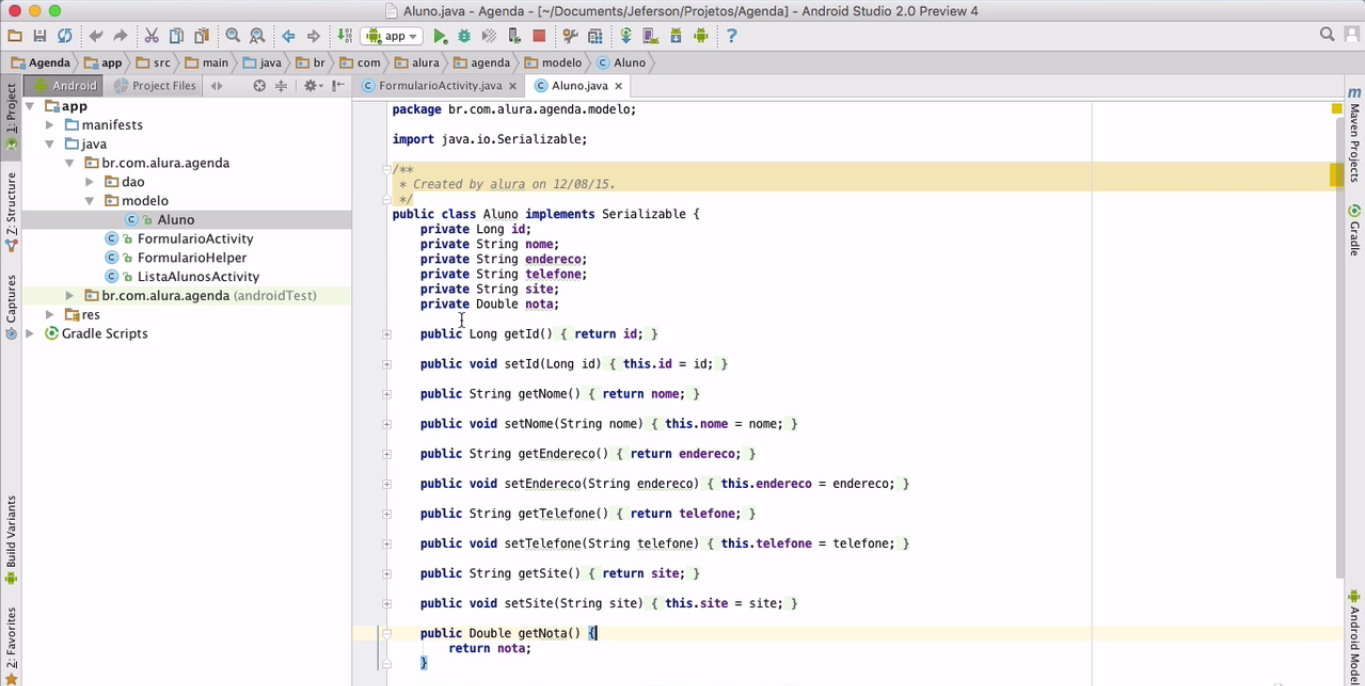
Salvando o arquivo de imagem

Nós já tivemos bastante trabalho incrementando o cadastro do aluno com o botão de câmera. Mas falta ainda salvar a imagem no banco. Iremos começar a trabalhar com as classes referentes a isto.

Abriremos o pacote chamado modelo e depois, acessaremos a classe Aluno.



Vamos adicionar mais um campo no código, referente a foto do aluno. Adicionaremos private e a foto. Mas qual seria o tipo?

Poderíamos guardar como Bitmap, no entanto, se salvarmos a imagem no banco, estaremos duplicando a mesma imagem. A melhor forma de salvar a imagem seria salvar o caminho.

private String caminhoFoto;

Precisaremos adicionar getters e setters. Iremos criar getCaminhoFoto e setCaminhoFoto.

public void getCaminhoFoto() {  
 return caminhoFoto;  
}  
  
public void setCaminhoFoto(String caminhoFoto) {  
 this.caminhoFoto = caminhoFoto;  
}

No método onOptionItemSelected, sempre que o item era do menu\_formulario\_ok - referente ao botão de **OK** - o código para inserir o aluno era executado.

@Override  
public boolean onOptionItemSelected(MenuItem item) {  
 switch (item.getItemId()) {  
 case R.id.menu\_formulario\_ok;  
 Aluno aluno = helper.pegaAluno();  
  
 AlunoDAO dao = new AlunoDAO(this);  
 if (aluno.getId() != null) {  
 dao.altera(aluno);  
 } else {  
 dao.insere(aluno);  
 }  
 dao.close();  
  
 Toast.makeText(FormularioActivity.this, "Aluno " + aluno.getNome() + " salvo!", Toast.LENGTH\_SHORT).show();  
  
 finish();  
 break;  
  
 }  
  
 return super.onOptionsItemSelected(item);  
  
 }

O dao era o responsável por alterar e inserir o aluno. Iremos fazer alterações na classe AlunoDAO.

