|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА  Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра математического обеспечения и стандартизации ИТ**

**ОТЧЁТ ПО ПРАКТИЧЕСКОМУ ЗАНЯТИЮ №9**

**по дисциплине**

**«Разработка мобильных приложений»**

|  |  |  |  |
| --- | --- | --- | --- |
| Отчет представлен к  рассмотрению:  Студенты группы ИНБО-04-20 | « » февраля 2022 г. | (подпись) | Ло В.Х. |
|  |  |  |  |
| Преподаватель | « » 2022 г. | (подпись) | Фандеев И.И. |

Москва, 2022г.

**СОДЕРЖАНИЕ**

[ОТЧЕТ ВЫПОЛНЕНИЯ ПРАКТИЧЕСКОЙ РАБОТЫ. 3](#_Toc100238502)

[1. Запрос общего файла 3](#_Toc100238503)

[1.1 Отправка запрошенного файла 3](#_Toc100238504)

[1.2 Доступ к запрошенному файлу 3](#_Toc100238506)

[2 Получение информации о файле 4](#_Toc100238507)

[2.1 Получение MIME типа файла 4](#_Toc100238508)

[2.2 Получение имени и размера файла 5](#_Toc100238509)

[3. Обмен данными с помощью NFC 5](#_Toc100238510)

[3.1 Отправка файла на другое устройство 6](#_Toc100238511)

[3.2 Требуемые разрешения 6](#_Toc100238512)

[3.3 Добавляем функции NFC 6](#_Toc100238513)

[3.4 Проверяем поддержку передачи файлов Android Beam 6](#_Toc100238514)

[3.5 Создание метода обратного вызова для передачи файлов 7](#_Toc100238517)

[3.6 Реализовать выбор файлов для отправки 8](#_Toc100238518)

[4. Получение файла с другого устройства (NFC) 8](#_Toc100238519)

[4.1 Отображение данных 8](#_Toc100238520)

[4.2 Запрос разрешения на файл 9](#_Toc100238521)

[4.3 Реализовать получение директории принятых файлов 9](#_Toc100238522)

[4.5 Реализовать получение директории из URI типа content 10](#_Toc100238523)

[4.6 Реализовать определение поставщика данных 11](#_Toc100238524)

[ВЫВОД 13](#_Toc100238525)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 13](#_Toc100238526)

