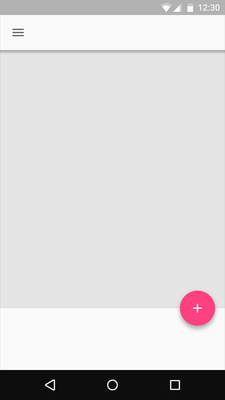
В этом уроке мы с вами научимся создавать и удалять заметки, при этом продолжая изучать Material Design.

FloatingActionButton

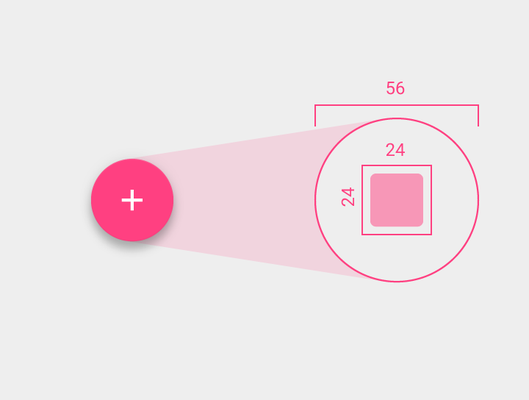
FloatingActionButton — один из часто используемых компонентов Material Design.

Как ясно из названия, это кнопка, которая висит над контентом и предназначена для выполнения основного действия на текущем экране.

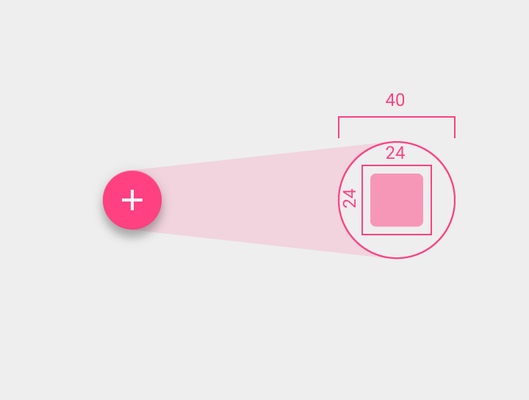
[](https://skillberg.com/media/uploads/2017/11/17/705eac5a92d74b20b4b0fa0f7606d02c.png)FloatingActionButton в Android

Размеры FloatingActionButton

Эта кнопка может быть двух размеров: **стандартного** или **маленького**.

[](https://skillberg.com/media/uploads/2017/11/17/a37fe96aa87147e18aa0817d15210a77.png)FloatingActionButton стандартного размера

В случае использования стандартного варианта, FloatingActionButton имеет размер 56x56 dp. Размер иконки внутри кнопки — 24x24 dp.

[](https://skillberg.com/media/uploads/2017/11/17/a1ad9ae2634b4b30a029cfc76e85bd13.png)FloatingActionButton маленького размера

Уменьшенный вариант имеет размер 40x40 dp, при этом размер иконки внутри кнопки остаётся неизменным — 24x24 dp.

В обоих вариантах расстояние от краёв экрана не должно быть меньше 16 dp на телефонах и 24 dp на планшетах и десктопах.

Использование FloatingActionButton

В нашем приложении мы будем использовать FloatingActionButton для создания заметки.

Давайте же приступим к практике. Откройте файл activity\_main.xml и измените его следующим образом:

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

<android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context="com.skillberg.notes.MainActivity">

<android.support.design.widget.AppBarLayout

android:layout\_width="match\_parent"

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

<include layout="@layout/toolbar" />

</android.support.design.widget.AppBarLayout>

<android.support.v7.widget.RecyclerView

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

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

<android.support.design.widget.FloatingActionButton

android:id="@+id/create\_fab"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_margin="16dp"

app:layout\_anchor="@id/notes\_rv"

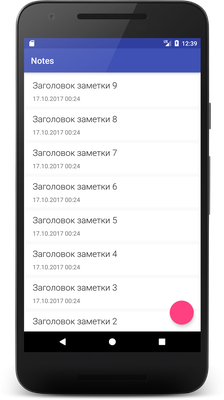
app:layout\_anchorGravity="bottom|end" />

