Итак, давайте доработаем NotesProvider, чтобы он мог работать с изображениями по аналогии с заметками.

Дорабатываем ContentProvider

В первую очередь, добавим константы для UriMatcher:

private static final int IMAGES = 3;

private static final int IMAGE = 4;

Теперь свяжем URI с константами:

static {

URI\_MATCHER.addURI(NotesContract.AUTHORITY, "notes", NOTES);

URI\_MATCHER.addURI(NotesContract.AUTHORITY, "notes/#", NOTE);

URI\_MATCHER.addURI(NotesContract.AUTHORITY, "images", IMAGES);

URI\_MATCHER.addURI(NotesContract.AUTHORITY, "images/#", IMAGE);

}

Метод getType() выглядит вот так:

@Nullable

@Override

public String getType(@NonNull Uri uri) {

switch (URI\_MATCHER.match(uri)) {

case NOTES:

return NotesContract.Notes.URI\_TYPE\_NOTE\_DIR;

case NOTE:

return NotesContract.Notes.URI\_TYPE\_NOTE\_ITEM;

case IMAGES:

return NotesContract.Images.URI\_TYPE\_IMAGE\_DIR;

case IMAGE:

return NotesContract.Images.URI\_TYPE\_IMAGE\_ITEM;

default:

return null;

}

}

Теперь добавим получение изображений в метод query():

case IMAGES:

if (TextUtils.isEmpty(sortOrder)) {

sortOrder = NotesContract.Images.\_ID + " ASC";

}

return db.query(NotesContract.Images.TABLE\_NAME,

projection,

selection,

selectionArgs,

null,

null,

sortOrder);

Единственное отличие — сортировать будем по \_id по возрастанию (изображение, добавленное первым будет отображено первым).

Нам потребуется только получение списка — отдельное изображение получать не придётся.

Дополним метод insert():

case IMAGES:

long imageRowId = db.insert(NotesContract.Images.TABLE\_NAME,

null,

contentValues);

if (imageRowId > 0) {

Uri imageUri = ContentUris.withAppendedId(NotesContract.Images.URI, imageRowId);

getContext().getContentResolver().notifyChange(uri, null);

return imageUri;

}

return null;

Метод update() нам не потребуется.

Реализуем метод delete(), заодно добавим и удаление отдельной заметки:

@Override

public int delete(@NonNull Uri uri, @Nullable String selection, @Nullable String[] selectionArgs) {

SQLiteDatabase db = notesDbHelper.getWritableDatabase();

switch (URI\_MATCHER.match(uri)) {

case NOTE:

String noteId = uri.getLastPathSegment();

if (TextUtils.isEmpty(selection)) {

selection = NotesContract.Notes.\_ID + " = ?";

selectionArgs = new String[]{noteId};

} else {

selection = selection + " AND " + NotesContract.Notes.\_ID + " = ?";

String[] newSelectionArgs = new String[selectionArgs.length + 1];

System.arraycopy(selectionArgs, 0, newSelectionArgs, 0, selectionArgs.length);

newSelectionArgs[newSelectionArgs.length - 1] = noteId;

selectionArgs = newSelectionArgs;

}

int noteRowsUpdated = db.delete(NotesContract.Notes.TABLE\_NAME, selection, selectionArgs);

getContext().getContentResolver().notifyChange(uri, null);

return noteRowsUpdated;

case IMAGE:

String imageId = uri.getLastPathSegment();

if (TextUtils.isEmpty(selection)) {

selection = NotesContract.Images.\_ID + " = ?";

selectionArgs = new String[]{imageId};

} else {

selection = selection + " AND " + NotesContract.Images.\_ID + " = ?";

String[] newSelectionArgs = new String[selectionArgs.length + 1];

System.arraycopy(selectionArgs, 0, newSelectionArgs, 0, selectionArgs.length);

newSelectionArgs[newSelectionArgs.length - 1] = imageId;

selectionArgs = newSelectionArgs;

}

int imageRowsUpdated = db.delete(NotesContract.Images.TABLE\_NAME, selection, selectionArgs);

getContext().getContentResolver().notifyChange(uri, null);

return imageRowsUpdated;

}

return 0;

}

В принципе, почти ничем не отличается от обновления — разве что отличается метод.

