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

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

**Отчет по практической работе №12**

по дисциплине «Разработка мобильных приложений»

|  |  |
| --- | --- |
| **Выполнил:**  Студент группыИКБО-20-23 | Комисарик М.А. |
| **Проверил:**  Старший преподаватель кафедры МОСИТ | Шешуков Л.С. |

Москва 2025 г.

СОДЕРЖАНИЕ

[**РТУ МИРЭА** 1](#_Toc197642652)

[СОДЕРЖАНИЕ 2](#_Toc197642653)

[1 ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ 3](#_Toc197642654)

[1.1 Работа с JSON 3](#_Toc197642655)

[1.2 ContentProvider 6](#_Toc197642656)

[2 ПРАКТИЧЕСКОЕ ЗАДАНИЕ 12](#_Toc197642657)

[2.1 ContentProvider 12](#_Toc197642658)

[2.1.1 Разрешения 12](#_Toc197642659)

[2.1.2 Реализация 13](#_Toc197642660)

[2.2 Чтение и запись JSON файлов 14](#_Toc197642661)

[2.2.1 Зависимости 14](#_Toc197642662)

[2.2.2 Разметка 15](#_Toc197642663)

[2.2.3 Реализация 16](#_Toc197642664)

[2.2.4 Тестирование 19](#_Toc197642665)

[ЗАКЛЮЧЕНИЕ 22](#_Toc197642666)

# ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ

## Работа с JSON

JSON (JavaScript Object Notation) — это легкий формат обмена данными, основанный на подмножестве языка JavaScript. JSON используется для передачи структурированных данных между клиентом и сервером. Он часто применяется в вебразработке, мобильных приложениях и других системах, где требуется передача данных.

JSON состоит из пар ключ-значение, где ключи обычно строковые, а значения могут быть строками, числами, логическими значениями, массивами, объектами или null. Его легко читать и понимать как человеку, так и компьютеру, что делает его идеальным для обмена данными.

В Android разработке JSON часто используется для передачи данных между клиентом и сервером или для сохранения информации локально.

Для работы с форматом json нет встроенных средств, но есть куча библиотек и пакетов, которые можно использовать для данной цели. Одним из наиболее популярных из них является пакет com.google.code.gson.

Для облегчения работы с JSON объектами с использованием данного пакета необходимо дополнительно добавить библиотеку Gson в build.gradle.kts (Рисунок 1).

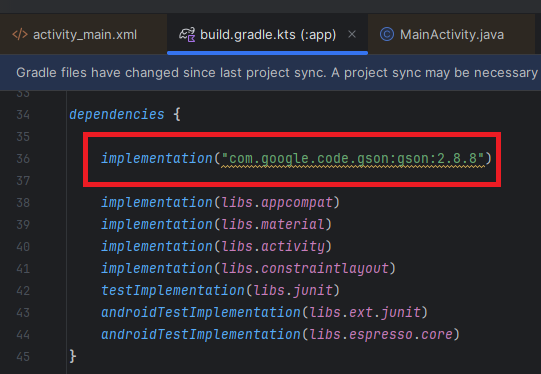


Рисунок 1 – Добавление зависимостей в сборку приложения

Теперь рассмотрим несколько примеров работы с JSON и классом User, состоящим из полей: имя, возраст, почта (Рисунок 2).



Рисунок 2 – Описание класса User

Далее необходимо создать JSON строки и десериализовать её данные (Рисунок 3).