</android.support.design.widget.CoordinatorLayout>

Среди атрибутов вы видите два незнакомых ранее:

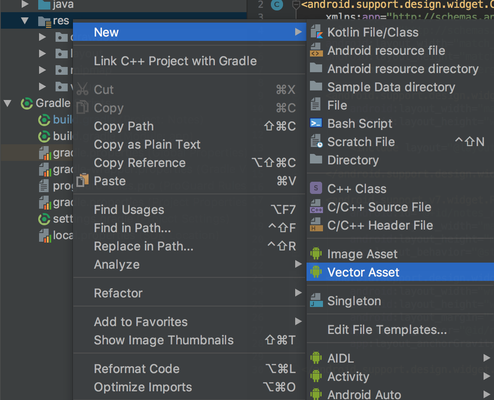
* layout\_anchor указывает на View, относительно которого будет показываться FloatingActionButton. Мы ссылаемся на RecyclerView.
* layout\_anchorGravity задаёт позиционирование относительно того элемента, на который мы ссылаемся. В нашем случае кнопка будет отображаться внизу справа (или слева на RTL локалях) относительно RecyclerView.

Запустите приложение:

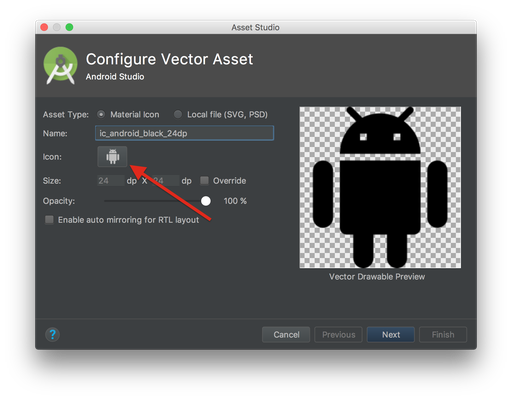
[](https://skillberg.com/media/uploads/2017/11/17/96ffa780879f4997a1052f656a520a1e.png)FloatingActionButton в нашем приложении

Как видите, кнопка отображается, но на ней нет иконки.

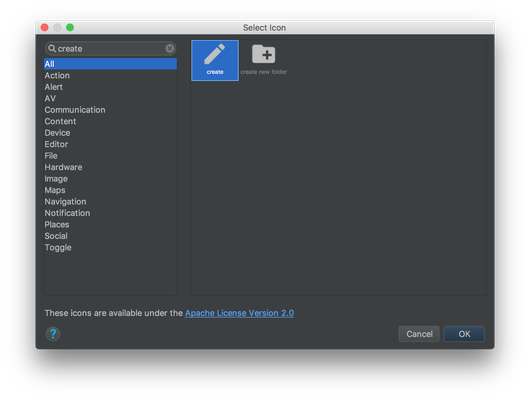
Для начала нужно создать иконку. Мы уже делали это ранее, но теперь для примера создадим векторную иконку:

[](https://skillberg.com/media/uploads/2017/11/17/2fa4b0faf79b4ca8b824a99d71d36156.png)Создание векторной иконки

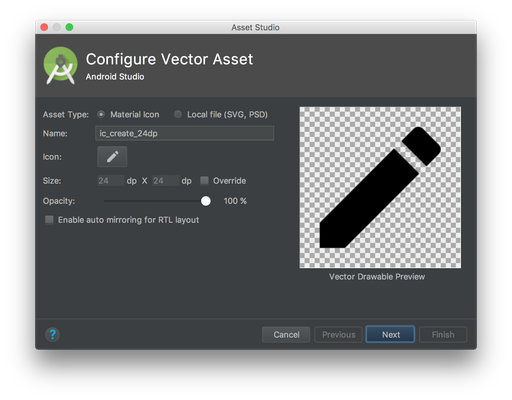
Нажмите на кнопку выбора иконки:

[](https://skillberg.com/media/uploads/2017/11/17/fe9125de77c74fd0a3190cdbf6140253.png)Кнопка выбора иконки

Далее выберите иконку **"Create"**:

[](https://skillberg.com/media/uploads/2017/11/17/b362b8fd0f0240058de5ae257a008ea6.png)Иконка "Create"

Размер оставляем дефолтным. В итоге получим вот такое окно:

[](https://skillberg.com/media/uploads/2017/11/17/c24ebabe3742493d92421afbe9ce96aa.png)

Иконка создана, теперь установим её во FloatingActionButton. Но прежде нам нужно решить одну проблему: векторные изображения поддерживаются только на **Android Lollipop** и выше, а на более ранних версиях они просто так работать не будут.

Чтобы заставить их работать, нужно использовать библиотеку AppCompat, и предварительно включить поддержку AppCompat векторных изображений.

Для этого откройте build.gradle и добавьте следующую строку в секцию defaultConfig: vectorDrawables.useSupportLibrary = true.

В итоге ваш build.gradle должен выглядеть примерно так:

apply plugin: 'com.android.application'

android {

compileSdkVersion 26

buildToolsVersion '26.0.2'

defaultConfig {

applicationId "com.skillberg.notes"

minSdkVersion 17

targetSdkVersion 26

versionCode 1

versionName "1.0"

vectorDrawables.useSupportLibrary = true

testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"

}

buildTypes {

release {

minifyEnabled false

proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'

}

}

}

dependencies {

implementation fileTree(dir: 'libs', include: ['\*.jar'])

implementation 'com.android.support:appcompat-v7:26.1.0'

implementation 'com.android.support.constraint:constraint-layout:1.0.2'

implementation 'com.android.support:recyclerview-v7:26.1.0'

implementation 'com.android.support:design:26.1.0'

implementation 'com.android.support:cardview-v7:26.1.0'

testImplementation 'junit:junit:4.12'

androidTestImplementation('com.android.support.test.espresso:espresso-core:3.0.1', {

exclude group: 'com.android.support', module: 'support-annotations'

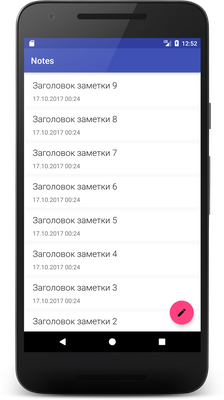
})

}

Отлично, теперь в activity\_main.xml добавьте к FloatingActionButton вот такой атрибут:

app:srcCompat="@drawable/ic\_create\_24dp"

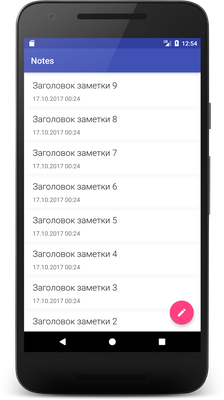
Запустите приложение:

[](https://skillberg.com/media/uploads/2017/11/17/104ad3ea6a4e4a7891de261767023d52.png)Иконка в FloatingActionButton

Неплохо, но иконка чёрная, а должна быть белой.

Это не беда — поскольку FloatingActionButton — это, по сути, навороченный ImageView, мы можем изменить цвет иконки, используя атрибут android:tint. Добавьте его к кнопке, и посмотрите, что получится:

android:tint="@android:color/white"

[](https://skillberg.com/media/uploads/2017/11/17/5cd643fc250b4849ada9917761e1b4fd.png)FloatingActionButton с белой иконкой

Супер!

Обработка нажатий на FloatingActionButton

Обработка нажатий на FloatingActionButton работает точно так же, как и на любых других View.