Сохраняем изображение

Теперь мы можем сохранить изображение в БД. Происходит это следующим образом:

1. Сохраняем изображение в файл.
2. Добавляем запись в соответствующую таблицу.

В случае, если мы получаем изображение с камеры, картинка уже сохранена. Если же мы выбираем его из галереи, нам нужно скопировать изображение к себе. Для этого нам понадобится следующий метод:

private void writeInputStreamToFile(InputStream inputStream, File outFile) throws IOException {

FileOutputStream fileOutputStream = new FileOutputStream(outFile);

byte[] buffer = new byte[8192];

int n;

while ((n = inputStream.read(buffer)) > 0) {

fileOutputStream.write(buffer, 0, n);

}

fileOutputStream.flush();

fileOutputStream.close();

inputStream.close();

}

Теперь добавим метод записи в БД:

private void addImageToDatabase(File file) {

if (noteId == -1) {

// На данный момент мы добавляем аттачи только в режиме редактирования

return;

}

ContentValues contentValues = new ContentValues();

contentValues.put(NotesContract.Images.COLUMN\_PATH, file.getAbsolutePath());

contentValues.put(NotesContract.Images.COLUMN\_NOTE\_ID, noteId);

getContentResolver().insert(NotesContract.Images.URI, contentValues);

}

Как видите, в данный момент мы добавляем изображение в БД только в случае редактирования уже существующей заметки. Почему так?

Для создания записи изображения нам нужно знать идентификатор заметки, а у нас его нет, пока мы не создадим заметку.

Как решить эту проблему? Один из вариантов: копируем изображения, но не вставляем их, а сохраняем в List. После сохранения заметки вставляем в БД.

Вообще, по-хорошему мы должны всегда вставлять изображения только при сохранении заметки — ведь закрытие Activity без сохранения означает, что пользователь отказался от изменений.

**Задание:** сделайте это самостоятельно.

Теперь подправим метод onActivityResult():

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

super.onActivityResult(requestCode, resultCode, data);

if (requestCode == REQUEST\_CODE\_PICK\_FROM\_GALLERY

&& resultCode == RESULT\_OK

&& data != null) {

// Получаем URI изображения

Uri imageUri = data.getData();

if (imageUri != null) {

try {

// Получаем InputStream, из которого будем декодировать Bitmap

InputStream inputStream = getContentResolver().openInputStream(imageUri);

// Копируем изображение в наш файл

File imageFile = createImageFile();

writeInputStreamToFile(inputStream, imageFile);

addImageToDatabase(imageFile);

} catch (IOException e) {

e.printStackTrace();

}

}

} else if (requestCode == REQUEST\_CODE\_TAKE\_PHOTO

&& resultCode == RESULT\_OK) {

// Сохраняем изображение

addImageToDatabase(currentImageFile);

// На всякий случай обнуляем файл

currentImageFile = null;

}

}

Запустите приложение и попробуйте добавить изображение. Конечно же, вы ничего не увидите, так как мы ещё не отображаем изображения.

Работаем с несколькими CursorLoader

Итак, нам нужно отобразить изображения. Давайте сделаем это в горизонтальном RecyclerView внутри CreateNoteActivity. Но для того, чтобы загрузить список изображений, нам потребуется CursorLoader, не так ли? Верно. Но у нас уже есть один лоадер — как же добавить ещё один?

Всё просто: как вы, возможно заметили, у каждого лоадера есть уникальный идентификатор, по которому мы можем понять, с каким лоадером имеем дело.