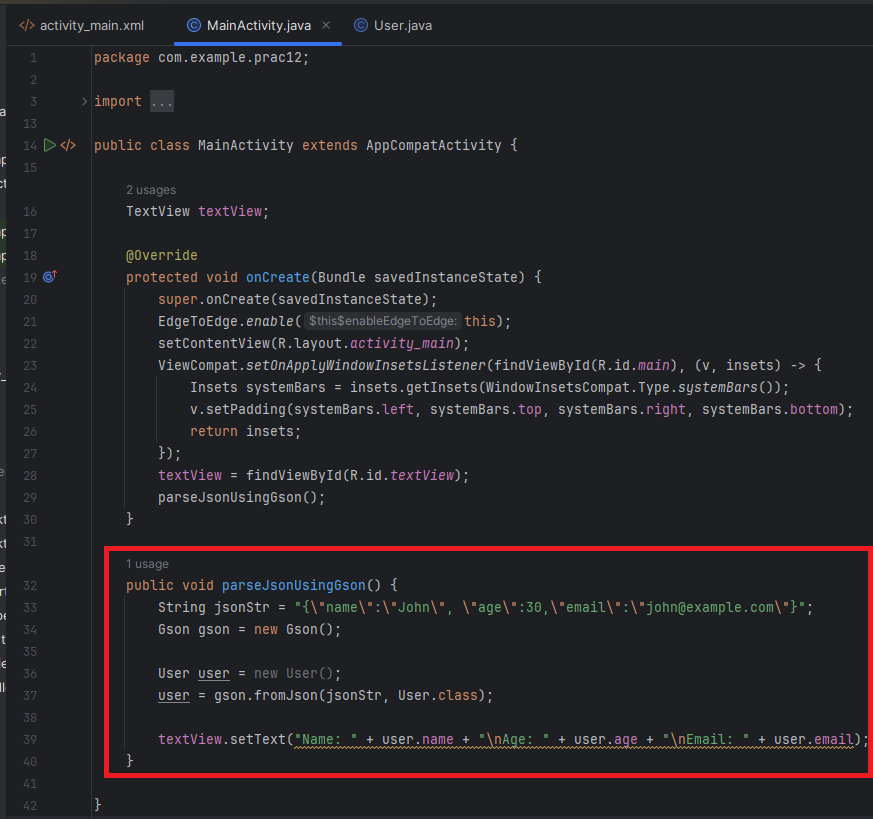


Рисунок 3 – Создание JSON строки и ее десериализация

При запуске приложения с activity, изображённой на рисунке Рисунок 3, на экране пользователя отобразятся полученные при десериализации данные (Рисунок 4).



Рисунок 4 – Отображение десериализованной JSON строки в приложении

Для работы с json создается объект Gson. Для десериализации выполняется метод fromJson(), в который передается объект класса User с сериализованными данными и тип, к которому надо десериализиовать данные.

Для создания json строки необходимо воспользоваться методом toJson(), в который подаются сериализуемые данные и который в данном случае вернет json строку (Рисунок 5).

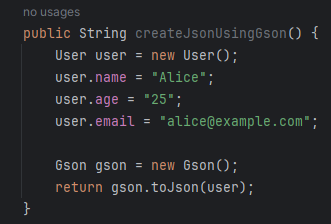


Рисунок 5 – Метод создания одной JSON строки

Для чтения же JSON массива потребуется метод fromJson() (Рисунок 6).



Рисунок 6 – Метод чтение JSON массива

При запуске приложения с вызовом метода, представленного на рисунке Рисунок 6, пользователю отобразятся десериализованные данные (Рисунок 7).

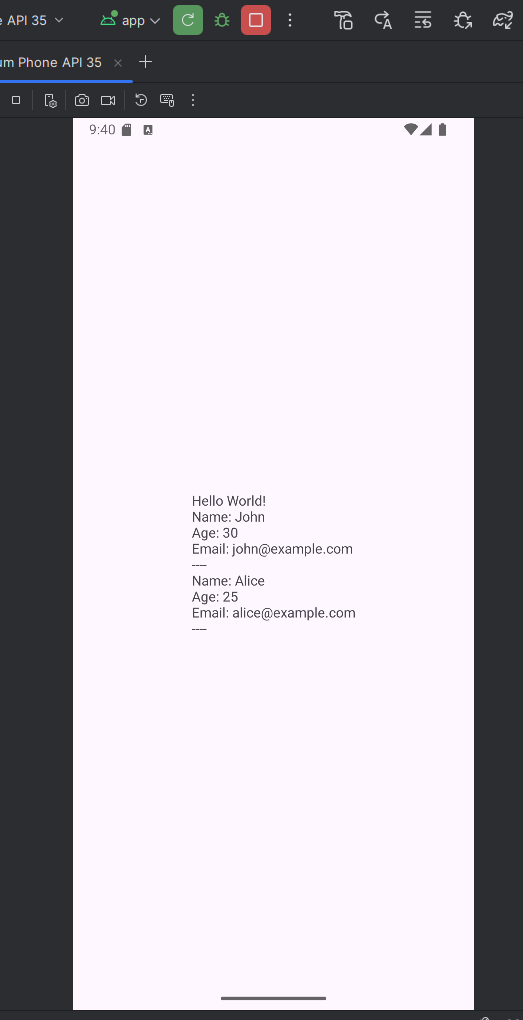


Рисунок 7 – Отображение десериализованных данных

## ContentProvider

Провайдеры контента (Content Providers) в Android представляют собой компоненты, которые позволяют приложениям делиться данными между собой. Провайдеры контента инкапсулируют данные и предоставляют механизмы для определения политик доступа к этим данным. Это особенно полезно в среде, где множество различных приложений пытаются работать с одним и тем же набором данных или когда данные должны быть предоставлены из одного приложения другим.