В конце метода MainActivity.onCreate() найдите кнопку и добавьте на неё обработчик кликов:

findViewById(R.id.create\_fab).setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

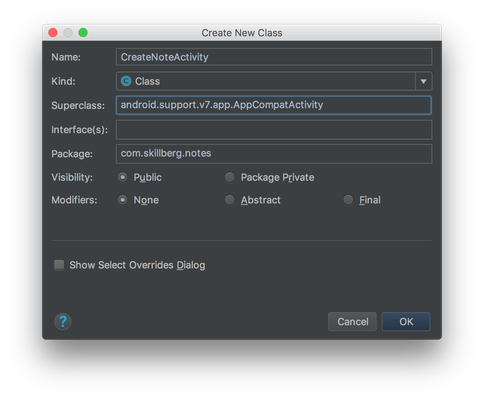
// Здесь будет обрабатываться клик

}

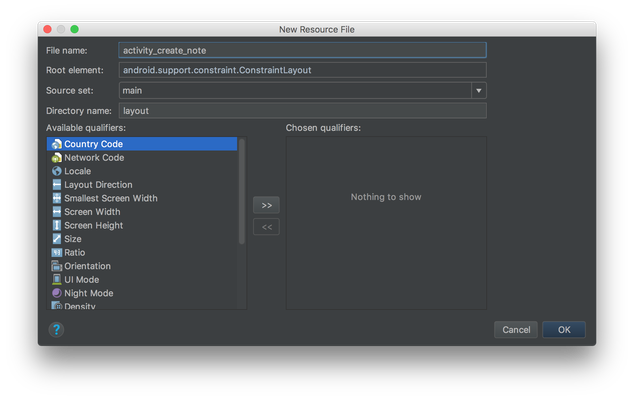
});

Что мы будем делать при нажатии? Просто запустим новую Activity, в которой будет создаваться заметка.

Создайте новую Activity:

[](https://skillberg.com/media/uploads/2017/11/17/f4706c6622e94dd6ac9d517ef4b1f455.png)Создание новой Activity

Так же нужно создать файл вёрстки для Activity:

[](https://skillberg.com/media/uploads/2017/11/17/03c57e998d484f02be58578272ce4f00.png)

Этот файл будет содержать следующее:

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

<android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context="com.skillberg.notes.CreateNoteActivity">

<android.support.design.widget.AppBarLayout

android:layout\_width="match\_parent"

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

<include layout="@layout/toolbar" />

</android.support.design.widget.AppBarLayout>

</android.support.design.widget.CoordinatorLayout>

В CreateNoteActivity установим его в качестве **ContentView** и зададим Toolbar:

public class CreateNoteActivity extends AppCompatActivity {

@Override

protected void onCreate(@Nullable Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

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

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

setSupportActionBar(toolbar);

}

}

Ну и, конечно, не забудьте зарегистрировать Activity в манифесте:

<activity android:name=".CreateNoteActivity" />

Теперь мы можем запустить CreateNoteActivity в обработчике кликов FloatingActionButton в MainActivity:

@Override

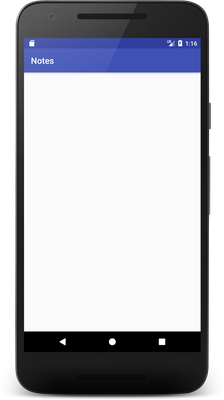
public void onClick(View v) {

Intent intent = new Intent(MainActivity.this, CreateNoteActivity.class);

startActivity(intent);

}

При нажатии на кнопку Activity запустится:

[](https://skillberg.com/media/uploads/2017/11/17/6e643bd54e734822be5184b777476760.png)CreateNoteActivity, запущенная нажатием на FloatingActionButton

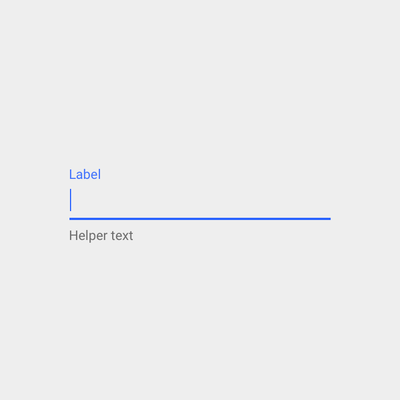
Отлично!