# ОТЧЕТ ВЫПОЛНЕНИЯ ПРАКТИЧЕСКОЙ РАБОТЫ.

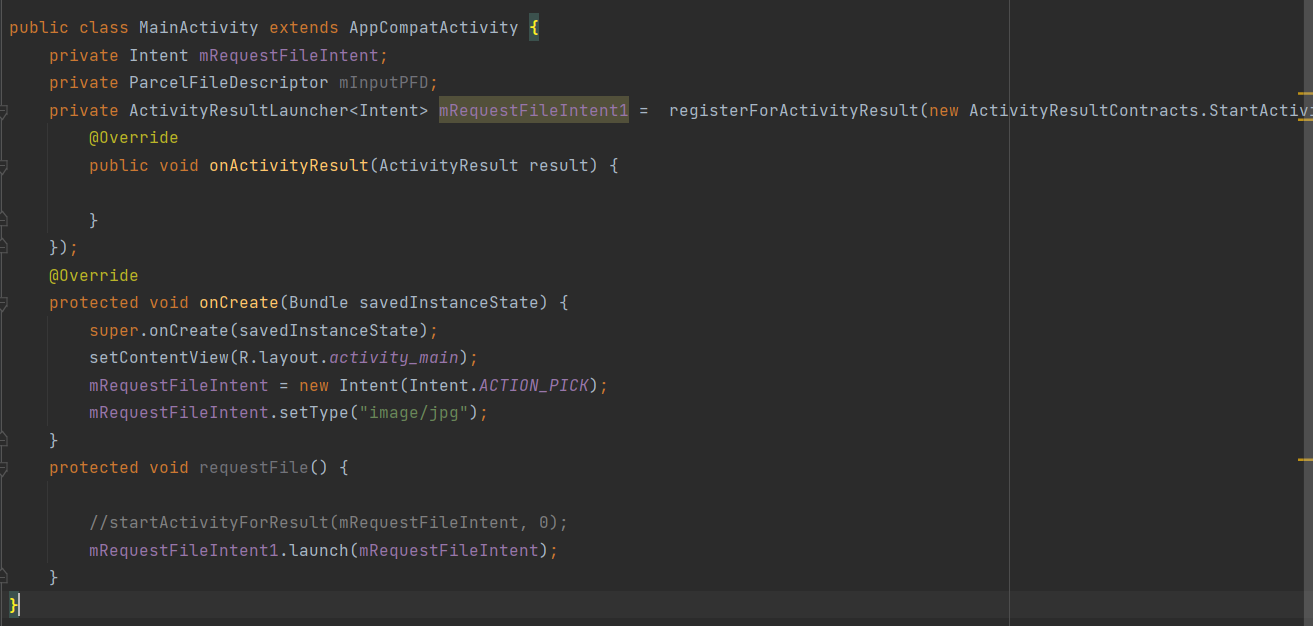
**1.** Запрос общего файла

Когда приложение (клиент) хочет получить общий файл другого приложения (сервер), оно отправляет ему запрос. В большинстве случаев, в ответ на запрос приложение открывает явление для выбора доступных файлов. Пользователь выбирает файл, после чего серверное приложение отправляет URI файла клиентскому приложению.

В данном материале мы расскажем, как запросить файл у другого приложения, получить его URI и как использовать URI, чтобы открыть файл.

**1.1** Отправка запрошенного файла

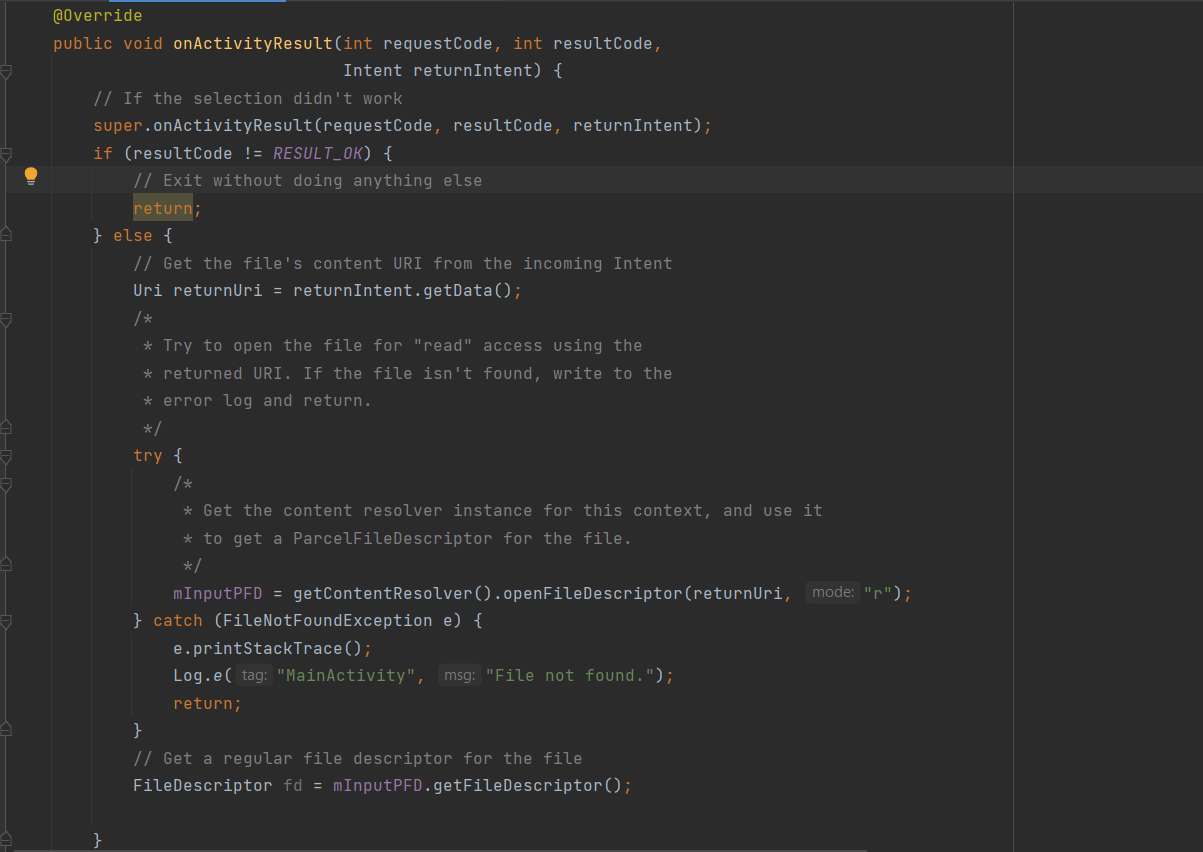
Чтобы запросить файл, клиентское приложение вызывает startActivityForResult и передает намерение, содержащее действие ACTION\_PICK и MIME тип, который приложение может принять.

