o arquivo de imagem? Se procurarmos na galeria do celular, veremos que a foto não está lá. Para não nos preocuparmos demais com o tema, vamos indicar no código onde queremos que a imagem seja salva.

Se temos uma Intent e queremos passar outro parâmetro para uma Activity, podemos usar o método putExtra(). Precisaremos passar um nome e um valor.

@Override  
public void onClick(View v) {  
 Intent intentCamera = new Intent(MediaStore.ACTION\_IMAGE\_CAPTURE);  
 intentCamera.putExtra("caminhoFoto", "/foto.jpg");  
 startActivity(intentCamera);  
}

Nossa Activity terá que conseguir ler a chave caminhoFotopara saber onde tem que salvar a /foto.jpg. Lembre-se que cada uma das aplicações do celular foram implementadas por uma equipe diferente. Então, sempre que precisarmos especificar o caminho para ação de tirar foto, foi convencionado que usaremos como chave uma constante específica: MediaStore.EXTRA\_OUTPUT. Mas quando definirmos o caminho, não poderemos usar uma *String*. Teremos que usar uma Uri.

Porém, existe um detalhe: cada aplicação só poderá ser da área em que foi instalada. Precisaremos dizer que /foto.jpg deverá estar dentro da área onde nossa aplicação pode escrever.

Nosso código ficará assim:

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

Para facilitar, vamos separar a imagem na variável caminhoFoto, que será uma *string* definida logo acima.

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

Já concatenamos a pasta em que está armazenado "/foto.jpg". Usaremos o método getExternalFilesDir e iremos passar o null como parâmetro, porque a raiz será o suficiente.

String caminhoFoto = getExternalFilesDir(null) + "/foto.jpg";

O trecho do nosso código ficou assim:

@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));  
 startActivity(intentCamera);  
}

Com File(caminhoFoto), criamos um objeto do tipo arquivo. A partir dele, vamos gerar a Uri. Então, o aplicativo da câmera irá conseguir salvar o arquivo, no caminho que pedimos.

Em seguida, se fizermos um novo teste e tentarmos salvar outra foto, veremos que ela será salva no caminho especificado: na pasta da aplicação, com o nome /foto.jpg.

Porém, se tirarmos uma segunda foto, perderemos a primeira. Para isto não acontecer, além de especificarmos o caminho, precisamos ter um nome dinâmico. Após apagar o nome foto, iremos usar uma ideia semelhante à usada pelas cameras fotográfica, que utilizam-se do *timestamp*, registando a data e a hora da foto. Adicionaremos o método System.currentTimeMillis(), que irá nos fornecer seguramente um número único.

@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);  
}