Создание заметки

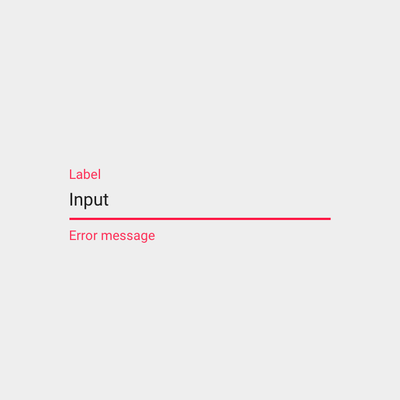
Для того, чтобы создать новую заметку, нам нужен интерфейс. Мы обойдёмся двумя уже знакомыми нам View — EditText. Один из них будет для заголовка заметки, второй — для текста.

TextInputLayout

Чтобы сделать процесс ввода данных более очевидным для пользователя, принято использовать подсказки. Они есть и в обычном EditText, однако, будет правильным использовать вот такие поля:

[](https://skillberg.com/media/uploads/2017/11/17/6eca1ee3b82a41fc8c81a93fdc1c9c60.png)TextInputLayout

Они понятны для пользователя, содержат подсказки и, что немаловажно, могут отображать сообщения об ошибках:

[](https://skillberg.com/media/uploads/2017/11/17/147eb146b2b94a3f92cc4d0bbd69248a.png)TextInputLayout с сообщением об ошибке

Создаются они совсем несложно — в Support-библиотеке есть специальный виджет TextInputLayout.

В activity\_create\_note.xml создайте контейнер для наших будущих View. Для этого вполне подойдёт вертикальный LinearLayout.

Поскольку мы не знаем, сколько текста будет вводить пользователь (высота текста может легко превысить высоту экрана), будет не лишним обернуть LinearLayout в ScrollView, который позволит скроллить контент:

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

<android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context="com.skillberg.notes.CreateNoteActivity">

<android.support.design.widget.AppBarLayout

android:layout\_width="match\_parent"

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

<include layout="@layout/toolbar" />

</android.support.design.widget.AppBarLayout>

<ScrollView

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

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

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical">

</LinearLayout>

</ScrollView>

</android.support.design.widget.CoordinatorLayout>

Обратите внимание, что ни в коем случае нельзя задавать высоту дочернего элемента ScrollView равной match\_parent, так как по сути внутренняя высота ScrollView бесконечна.

Кроме того, нельзя добавлять в ScrollView более одного дочернего View. Если нужно добавить несколько дочерних View, нужно добавлять их в контейнере (как мы сейчас и делаем).

Внутрь LinearLayout добавьте следующую разметку:

<android.support.design.widget.TextInputLayout

android:id="@+id/title\_til"

android:layout\_width="match\_parent"

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

<android.support.design.widget.TextInputEditText

android:id="@+id/title\_et"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="@string/hint\_title"

android:inputType="textCapSentences" />

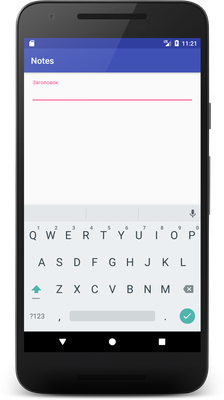
</android.support.design.widget.TextInputLayout>

Как видите, TextInputLayout служит контейнером для EditText.

Кроме того, вы могли обратить внимание, что мы используем TextInputEditText, а не просто EditText. Он почти ничем не отличается от обычного EditText, кроме улучшенной поддержки TextInputLayout.

В атрибут android:hint установлен текст подсказки (я заранее вынес его в ресурсы). Значение textCapSentences в атрибуте android:inputType говорит о том, что первая буква каждого предложения будет заглавной.

Запустив приложение, вы увидите следующее:

[](https://skillberg.com/media/uploads/2017/11/17/17fb2d3c27554ae0a4ed09c3987a7a69.png)TextInputEditText внутри TextInputLayout

Подсказка поднялась вверх. Если фокус из поля пропадёт, то она сместится вниз и будет выглядеть так же, как и в обычном EditText.