****

**Рисунок 1 –** Отправка запрошенного файла

**1.2** Доступ к запрошенному файлу

Серверное приложение возвращает URI файла клиенту внутри намерения. Клиентское приложение может принять это намерение в методе onActivityResult(). После получения URI, клиентское приложение может открыть файл, получив файловый дескриптор – объект типа FileDescriptor.



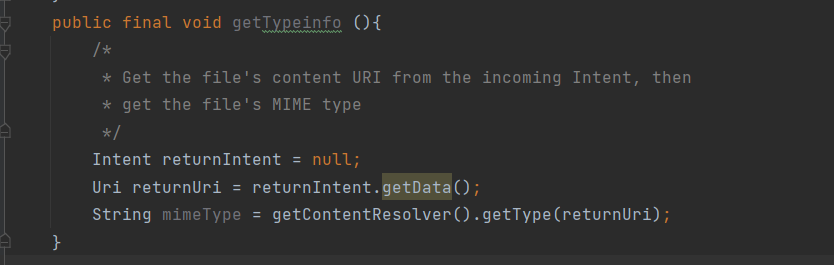
**Рисунок 2 –** Доступ к запрошенному файлу

**2** Получение информации о файле

Прежде чем начать работать с файлом, клиентское приложение может получить информацию о файле от серверного приложения, включая типа данных и размер файла. Тип данных позволяет определить, может ли клиентское приложение его обработать, а размер файла помогает выделить под него место в памяти и кэше.