Criamos uma linha referente à foto no onCreate. Vamos inserir caminhoFoto e especificar que é do tipo TEXT.

public class AlunoDAO extends SQLiteOpenHelper {  
 public AlunoDAO(Context context) { super(context, "Agenda" null, 1); }  
  
 @Override  
 public void onCreate(SQLiteDatabase db) {  
 String sql = "CREATE TABLE Alunos (id INTEGER PRIMARY KEY, " +  
 "nome TEXT NOT NULL, " +  
 "endereco TEXT, " +  
 "telefone TEXT, " +  
 "site TEXT, " +  
 "nota REAL, " +  
 "caminhoFoto TEXT);";  
  
 }  
  
 @Override  
 public void onUpgrade{SQLiteDatabase db, int oldVersion, int newVErsion) {  
 String sql = "DROP TABLE IF EXISTS Alunos";  
 db.execSQL(sql);  
 onCreate(db);  
  
 }}  
}

Para os alunos que tem uma versão anterior do banco de dados e ainda não têm a coluna de caminhoFoto, será executado o método onUpgrade.

Agora, não será suficiente descartar a tabela e recriá-la, porque iremos perder todos os alunos já cadastrados.

Iremos substituir a instrução do SQL DROP TABLE IF EXISTS Alunos por "ALTER TABLE", que irá adicionar uma coluna a mais. Também teremos que especificar o nome e o tipo de dado da coluna.

@Override  
public void onUpgrade{SQLiteDatabase db, int oldVersion, int newVersion) {  
 String sql = "ALTER TABLE Alunos ADD COLUMN caminhoFoto TEXT";  
 db.execSQL(sql);  
  
 }

Observe que nós vamos executar o SQL, mas não vamos mais chamar o onCreate.

Iremos adicionar também um switch para resolver a questão do uso de diferentes versões do banco de dados. É uma opção melhor do que inserirmos um if, que iria exigir uma extensa duplicação de código.

@Override  
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
 switch (oldVersion) {  
 case 1:  
 String sql = "ALTER TABLE Alunos ADD COLUMN caminhoFoto TEXT";  
 db.execSQL(sql); // indo para versao 2  
 break;  
  
 case 2:  
 String sql = "ALTER TABLE Alunos ADD COLUMN cpf TEXT";  
 db.execSQL(sql); // indo para versao 3  
 break;  
  
 }  
 }

Porém, se usarmos o break no case 1, o trecho referente ao case 2, não será executado. Iremos retirar o break e o código será executado diretamente, sem a verificação de quais são os cases. No exemplo, ele continuará executando da versão 2 para a versão 3 e seguiria assim até as próximas.

Como o escopo de todas será o mesmo, a variável sql não poderá ter o mesmo nome. A solução será declará-la vazia, e não usá-la mais dentro do case.

@Override  
public void onUpgrade(SQLiteDatabase db, int oldVersionm int newVersion) {  
 String sql = "";  
 switch (oldVersion) {  
 case 1:  
 sql = "ALTER TABLE Alunos ADD COLUMN caminhoFoto TEXT";  
 db.execSQL(sql); // indo para versao 2  
 break;  
  
 case 2:  
 sql = "ALTER TABLE Alunos ADD COLUMN cpf TEXT";  
 db.execSQL(sql); // indo para versao 3  
 break;  
  
 }  
 }

Vamos apagar o case 2 do nosso código, porque não temos cpf.

Teremos que mudar no construtor, o parâmetro que indica a versão, para que possa ser feito o upgrade.

public class AlunoDAO extends SQLiteOpenHelper {  
 public AlunoDAO(Context context) { super(context, "Agenda" null, 2); }

Em seguida, iremos mudar os outros métodos do DAO.

public void insere(Aluno aluno) {  
 SQLiteDatabase db = getWritableDatabase();  
  
 ContentValues dados = pegaDadosDoAluno(aluno);  
  
 db.insert("Alunos ", null, dados);  
}  
  
@NonNull  
private ContentValues pegaDadosDoAluno(Aluno aluno) {  
 ContectValues dados = new ConteValues();  
 dados.put("nome", aluno.getNome());  
 dados.put("endereco", aluno.getEndereco());  
 dados.put("telefone", aluno.getTelefone());  
 dados.put("site", aluno.getSite());  
 dados.put("nota", aluno.getNota());  
 dados.put("caminhoFoto", aluno.getCaminhoFoto())  
 return dados;  
}

