# 8. Хранение данных

## Возможные варианты хранения данных

* [Internal file storage](https://developer.android.com/guide/topics/data/data-storage#filesInternal): Store app-private files on the device file system.
* [External file storage](https://developer.android.com/guide/topics/data/data-storage#filesExternal): Store files on the shared external file system. This is usually for shared user files, such as photos.
* [Shared preferences](https://developer.android.com/guide/topics/data/data-storage#pref): Store private primitive data in key-value pairs.
* [Databases](https://developer.android.com/guide/topics/data/data-storage#db): Store structured data in a private database.

Рассказать про каждый вид, плюсы и минусы.  
  
**Internal storage:**

* It's always available.
* Files saved here are accessible by only your app.
* When the user uninstalls your app, the system removes all your app's files from internal storage.

Internal storage is best when you want to be sure that neither the user nor other apps can access your files.

**External storage:**

* It's not always available, because the user can mount the external storage as USB storage and in some cases remove it from the device.
* It's world-readable, so files saved here may be read outside of your control.
* When the user uninstalls your app, the system removes your app's files from here only if you save them in the directory from [getExternalFilesDir()](https://developer.android.com/reference/android/content/Context.html#getExternalFilesDir(java.lang.String)).

## 

## Сохранение файла во внутреннюю память

When saving a file to internal storage, you can acquire the appropriate directory as a [File](https://developer.android.com/reference/java/io/File.html) by calling one of two methods:

[getFilesDir()](https://developer.android.com/reference/android/content/Context.html#getFilesDir())

Returns a [File](https://developer.android.com/reference/java/io/File.html) representing an internal directory for your app.

[getCacheDir()](https://developer.android.com/reference/android/content/Context.html#getCacheDir())

Returns a [File](https://developer.android.com/reference/java/io/File.html) representing an internal directory for your app's temporary cache files. Be sure to delete each file once it is no longer needed and implement a reasonable size limit for the amount of memory you use at any given time, such as 1MB.

**Caution:** If the system runs low on storage, it may delete your cache files without warning.

Чтобы создать файл:

File file = new File(context.getFilesDir(), filename);

Сохранить строку в файл

String filename = "myfile";

String fileContents = "Hello world!";

FileOutputStream outputStream;

try {

outputStream = openFileOutput(filename, Context.MODE\_PRIVATE);

outputStream.write(fileContents.getBytes());

outputStream.close();

} catch (Exception e) {

e.printStackTrace();

}

Рассказать про флаг последний

Notice that the [openFileOutput()](https://developer.android.com/reference/android/content/Context.html#openFileOutput(java.lang.String,%20int)) method requires a file mode parameter. Passing [MODE\_PRIVATE](https://developer.android.com/reference/android/content/Context.html#MODE_PRIVATE) makes it private to your app. The other mode options, [MODE\_WORLD\_READABLE](https://developer.android.com/reference/android/content/Context.html#MODE_WORLD_READABLE) and [MODE\_WORLD\_WRITEABLE](https://developer.android.com/reference/android/content/Context.html#MODE_WORLD_WRITEABLE), have been deprecated since API level 17. Starting with Android 7.0 (API level 24), Android throws a [SecurityException](https://developer.android.com/reference/java/lang/SecurityException.html) if you use them. If your app needs to share private files with other apps, you should instead use a [FileProvider](https://developer.android.com/reference/android/support/v4/content/FileProvider.html) with the [FLAG\_GRANT\_READ\_URI\_PERMISSION](https://developer.android.com/reference/android/content/Intent.html#FLAG_GRANT_READ_URI_PERMISSION). For more information, see [Sharing Files](https://developer.android.com/training/secure-file-sharing/index.html).

Создание файла в кеше и сохранение туда строки:

private File getTempFile(Context context, String url) {

File file;

try {

String fileName = Uri.parse(url).getLastPathSegment();

file = File.createTempFile(fileName, null, context.getCacheDir());

} catch (IOException e) {

// Error while creating file

}

return file;

}

### Open an existing file

To read an existing file, call [openFileInput(name)](https://developer.android.com/reference/android/content/Context.html#openFileInput(java.lang.String)), passing the name of the file.

### Open a directory

You can open a directory on the internal file system with the following methods:

[getFilesDir()](https://developer.android.com/reference/android/content/Context.html#getFilesDir())

Returns a [File](https://developer.android.com/reference/java/io/File.html) representing the directory on the file system that's uniquely associated with your app.