Создадим идентификаторы для наших лоадеров:

private static final int LOADER\_NOTE = 0;

private static final int LOADER\_IMAGES = 1;

Добавим константу для инициализации лоадера заметки:

getLoaderManager().initLoader(

LOADER\_NOTE, // Идентификатор загрузчика

null, // Аргументы

this // Callback для событий загрузчика

);

Сразу после этого инициализируем лоадер для изображений:

getLoaderManager().initLoader(

LOADER\_IMAGES,

null,

this

);

А теперь изменим onCreateLoader():

@Override

public Loader<Cursor> onCreateLoader(int id, Bundle args) {

if (id == LOADER\_NOTE) {

return new CursorLoader(

this, // Контекст

ContentUris.withAppendedId(NotesContract.Notes.URI, noteId), // URI

NotesContract.Notes.SINGLE\_PROJECTION, // Столбцы

null, // Параметры выборки

null, // Аргументы выборки

null // Сортировка по умолчанию

);

} else {

return new CursorLoader(

this,

NotesContract.Images.URI,

NotesContract.Images.PROJECTION,

NotesContract.Images.COLUMN\_NOTE\_ID + " = ?",

new String[]{String.valueOf(noteId)},

null

);

}

}

И onLoadFinished():

@Override

public void onLoadFinished(Loader<Cursor> loader, Cursor cursor) {

if (loader.getId() == LOADER\_NOTE) {

cursor.setNotificationUri(getContentResolver(), NotesContract.Notes.URI);

displayNote(cursor);

} else {

cursor.setNotificationUri(getContentResolver(), NotesContract.Notes.URI);

// TODO: Отображение списка изображений

}

}

Добавим в разметку CreateNoteActivity RecyclerView (в конец LinearLayout):

<android.support.v7.widget.RecyclerView

android:id="@+id/images\_rv"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="16dp" />

Кроме того, нужно обязательно заменить ScrollView на NestedScrollView:

<android.support.v4.widget.NestedScrollView

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior">

Дело в том, что нельзя использовать скроллящиеся View внутри ScrollView, и NestedScrollView создан как раз для решения этой проблемы.

Создадим файл разметки для отображения картинки:

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

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

android:layout\_width="90dp"

android:layout\_height="90dp"

android:layout\_margin="8dp"

android:adjustViewBounds="true" />

Да, он состоит из единственного элемента — ImageView.

Теперь создаём адаптер:

public class NoteImagesAdapter extends CursorRecyclerAdapter<NoteImagesAdapter.ViewHolder> {

public NoteImagesAdapter(Cursor cursor) {

super(cursor);

}

@Override

public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

LayoutInflater layoutInflater = LayoutInflater.from(parent.getContext());

View view = layoutInflater.inflate(R.layout.view\_item\_note\_image, parent, false);

return new ViewHolder(view);

}

@Override

public void onBindViewHolder(ViewHolder viewHolder, Cursor cursor) {

long imageId = cursor.getLong(cursor.getColumnIndexOrThrow(NotesContract.Images.\_ID));

String imagePath = cursor.getString(cursor.getColumnIndexOrThrow(NotesContract.Images.COLUMN\_PATH));

Bitmap bitmap = BitmapFactory.decodeFile(imagePath);

viewHolder.imageView.setImageBitmap(bitmap);

viewHolder.itemView.setTag(imageId);

}

/\*\*

\* ViewHolder

\*/

class ViewHolder extends RecyclerView.ViewHolder {

private ImageView imageView;

public ViewHolder(View itemView) {

super(itemView);

imageView = (ImageView) itemView;

}

}

}

Тут, в принципе, ничего нового, так что разбирать подробно не будем.

В CreateNoteActivity инициализируем RecyclerView:

RecyclerView recyclerView = findViewById(R.id.images\_rv);

recyclerView.setLayoutManager(new LinearLayoutManager(this, LinearLayoutManager.HORIZONTAL, false));

noteImagesAdapter = new NoteImagesAdapter(null);

recyclerView.setAdapter(noteImagesAdapter);

И передаём курсор в адаптер внутри onLoadFinished():

@Override

public void onLoadFinished(Loader<Cursor> loader, Cursor cursor) {

if (loader.getId() == LOADER\_NOTE) {

cursor.setNotificationUri(getContentResolver(), NotesContract.Notes.URI);

displayNote(cursor);