Adicionamos uma linha para caminhoFoto. Faremos o mesmo no método buscaAluno:

List<ALuno> alunos = new ArrayList<~>();  
while (c.moveToNext()) {  
 Aluno aluno = new Aluno();  
 aluno.setId(c.getLong(c.getColumnIndex("id")));  
 aluno.setNome(c.getString(c.getColumnIndex("nome")));  
 aluno.setEndereco(c.getString(c.getColumnIndex("endereco")));  
 aluno.setTelefone(c.getString(c.getColumnIndex("telefone")));  
 aluno.setSite(c.getString(c.getColumnIndex("site")));  
 aluno.setNota(c.getString(c.getColumnIndex("nota")));  
 aluno.setCaminhoFoto(c.getString(c.getColumnIndex("caminhoFoto")));  
  
 alunos.add(aluno);  
}  
  
c.close();  
  
return alunos;

Nos métodos deleta e pegaDadosDoAluno não receberam alterações, porque já fizemos a refatoração no insere.

No onOptionsItemSelected, iremos mudar o campo pegaAluno, para conseguirmos criar o aluno com o caminho da foto.

Atualmente, nosso código está assim:

@Override  
public boolean onOptionItemSelected(MenuItem item) {  
 switch (item.getItemId()) {  
 case R.id.menu\_formulario\_ok;  
 Aluno aluno = helper.pegaAluno();  
  
 AlunoDAO dao = new AlunoDAO(this);  
 if (aluno.getId() != null) {  
 dao.altera(aluno);  
 } else {  
 dao.insere(aluno);  
 }  
 dao.close();  
  
 Toast.makeText(FormularioActivity.this, "Aluno " + aluno.getNome() + " salvo!", Toast.LENGTH\_SHORT).show();  
  
 finish();  
 break;  
  
 }  
  
 return super.onOptionsItemSelected(item);  
  
 }

No FormularioHelper, iremos adicionar a seguinte linha no pegaAluno:

public Aluno pegaAluno() {  
 aluno.setNome(campoNome.getText().toString());  
 aluno.setEndereco(campoEndereco.getText().toString());  
 aluno.setTelefone(campoTelefone.getText().toString());  
 aluno.setSite(campoSite.getText().toString());  
 aluno.setNota(Double.valueOf(campoNota.getProgress());  
 aluno.setCaminhoFoto(campoFoto);  
 return aluno;  
  
 }

Mas como não temos o campoFoto ainda, precisaremos criar um findViewById. Antes vamos adicionar um private final

private final ImageView campoFoto;

Vamos importá-lo com ALT + ENTER e inserir campoFoto no construtor.

public formularioHelper(FormularioActivity activity) {  
 campoNome = (EditText) activity.findViewById(R.id.formulario\_nome);  
 campoEndereco = (EditText) activity.findViewById(R.id.formulario\_endereco);  
 campoTelefone = (EditText) activity.findViewById(R.id.formulario\_telefone);  
 campoSite = (EditText) activity.findViewById(R.id.formulario\_site);  
 campoNota = (RatingBar) activity.findViewById(R.id.formulario\_nota);  
 campoFoto = activity.findViewById(R.id.formulario\_foto);  
 aluno = new Aluno();  
}

Após fazer o cast de formulario\_foto para ImageView - que não guarda o bitmap, sem ter relação com a imagem. De volta ao FormularioActivity, vamos até o onActivityResult. Nele, adicionaremos o método setTag e dentro, o caminhoFoto

@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
 if (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.setScaleType(ImageView.ScaleType.FIT\_XY);  
 foto.setTag(caminhoFoto);  
  
  
 }  
 }  
 }

Agora, associamos o caminho da foto com o ImageView. Ao retornarmos ao formulario\_helper, no pegaAluno, adicionaremos o caminho da foto, juntamente com um getTag. Nós sabemos que o *object* era uma *string*, faremos um cast e adicionaremo String.

aluno.setCaminhoFoto((String) campoFoto.getTag());

Nosso código ficou assim:

public Aluno pegaAluno() {  
 aluno.setNome(campoNome.getText().toString());  
 aluno.setEndereco(campoEndereco.getText().toString());  
 aluno.setTelefone(campoTelefone.getText().toString());  
 aluno.setSite(campoSite.getText().toString());  
 aluno.setNota(Double.valueOf(campoNota.getProgress());  
 aluno.setCaminhoFoto((String) campoFoto.getTag());  
 return aluno;  
  
 }

Fizemos algumas alterações, mas se voltarmos ao aplicativo e tentarmos inserir uma imagem no perfil dos alunos novamente, a foto ainda não ficará no cadastro.