2.1 Получение MIME типа файла

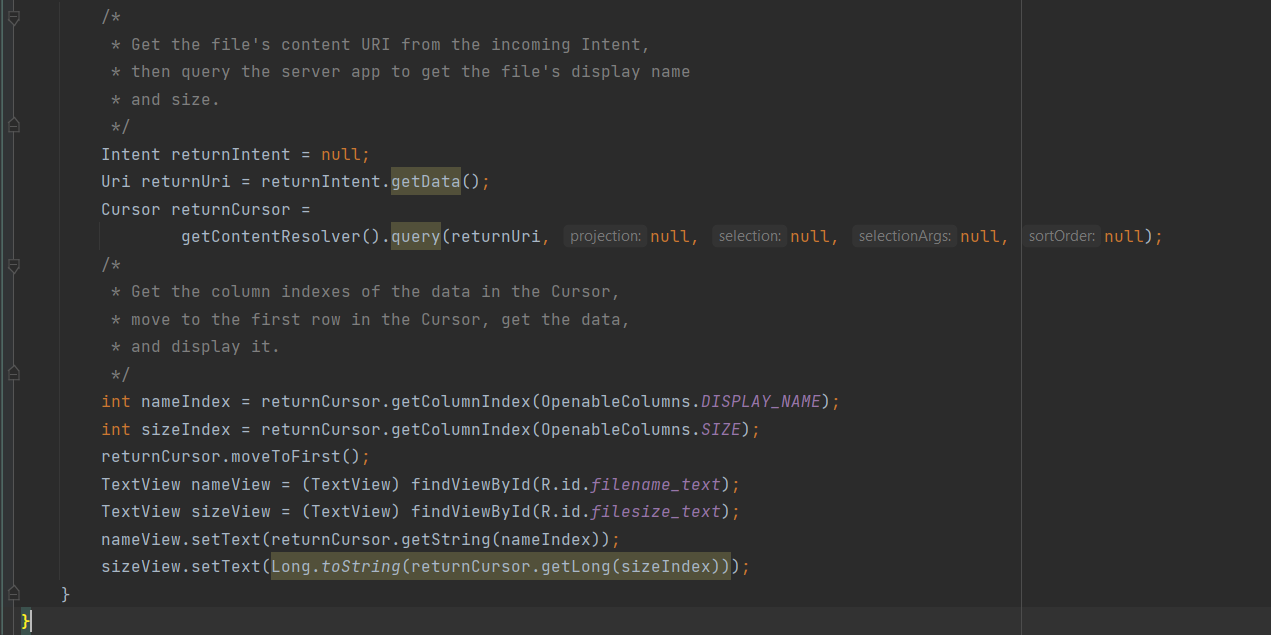
Тип файла показывает, как приложение должно обрабатывать его содержимое. Чтобы получить тип файла по его URI, клиентское приложение может вызвать метод ContentResolver.getType(). Метод возвращает MIME тип. По умолчанию, FileProvider определяет тип файла по его расширению.



**Рисунок 3 –** Получение MIME типа файла

2.2 Получение имени и размера файла

Для обработки данных, переданных в намерении, необходимо сначала получить экземпляр объекта типа Intent с помощью метода getIntent(). После этого вы можете изучить данные, чтобы решить, что с ними делать. Не забывайте, что если явление может быть запущено из разных приложений, а также из списка программ.



**Рисунок 4 –** Получение имени и размера файла

3. Обмен данными с помощью NFC

Android позволяет передавать большие файлы между устройствами, используя технологию Android Beam. Технология имеет простой интерфейс и позволяет пользователям начать передачу просто соприкоснувшись устройствами. Android Beam автоматически передает файл с одного устройства на другое и информирует пользователя после окончания процесса.

3.1 Отправка файла на другое устройство

В этом уроке мы рассмотрим как передавать большие файлы на другое устройство, используя Android Beam. Для отправки файлов нужно получить разрешение наиспользование NFC и внешнего хранилища, убедиться, что устройство поддерживает NFC и предоставить URI файлов.

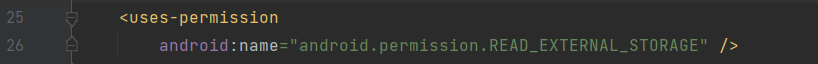
3.2 Требуемые разрешения

Позволяет вашему приложению передавать данные по NFC. Чтобы указать это разрешение, добавьте следующий элемент в качестве дочернего [<manifest>](http://developer.android.com/guide/topics/manifest/manifest-element.html) элемента:



**Рисунок 5 – Позволяет вашему приложению передавать данные по NFC**

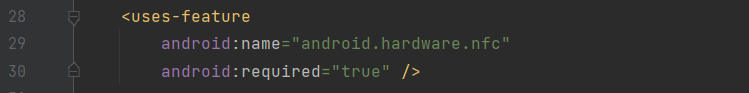
Позволяет вашему приложению чтение из внешнего хранилища. Чтобы указать это разрешение, добавьте следующий элемент в качестве дочернего [<manifest>](http://developer.android.com/guide/topics/manifest/manifest-element.html) элемента:



**Рисунок 6 – Позволяет вашему приложению чтение из внешнего хранилища.**

**3.3 Добавляем функции NFC**

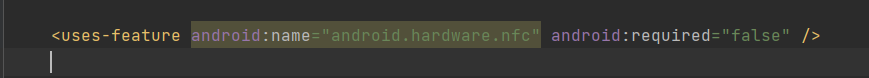
Добавьте дочерний элемент <uses-feature> в <manifest>, чтобы включить использование NFC приложением. Установите атрибут android:required равным true, чтобы указать, что приложение не будет работать, если NFC отсутствует.