[getDir(name, mode)](https://developer.android.com/reference/android/content/Context.html#getDir(java.lang.String,%20int))

Creates a new directory (or opens an existing directory) within your app's unique file system directory. This new directory appears inside the directory provided by [getFilesDir()](https://developer.android.com/reference/android/content/Context.html#getFilesDir()).

[getCacheDir()](https://developer.android.com/reference/android/content/Context.html#getCacheDir())

Returns a [File](https://developer.android.com/reference/java/io/File.html) representing the cache directory on the file system that's uniquely associated with your app. This directory is meant for temporary files, and it should be cleaned up regularly. The system may delete files there if it runs low on disk space, so make sure you check for the existence of your cache files before reading them.

## Сохранение файла во внешнюю память

### Request external storage permissions

<manifest ...>

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

...

</manifest>

<manifest ...>

<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />

...

</manifest>

### Verify that external storage is available

/\* Checks if external storage is available for read and write \*/

public boolean isExternalStorageWritable() {

String state = Environment.getExternalStorageState();

if (Environment.MEDIA\_MOUNTED.equals(state)) {

return true;

}

return false;

}

### Save to a public directory

If you want to save public files on the external storage, use the [getExternalStoragePublicDirectory()](https://developer.android.com/reference/android/os/Environment.html#getExternalStoragePublicDirectory(java.lang.String)) method to get a [File](https://developer.android.com/reference/java/io/File.html) representing the appropriate directory on the external storage. The method takes an argument specifying the type of file you want to save so that they can be logically organized with other public files, such as [DIRECTORY\_MUSIC](https://developer.android.com/reference/android/os/Environment.html#DIRECTORY_MUSIC) or[DIRECTORY\_PICTURES](https://developer.android.com/reference/android/os/Environment.html#DIRECTORY_PICTURES). For example:

### 

public File getPublicAlbumStorageDir(String albumName) {

// Get the directory for the user's public pictures directory.

File file = new File(Environment.getExternalStoragePublicDirectory(

Environment.DIRECTORY\_PICTURES), albumName);

if (!file.mkdirs()) {

Log.e(LOG\_TAG, "Directory not created");

}

return file;

}

## Delete a file

myFile.delete();

## 

## Сохранение в файл ключ-значение

Рассказать, что это такое

Показать пример создания и работы с ними:

Context context = getActivity();

SharedPreferences sharedPref = context.getSharedPreferences(

getString(R.string.preference\_file\_key), Context.MODE\_PRIVATE);

Запись:

SharedPreferences.Editor editor = sharedPref.edit();

editor.putInt(getString(R.string.saved\_high\_score\_key), newHighScore);

editor.commit();

Чтение:

int defaultValue = getResources().getInteger(R.integer.saved\_high\_score\_default\_key);

int highScore = sharedPref.getInt(getString(R.string.saved\_high\_score\_key), defaultValue);

## База данных в android

## Что такое SQLite?

SQLite — это встраиваемая кроссплатформенная БД, которая поддерживает достаточно полный набор команд SQL и доступна в исходных кодах (на языке C).

Исходные коды SQLite находятся в public domain, то есть вообще никаких ограничений на использование.

Сайт (с прекрасной документацией на английском): [http://sqlite.org](http://sqlite.org/)

Слово «встраиваемый» (embedded) означает, что SQLite не использует парадигму [клиент-сервер](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80), то есть [движок](https://ru.wikipedia.org/w/index.php?title=%D0%94%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA&action=edit&redlink=1) SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а представляет собой [библиотеку](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), с которой программа компонуется, и движок становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций ([API](https://ru.wikipedia.org/wiki/API)) библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором исполняется программа. Простота реализации достигается за счёт того, что перед началом исполнения транзакции записи весь файл, хранящий базу данных, блокируется; [ACID](https://ru.wikipedia.org/wiki/ACID)-функции достигаются в том числе за счёт создания файла журнала.