Теперь добавим поле для текста. Оно будет точно таким же, но с поддержкой многострочного ввода и по умолчанию будет содержать несколько строк.

Ниже поля для заголовка добавьте следующее:

<android.support.design.widget.TextInputLayout

android:id="@+id/text\_til"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="8dp">

<android.support.design.widget.TextInputEditText

android:id="@+id/text\_et"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:gravity="start|top"

android:hint="@string/hint\_text"

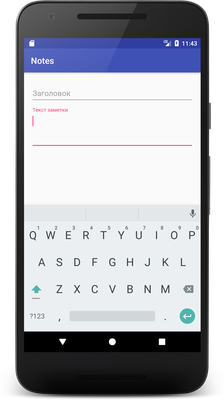
android:inputType="textCapSentences|textMultiLine"

android:lines="3"

android:maxLines="999" />

</android.support.design.widget.TextInputLayout>

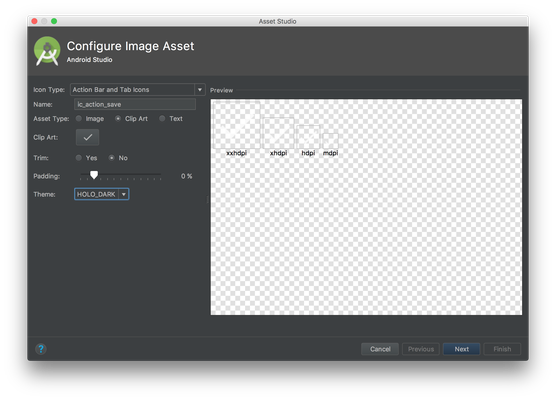
Разница в том, что мы задали дефолтное количество строк (3), максимальное количество строк и сделали EditText многострочным, добавив флаг textMultiLine к атрибуту android:inputType.

[](https://skillberg.com/media/uploads/2017/11/17/2401ade26c5a492ca96dc4e1543549a2.png)Многострочный TextInputEditText внутри TextInputLayout

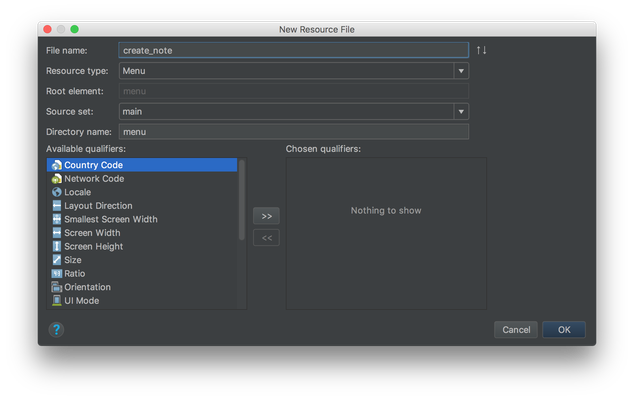
Кнопка сохранения

Сохранять заметку мы будем кнопкой в виде галочки в тулбаре.

Для начала создадим саму иконку (растровую). Как это делать вы уже знаете. Параметры будут такими:

[](https://skillberg.com/media/uploads/2017/11/17/25da8e48925c4c20b922855926c446ae.png)Создаём иконку сохранения заметки

Теперь создадим меню:

[](https://skillberg.com/media/uploads/2017/11/17/b339eca2f9b441be9b4756895364e940.png)Создаём меню для сохранения заметки

Содержимое меню будет таким:

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

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

xmlns:app="http://schemas.android.com/apk/res-auto">

<item

android:id="@+id/action\_save"

android:icon="@drawable/ic\_action\_save"

android:title="@string/title\_save"

app:showAsAction="ifRoom" />

</menu>

Теперь создадим меню в CreateNoteActivity:

@Override

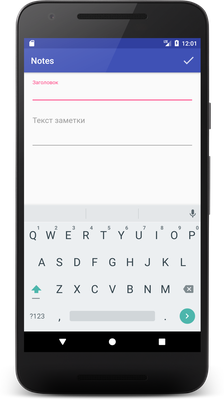
public boolean onCreateOptionsMenu(Menu menu) {

super.onCreateOptionsMenu(menu);

getMenuInflater().inflate(R.menu.create\_note, menu);

return true;

}

[](https://skillberg.com/media/uploads/2017/11/17/ad4bbc964fc54dcb9c945fa369541062.png)Иконка сохранения в меню

Создайте пока что пустой метод сохранения заметки:

private void saveNote() {

}

Ну и, конечно, метод для обработки клика на элемент меню, в котором будет вызван метод saveNote().