**Рисунок 7 –** Добавьте дочерний элемент <uses-feature> в <manifest>

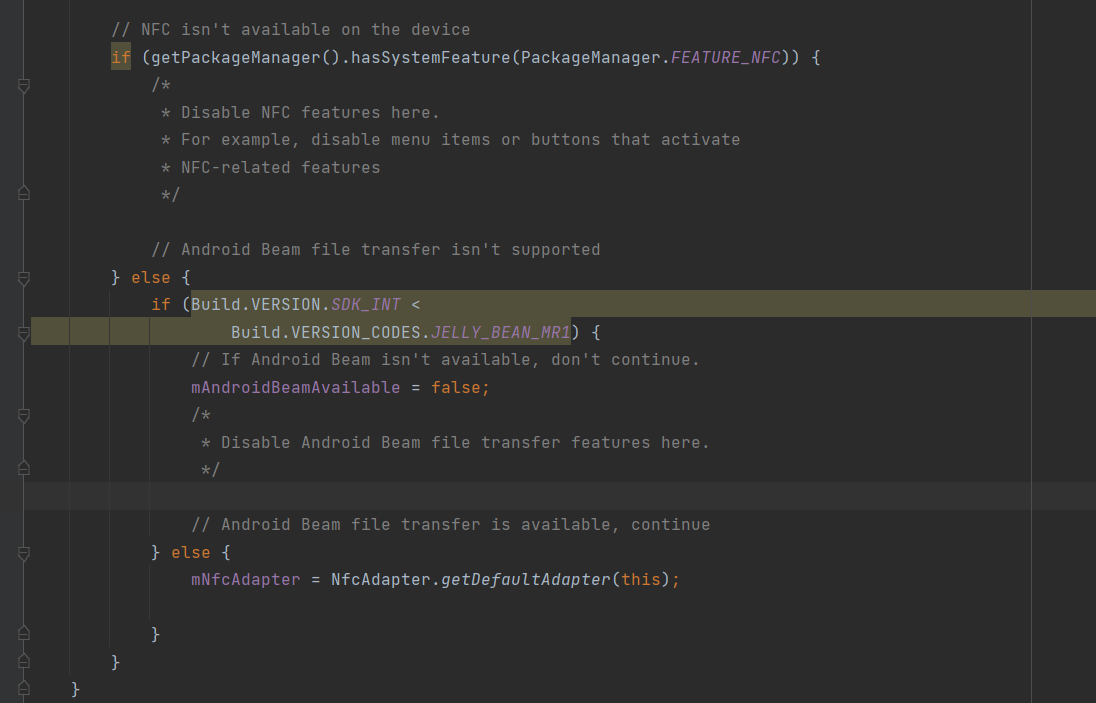
3.4 Проверяем поддержку передачи файлов Android Beam

Чтобы указать в манифесте вашего приложения, что NFC не является обязательным для работы, вы можете использовать следующий элемент:



**Рисунок 8 – Указать в манифесте**

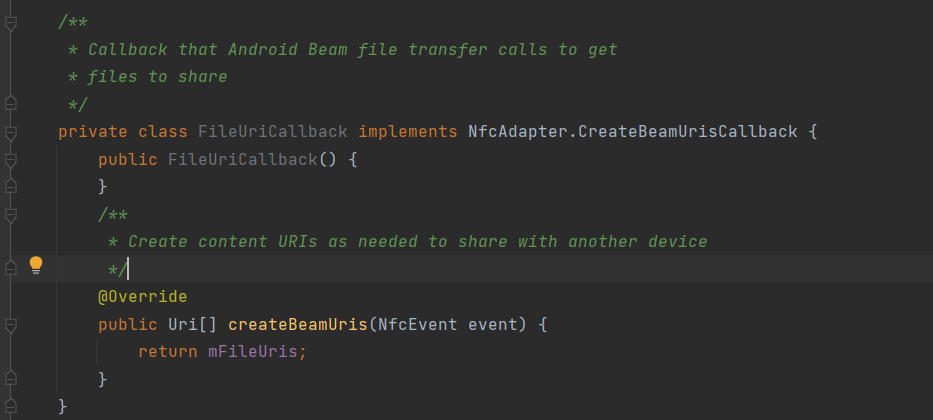
Чтобы программно проверить поддержку Android Beam для передачи файлов, вызовите метод PackageManager.hasSystemFeature() с аргументом FEATURE\_NFC. Затем проверьте значение SDK\_INT. Если передача файлов Android Beam поддерживается, получите экземпляр NFC контроллера.



**Рисунок 9 –** **Проверить поддержку Android Beam file transfer в коде**

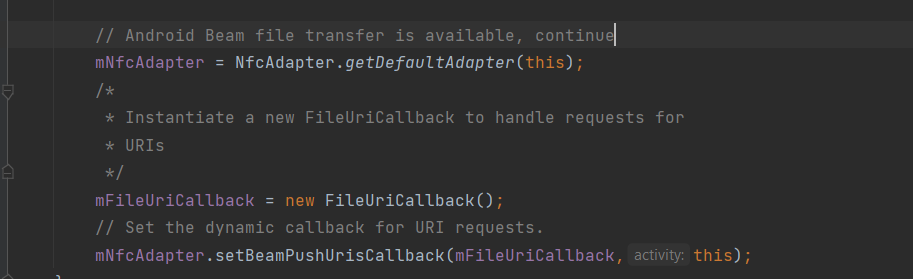
**3.5 Создание метода обратного вызова для передачи файлов**

После проверки поддержки Android Beam, создайте метод обратного вызова, который система будет вызывать при попытке передать файл через NFC. Этот метод будет возвращать массив Uri объектов. Android Beam передаст файлы по указанным Uri на другое устройство.



**Рисунок 10 – Добавить метод обратного вызова**

После того, как вы реализовали интерфейс, предоставьте метод обратного вызова для Android Beam file transfer, вызвав [setBeamPushUrisCallback()](http://developer.android.com/reference/android/nfc/NfcAdapter.html#setBeamPushUrisCallback(android.nfc.NfcAdapter.CreateBeamUrisCallback,%20android.app.Activity)).



**Рисунок 11 – Предоставьте метод обратного вызова для Android Beam file**

**3.6** Реализовать выбор файлов для отправки

Для передачи одного или нескольких файлов на другое устройство с NFC, получите файловые URI (со схемой file) для каждого файла и добавьте их в массив Uri объектов.

/\*  
 \* Create a list of URIs, get a File,  
 \* and set its permissions  
 \*/  
private Uri[] mFileUris = new Uri[10];  
String transferFile = "transferimage.jpg";  
File extDir = getExternalFilesDir(null);  
File requestFile = new File(extDir, transferFile);  
 requestFile.setReadable(true, false);  
// Get a URI for the File and add it to the list of URIs  
fileUri = Uri.fromFile(requestFile);  
 if (fileUri != null) {  
 mFileUris[0] = fileUri;  
} else {  
 Log.*e*("My Activity", "No File URI available for file.");  
}

**Рисунок 12 –** Реализовать выбор файлов для отправки

4. Получение файла с другого устройства (NFC)

4.1 Отображение данных

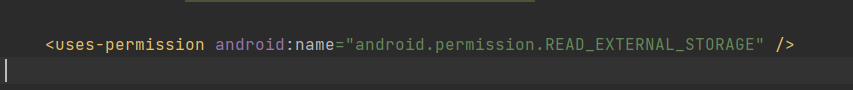
После завершения передачи, Android Beam передает уведомление, включающее намерение с действием ACTION\_VIEW, MIME тип и URI первого файла. При клике на уведомление, намерение отправляется в систему. Чтобы ваше приложение могло обработать это намерение, добавьте элемент <intent-filter> в элемент <activity> явления, которое должно отвечать на запрос. В элемент <intent-filter> добавьте следующие дочерние элементы:

<activity  
 android:name="com.example.android.nfctransfer.ViewActivity"  
 android:label="Android Beam Viewer"  
 android:exported="true"  
 tools:ignore="WrongManifestParent">  
 <intent-filter>  
 <action android:name="android.intent.action.VIEW"/>  
 <category android:name="android.intent.category.DEFAULT"/>  
 </intent-filter>  
</activity>

**Рисунок 13 –** Отображение данных

4.2 Запрос разрешения на файл

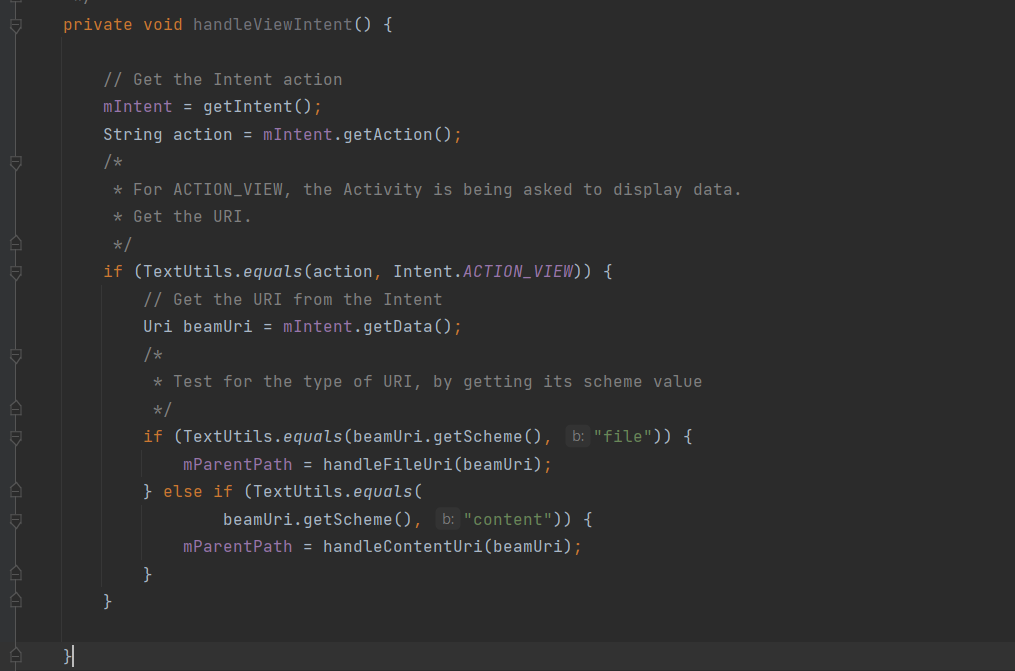
После добавления объекта FileProvider в манифест, необходимо указать директории, содержащие файлы, которыми вы готовы поделиться. Для указания директорий создайте файл filepath.xml в директории res/xml вашего проекта. В файле добавьте XML элементы для каждой директории. Далее приведен пример файла filepath.xml, который показывает как поделиться подкаталогом директории files/ из вашего внутреннего хранилища:



**Рисунок 14 –** Запрос разрешения на файл

4.3 Реализовать получение директории принятых файлов

Android Beam размещает все переданные за раз файлы в одну директорию. Намерение, которое передает Android Beam в уведомлении, содержит URI первого переданного файла. Однако не забывайте, что приложение может получить намерение с действием ACTION\_VIEW не только от Android Beam, но и от других приложений. Чтобы решить, как обрабатывать входящее намерение, необходимо проверить схему и authority.