Isto acontece, porque não alteramos o método preencheFormulario do arquivo FormularioActivity.

@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) {  
 helper.preencheFormulario(aluno);  
 }

Agora, vamos até método preencheFormulario, do FormularioHelper. Também precisaremos criar um referência para o campo da foto.

Como isto é uma responsabilidade do FormularioHelper, então, iremos mover para ele, o código do onActivityResultque usávamos para preencher nossa imagem.

As seguintes linhas serão substituídas:

ImageView foto = (ImageView) findViewById(R.id.formulario\_foto);  
Bitmap bitmap = BitmapFactory.decodeFile(caminhoFoto);  
Bitmap bitmapReduzido = Bitmap.createScaledBitmap(bitmap, 300, 300, true);  
foto.setScaleType(ImageView.ScaleType.FIT\_XY);  
foto.setTag(caminhoFoto);

No lugar, iremos adicionar uma chamada para o helper e passar como parâmetro o caminho da imagem.

@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
 if (resultCode == Activity.RESULT\_OK) {  
 if (requestCode == CODIGO\_CAMERA) {  
 helper.carregaImagem(caminhoFoto);   
  
 }  
 }  
 }

Iremos colar no Helper, as linhas retiradas. Elas estarão no método que iremos criar carregaImagem.

public void carregaImagem(String caminhoFoto) {  
 ImageView foto = (ImageView) findViewById(R.id.formulario\_foto);  
 Bitmap bitmap = BitmapFactory.decodeFile(caminhoFoto);  
 Bitmap bitmapReduzido = Bitmap.createScaledBitmap(bitmap, 300, 300, true);  
 foto.setImageBitmap(bitmapReduzido);  
 foto.setScaleType(ImageView.ScaleType.FIT\_XY);  
 foto.setTag(caminhoFoto);  
}

Faremos ainda algumas alterações:

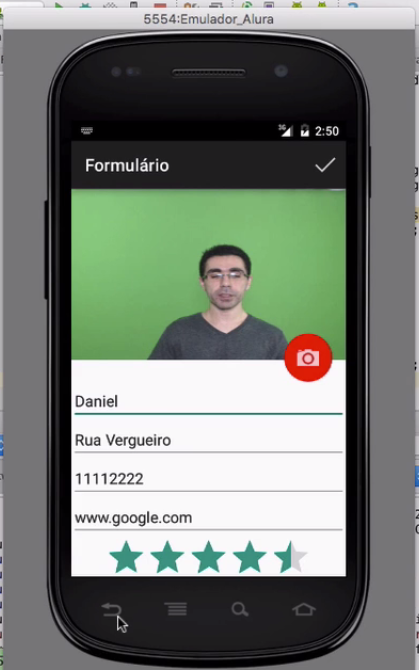
public void carregaImagem(String caminhoFoto) {  
 Bitmap bitmap = BitmapFactory.decodeFile(caminhoFoto);  
 Bitmap bitmapReduzido = Bitmap.createScaledBitmap(bitmap, 300, 300, true);  
 campofoto.setImageBitmap(bitmapReduzido);  
 campofoto.setScaleType(ImageView.ScaleType.FIT\_XY);  
 campofoto.setTag(caminhoFoto);  
 }

Como temos alunos que ainda não possuem fotos no cadastro, devemos verificar se o caminhoFoto é diferente de nulo.

public void carregaImagem(String caminhoFoto) {  
 if (caminhoFoto = != null) {  
 Bitmap bitmap = BitmapFactory.decodeFile(caminhoFoto);  
 Bitmap bitmapReduzido = Bitmap.createScaledBitmap(bitmap, 300, 300, true);  
 campofoto.setImageBitmap(bitmapReduzido);  
 campofoto.setScaleType(ImageView.ScaleType.FIT\_XY);  
 campofoto.setTag(caminhoFoto);  
 }  
}

No método preencheFormulario, vamos inserir carregaImagem e o caminho da foto.

Se tentarmos o aplicativo novamente, veremos que a foto tirada anteriormente já estará salva no perfil do aluno.



Com isto, concluímos a parte da câmera no nosso aplicativo.

Vamos revisar o que fizemos até aqui:

* Atualizamos o modelo de aluno, adicionando um campo para foto, criando getters e setters.
* No AlunoDAO, atualizamos os métodos de criação e atualização do nosso banco, inserindo uma coluna.
* Alteramos o FormularioHelper que transferia os dados do formulário para o banco e passava o aluno montado. Ele também preenchia os campos do formulário com o aluno.
* Com o método setTag associamos as imagens com o ImageView do Android.