@Override

public boolean onOptionsItemSelected(MenuItem item) {

switch (item.getItemId()) {

case R.id.action\_save:

saveNote();

return true;

default:

return super.onOptionsItemSelected(item);

}

}

Проверка текста заметки и отображение ошибки

Прежде, чем сохранить заметку, нужно проверить, заполнил ли пользователь все нужные поля.

Конечно же, сначала нужно найти поля EditText, и, заодно, TextInputLayout — они будут нужны нам, чтобы отобразить ошибку.

private TextInputEditText titleEt;

private TextInputEditText textEt;

private TextInputLayout titleTil;

private TextInputLayout textTil;

lang=java

@Override

protected void onCreate(@Nullable Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

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

// ...

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

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

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

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

}

В методе saveNote() проверим, что поля заполнены, и, если это не так — покажем ошибку:

private void saveNote() {

String title = titleEt.getText().toString().trim();

String text = textEt.getText().toString().trim();

boolean isCorrect = true;

if (TextUtils.isEmpty(title)) {

isCorrect = false;

titleTil.setError(getString(R.string.error\_empty\_field));

titleTil.setErrorEnabled(true);

} else {

titleTil.setErrorEnabled(false);

}

if (TextUtils.isEmpty(text)) {

isCorrect = false;

textTil.setError(getString(R.string.error\_empty\_field));

textTil.setErrorEnabled(true);

} else {

textTil.setErrorEnabled(false);

}

if (isCorrect) {

long currentTime = System.currentTimeMillis();

ContentValues contentValues = new ContentValues();

contentValues.put(NotesContract.Notes.COLUMN\_TITLE, title);

contentValues.put(NotesContract.Notes.COLUMN\_NOTE, text);

contentValues.put(NotesContract.Notes.COLUMN\_CREATED\_TS, currentTime);

contentValues.put(NotesContract.Notes.COLUMN\_UPDATED\_TS, currentTime);

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

finish();

}

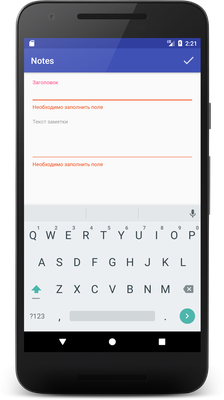
}

Тут всё достаточно просто: проверяем, не пустой ли текст в поле, и, если он пустой — показываем сообщение об ошибке.

Для отображения ошибки мы используем два метода:

* setError() — устанавливает текст ошибки (я перенёс текст в ресурсы заранее)
* setErrorEnabled() — если передать в этот метод true, текст ошибки будет показан, если же false, то он будет скрыт.

Если не ввести текст, и нажать на кнопку сохранения, получим сообщение об ошибке:

[](https://skillberg.com/media/uploads/2017/11/17/f87af3f38aaa4d02854e576526072da6.png)Отображение ошибки в TextInputLayout

Если же все данные корректны, вставляем заметку в БД и завершаем работу Activity.

Попробуйте ввести данные в поля и сохранить заметку, и вы увидите, что она успешно сохранилась и отображается в списке! :)

В следующем уроке мы будем отображать и редактировать заметки.

ПРИМЕРЫ КОДА

[ИСХОДНЫЙ КОД УРОКА](https://github.com/Skillberg/NotesAndroid/tree/lesson-24)

ДОМАШНЕЕ ЗАДАНИЕ

Перенесите создание заметки в фоновый поток.