Провайдеры контента в Android предоставляют следующие преимущества:

* безопасный доступ к данным: провайдеры контента могут ограничивать доступ к данным приложения, предоставляя только необходимые данные и скрывая внутреннюю структуру базы данных;
* стандартизация доступа к данным: они предлагают унифицированный интерфейс для работы с данными, что упрощает обмен данными между приложениями;
* управление данными: провайдеры контента упрощают управление данными, так как обработка запросов, вставка, удаление и обновление данных производятся через стандартизированные API;
* интеграция с Android системой: провайдеры могут легко интегрироваться с другими Android компонентами, такими как загрузчики (Loaders) и система поиска.

Рассмотрим следующий пример с провайдером контента.

Имеется приложение по работе с коллекцией книг, хранящейся в базе данных SQLite и вы хотите предоставить другим приложениям доступ к информации о книгах пользователя.

Для начала необходимо определить URI (Uniform Resource Identifier) — уникальный идентификатор, используемый для идентификации данных в провайдере контента (Рисунок 8). В Android URI для провайдера контента обычно включает в себя authority, который является символьным идентификатором провайдера.

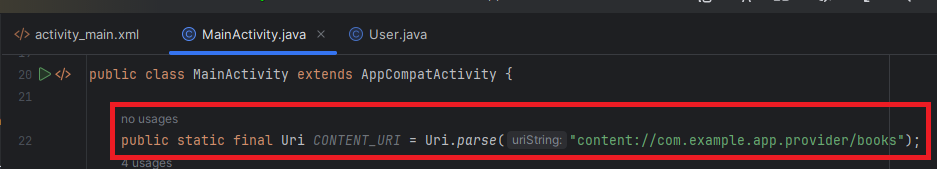


Рисунок 8 – Объявление и инициализация поля URI

Далее необходимо реализовать класс провайдера и добавить в него базу данных.

Для создания собственного контент-провайдера нужно унаследоваться от абстрактного класса ContentProvider (Рисунок 9).

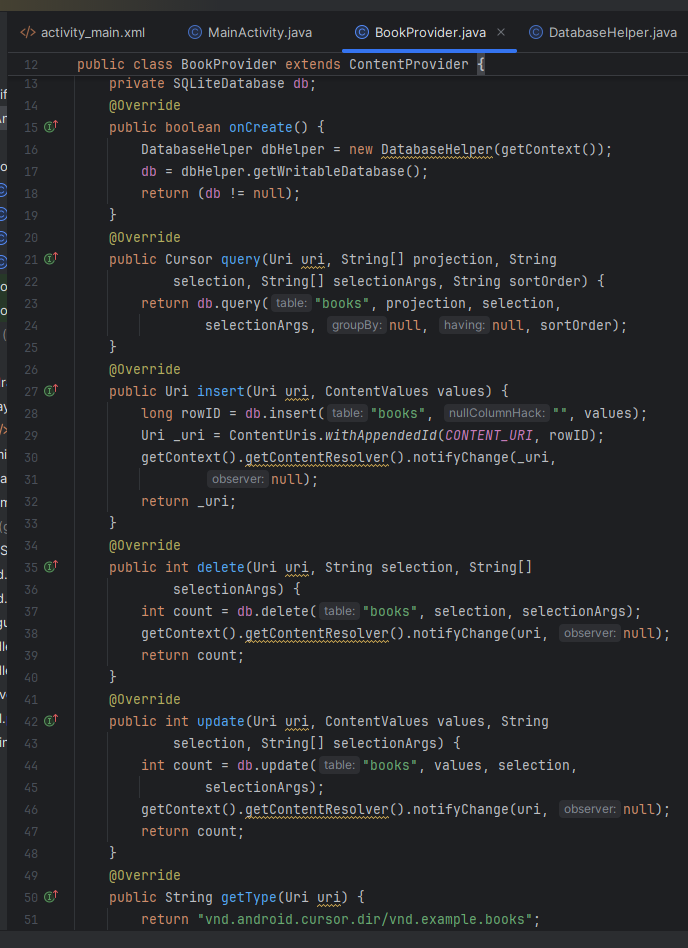


Рисунок 9 – Описание класса BookProvider

Последним шагом станет добавление провайдера в манифесте (Рисунок 10) и заполнение базы данных информацией о книгах.