**Рисунок 15 –** Реализовать получение директории принятых файлов

4.4 Реализовать получение директории из URI типа file

Если входящее намерение включает URI со схемой file, этот URI включает в себя абсолютный путь до файла и его имя. Как было сказано ранее, в Android Beam все файлы текущий передачи находятся в одном и том же каталоге. Чтобы получить путь до каталога, необходимо взять полный путь из URI (URI это вся строка, кроме префикса file:).

public String handleFileUri(Uri beamUri) {  
 // Get the path part of the URI  
 String fileName = beamUri.getPath();  
 // Create a File object for this filename  
 File copiedFile = new File(fileName);  
 // Get a string containing the file's parent directory  
 return copiedFile.getParent();  
}

**Рисунок 16 –** Реализовать получение директории из URI типа file

4.5 Реализовать получение директории из URI типа content

Если входящее намерение содержит URI типа content, директория и имя файла хранятся в поставщике содержимого MediaStore. Сделать вывод, что перед нами URI типа content можно проверив значение authority. URI типа content для провадера MediaStore может быть получено из Android Beam или любого другого приложения, но в обоих случаях вы можете получить директорию и имя файла для этого URI.

Примечание: в Android Beam вы получите URI типа content в том случае, если первый принятый файл имеет MIME тип “audio/\*”, “image/\*” или “video/\*”, то есть является медиа-файлом. Android Beam индексирует все такие файлы с помощью медиа-сканера. Медиа-сканер записывает результаты индексации в поставщик данных MediaStore, а затем передает URI первого файла обратно в Android Beam. Это будет URI, который вы получили в уведомлении. Получить каталог первого файла можно из MediaStore, используя URI со схемой content.

**4.6** Реализовать определение поставщика данных

Чтобы получить каталог из URI типа content, определите тип поставщика содержимого, связанного с URI, с помощью метода Uri.getAuthority(). Метод возвращает значение authority, которое означает следующее:

MediaStore.AUTHORITY

URI файла, который отслеживается MediaStore. Получите полный путь до файла из MediaStore, а директорию вычислите из это пути. Любое другое значение authority URI любого другого поставщика содержимого. Нельзя получить директорию по URI.

Чтобы получить директорию при поставщике содержимого MediaStore, выполните запрос, в котором укажите столбец MediaColumns.DATA, а в качестве аргумента передайте URI. Возвращенный курсор (Cursor) включает полный путь и имя файла для заданного URI. По этому пути расположены также остальные файлы, принятые вместе с текущим.

public File handleContentUri(Uri beamUri) {  
 // Position of the filename in the query Cursor  
 int filenameIndex;  
 // File object for the filename  
 File copiedFile;  
 // The filename stored in MediaStore  
 String fileName;  
 // Test the authority of the URI  
 if (!TextUtils.*equals*(beamUri.getAuthority(), MediaStore.*AUTHORITY*)) {  
 /\*  
 \* Handle content URIs for other content providers  
 \*/  
 // For a MediaStore content URI  
 } else {  
 // Get the column that contains the file name  
 String[] projection = { MediaStore.MediaColumns.*DATA* };  
 Cursor pathCursor =  
 getContentResolver().query(beamUri, projection,  
 null, null, null);  
 // Check for a valid cursor  
 if (pathCursor != null &&  
 pathCursor.moveToFirst()) {  
 // Get the column index in the Cursor  
 filenameIndex = pathCursor.getColumnIndex(  
 MediaStore.MediaColumns.*DATA*);  
 // Get the full file name including path  
 fileName = pathCursor.getString(filenameIndex);  
 // Create a File object for the filename  
 copiedFile = new File(fileName);  
 // Return the parent directory of the file  
 return new File(copiedFile.getParent());  
 } else {  
 // The query didn't work; return null  
 return null;  
 }  
 }

**Рисунок 17 –** Реализовать определение поставщика данных

# ВЫВОД

В ходе выполнения практической работы были изучены:

* Методические указания Обмен файлами (2). Обмен данными с помощью NFC

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Лекции по дисциплине «Разработка мобильных приложений» / И. В. Синицын, МИРЭА — Российский технологический университет, 2022.