Пройтись по примеру <https://developer.android.com/training/data-storage/sqlite> в студии

Для отладки использую <https://github.com/amitshekhariitbhu/Android-Debug-Database>

adb forward tcp:8080 tcp:8080 - для эмулятора

<http://localhost:8080/>

## Room

There are 3 major components in Room:

* [**Database:**](https://developer.android.com/reference/android/arch/persistence/room/Database.html) Contains the database holder and serves as the main access point for the underlying connection to your app's persisted, relational data.
* The class that's annotated with [@Database](https://developer.android.com/reference/android/arch/persistence/room/Database.html) should satisfy the following conditions:
  + Be an abstract class that extends [RoomDatabase](https://developer.android.com/reference/android/arch/persistence/room/RoomDatabase.html).
  + Include the list of entities associated with the database within the annotation.
  + Contain an abstract method that has 0 arguments and returns the class that is annotated with [@Dao](https://developer.android.com/reference/android/arch/persistence/room/Dao.html).
* At runtime, you can acquire an instance of [Database](https://developer.android.com/reference/android/arch/persistence/room/Database.html) by calling [Room.databaseBuilder()](https://developer.android.com/reference/android/arch/persistence/room/Room.html#databaseBuilder(android.content.Context,%20java.lang.Class%3CT%3E,%20java.lang.String)) or[Room.inMemoryDatabaseBuilder()](https://developer.android.com/reference/android/arch/persistence/room/Room.html#inMemoryDatabaseBuilder(android.content.Context,%20java.lang.Class%3CT%3E)).
* [**Entity:**](https://developer.android.com/training/data-storage/room/defining-data.html) Represents a table within the database.
* [**DAO:**](https://developer.android.com/training/data-storage/room/accessing-data.html) Contains the methods used for accessing the database.

Как добавить в проект

def room\_version = "1.1.1"

def room\_version = "1.1.1"

implementation "android.arch.persistence.room:runtime:$room\_version"

annotationProcessor "android.arch.persistence.room:compiler:$room\_version"

// optional - RxJava support for Room

implementation "android.arch.persistence.room:rxjava2:$room\_version"

рассказать про миграции

#### Полезные ссылки

1. Основная статья из документации <https://developer.android.com/guide/topics/data/data-storage>
2. Работа с чистым Sqlite <https://developer.android.com/training/data-storage/sqlite>
3. Что такое sqlite - <https://ru.wikipedia.org/wiki/SQLite> , немного устаревшая статья, но позволит понять суть <https://habr.com/post/149356/>
4. MUST HAVE урок по room от гугл <https://codelabs.developers.google.com/codelabs/android-persistence/#0>
5. Урок по room от старт андроид <https://startandroid.ru/ru/courses/architecture-components/27-course/architecture-components/529-urok-5-room-osnovy.html>

#### Задание

Основная задача - написать записную книжку с дизайном, как у приложения Google Keep. <https://play.google.com/store/apps/details?id=com.google.android.keep&hl=ru> Но с сокращенным функционалом.

Использовать Room в связке с RxJava.  
**Экран 1 - основной.**

Содержит список созданных ранее заметок. Не забыть предусмотреть состояние, когда данных нет и когда данные получаются из БД.

Кроме этого есть кнопка поиска. При нажатии на нее иконка разворачивается в строку поиска. Делать вот так: <https://stackoverflow.com/a/34255229/1120982> При печатании должен выполняться поиск в базе и отображаться результат найденных заметок. Искать по заголовку и по тексту заметки. - **поиск - это дополнительное задание, по желанию**

При долгом нажатии на заметку открывается алерт диалог, в котором есть 2 пункта - “удалить” и “архивировать”. Если удалять - заметка удаляется из бд. Если архивировать, то она не удаляется, а становится невидимой (помещается в архив).   
Список заметок не должен отображать архивированные заметки!

**Экран необходимо реализовать через горячую подписку (Flowable )на таблицу с заметками**, чтобы список автоматически перестраивался при добавлении/удалении заметок

Круг с плюсиком - FloatingActionButton

**Экран 2 - создание заметки**

Он же является экраном редактирования заметки

Заметка может хранить только текст и заголовок. Заголовок не является обязательным. Все содержимое заметки сохраняется в бд, когда пользователь нажимает кнопку “Назад”.

Если пользователь редактирует заметку и при этому полностью удаляет все ее содержимое и заголовок, то заметка не удаляется из списка, она остается пустой.

У заметки можно менять фон. Фон задается через алерт диалог.

В дизайне есть неточность - у заметки цвет текста всегда белый должен быть.