} else {

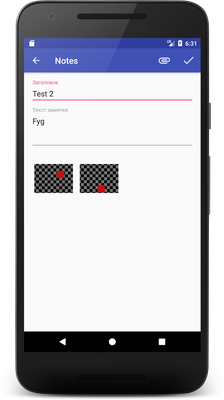
cursor.setNotificationUri(getContentResolver(), NotesContract.Images.URI);

noteImagesAdapter.swapCursor(cursor);

}

}

Готово! Попробуйте добавить изображения в какую-нибудь заметку:

[](https://skillberg.com/media/uploads/2017/11/17/03a5abc4f39d40599e4f0a7c9b33eb9b.png)Список приаттаченных изображений

Переиспользуем код

Помимо отображения изображений при редактировании заметки нам нужно показывать их и при просмотре заметки.

Поскольку код и там и там очень похож, было бы неплохо не дублировать код в двух Activity.

Давайте создадим абстрактную Activity, которая будет уметь загружать заметку и изображения, и наследуем от неё остальные две Activity.

Класс будет абстрактным, чтобы его нельзя было инстанциировать напрямую. Реализация будет достаточно простой — я банально скопировал нужный код из CreateNoteActivity:

public abstract class BaseNoteActivity extends AppCompatActivity implements LoaderManager.LoaderCallbacks<Cursor> {

protected static final int LOADER\_NOTE = 0;

protected static final int LOADER\_IMAGES = 1;

protected long noteId = -1;

protected NoteImagesAdapter noteImagesAdapter;

/\*\*

\* Инициализируем загрузчик заметки

\*/

protected void initNoteLoader() {

getLoaderManager().initLoader(

LOADER\_NOTE, // Идентификатор загрузчика

null, // Аргументы

this // Callback для событий загрузчика

);

}

/\*\*

\* Инициализируем загрузчик изображений

\*/

protected void initImagesLoader() {

getLoaderManager().initLoader(

LOADER\_IMAGES,

null,

this

);

}

@Override

public Loader<Cursor> onCreateLoader(int id, Bundle args) {

if (id == LOADER\_NOTE) {

return new CursorLoader(

this, // Контекст

ContentUris.withAppendedId(NotesContract.Notes.URI, noteId), // URI

NotesContract.Notes.SINGLE\_PROJECTION, // Столбцы

null, // Параметры выборки

null, // Аргументы выборки

null // Сортировка по умолчанию

);

} else {

return new CursorLoader(

this,

NotesContract.Images.URI,

NotesContract.Images.PROJECTION,

NotesContract.Images.COLUMN\_NOTE\_ID + " = ?",

new String[]{String.valueOf(noteId)},

null

);

}

}

@Override

public void onLoadFinished(Loader<Cursor> loader, Cursor cursor) {

if (loader.getId() == LOADER\_NOTE) {

cursor.setNotificationUri(getContentResolver(), NotesContract.Notes.URI);

displayNote(cursor);

} else {

cursor.setNotificationUri(getContentResolver(), NotesContract.Images.URI);

noteImagesAdapter.swapCursor(cursor);

}

}

@Override

public void onLoaderReset(Loader<Cursor> loader) {

}

/\*\*

\* Отображаем заметку. Этот метод должен быть реализован в Activity

\*/

protected abstract void displayNote(Cursor cursor);

}

Теперь наследуем CreateNoteActivity от BaseNoteActivity, попутно удалив ненужные методы.

Методы, которые остались без изменений:

* addImageToDatabase()
* writeInputStreamToFile()
* takePhoto()
* pickImageFromGallery()
* createImageFile()
* showImageSelectionDialog()
* saveNote()
* onActivityResult()
* onOptionsItemSelected()
* onCreateOptionsMenu()

Код, который изменился:

public class CreateNoteActivity extends BaseNoteActivity {

public static final String EXTRA\_NOTE\_ID = "note\_id";

private static final int REQUEST\_CODE\_PICK\_FROM\_GALLERY = 1;

private static final int REQUEST\_CODE\_TAKE\_PHOTO = 2;

private TextInputEditText titleEt;

private TextInputEditText textEt;

private TextInputLayout titleTil;

private TextInputLayout textTil;

private File currentImageFile;

@Override

protected void onCreate(@Nullable Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_create\_note);

Toolbar toolbar = findViewById(R.id.toolbar);

setSupportActionBar(toolbar);

getSupportActionBar().setDisplayHomeAsUpEnabled(true);

titleEt = findViewById(R.id.title\_et);

textEt = findViewById(R.id.text\_et);

titleTil = findViewById(R.id.title\_til);

textTil = findViewById(R.id.text\_til);

RecyclerView recyclerView = findViewById(R.id.images\_rv);

recyclerView.setLayoutManager(new LinearLayoutManager(this, LinearLayoutManager.HORIZONTAL, false));

noteImagesAdapter = new NoteImagesAdapter(null);

recyclerView.setAdapter(noteImagesAdapter);

noteId = getIntent().getLongExtra(EXTRA\_NOTE\_ID, -1);

if (noteId != -1) {

initNoteLoader();

initImagesLoader();

}

}

/\*\*

\* Отображаем данные из курсора

\*/

@Override

protected void displayNote(Cursor cursor) {

if (!cursor.moveToFirst()) {

// Если не получилось перейти к первой строке — завершаем Activity

finish();

}

String title = cursor.getString(cursor.getColumnIndexOrThrow(NotesContract.Notes.COLUMN\_TITLE));

String noteText = cursor.getString(cursor.getColumnIndexOrThrow(NotesContract.Notes.COLUMN\_NOTE));

titleEt.setText(title);

textEt.setText(noteText);

}

// ...

Запустите приложение, чтобы убедиться, что всё работает.

Отлично, теперь используем этот же код в NoteActivity:

public class NoteActivity extends BaseNoteActivity {

public static final String EXTRA\_NOTE\_ID = "note\_id";

private TextView noteTv;

@Override

protected void onCreate(@Nullable Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_note);

Toolbar toolbar = findViewById(R.id.toolbar);

setSupportActionBar(toolbar);

getSupportActionBar().setDisplayHomeAsUpEnabled(true);

noteTv = findViewById(R.id.text\_tv);

RecyclerView recyclerView = findViewById(R.id.images\_rv);

recyclerView.setLayoutManager(new LinearLayoutManager(this, LinearLayoutManager.HORIZONTAL, false));

noteImagesAdapter = new NoteImagesAdapter(null);

recyclerView.setAdapter(noteImagesAdapter);

noteId = getIntent().getLongExtra(EXTRA\_NOTE\_ID, -1);

if (noteId != -1) {

initNoteLoader();

initImagesLoader();

} else {

finish();

}

}

/\*\*

\* Отображаем данные из курсора

\*/

@Override

protected void displayNote(Cursor cursor) {

if (!cursor.moveToFirst()) {

// Если не получилось перейти к первой строке — завершаем Activity

finish();

return;

}

String title = cursor.getString(cursor.getColumnIndexOrThrow(NotesContract.Notes.COLUMN\_TITLE));

String noteText = cursor.getString(cursor.getColumnIndexOrThrow(NotesContract.Notes.COLUMN\_NOTE));

setTitle(title);

noteTv.setText(noteText);

}

// ...

Методы, которые не изменились:

* onCreateOptionsMenu()
* onOptionsItemSelected()
* editNote()

Само собой, нужно так же добавить RecyclerView в разметку и заменить ScrollView на NestedScrollView.

Запустите приложение:

[](https://skillberg.com/media/uploads/2017/11/17/e69a4a4c062f4a35ad877a25de78db8c.png)Отображение изображений в заметке

Отлично! В следующем уроке мы завершим работу над этим приложением, реализовав удаление заметок.

ПРИМЕРЫ КОДА

[ИСХОДНЫЙ КОД УРОКА](https://github.com/Skillberg/NotesAndroid/tree/lesson-27)

ДОМАШНЕЕ ЗАДАНИЕ

Реализуйте добавление изображений в заметку при сохранении заметки, а не сразу после выбора.<br/>Также должна быть возможность добавить изображения в новую заметку.