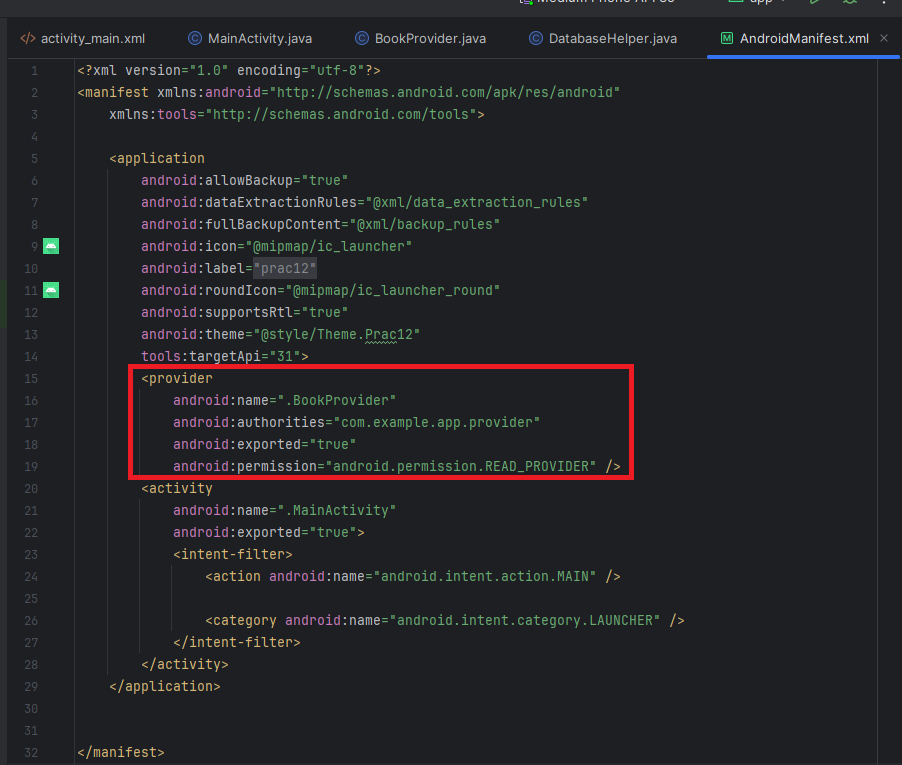


Рисунок 10 – Описание тега "provider" в файле AndroidManifest.xml

Также в манифесте другого приложения даем разрешение на чтение данных из провайдера (Рисунок 11).

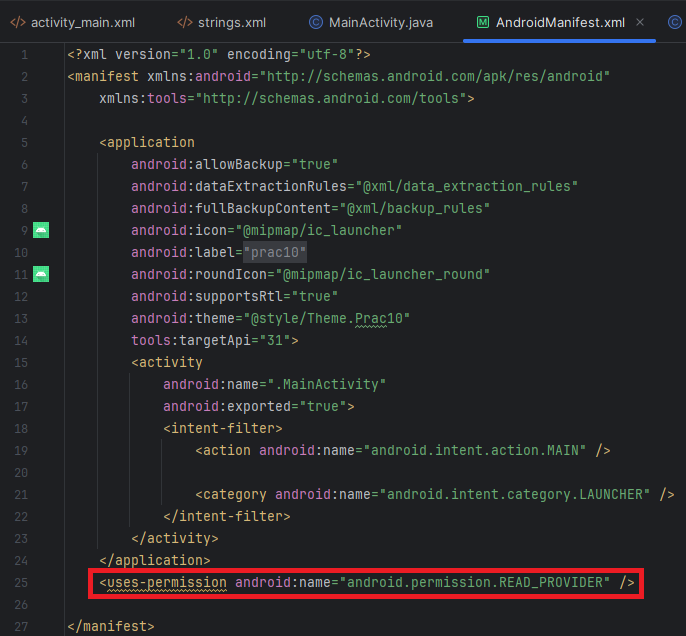


Рисунок 11 – Разрешение READ\_PROVIDER в файле AndroidManifest.xml

Далее необходимо описать логику работу главной activity, через которую и будет производиться запись данных в провайдер контента при помощи метода insert() (Рисунок 12).

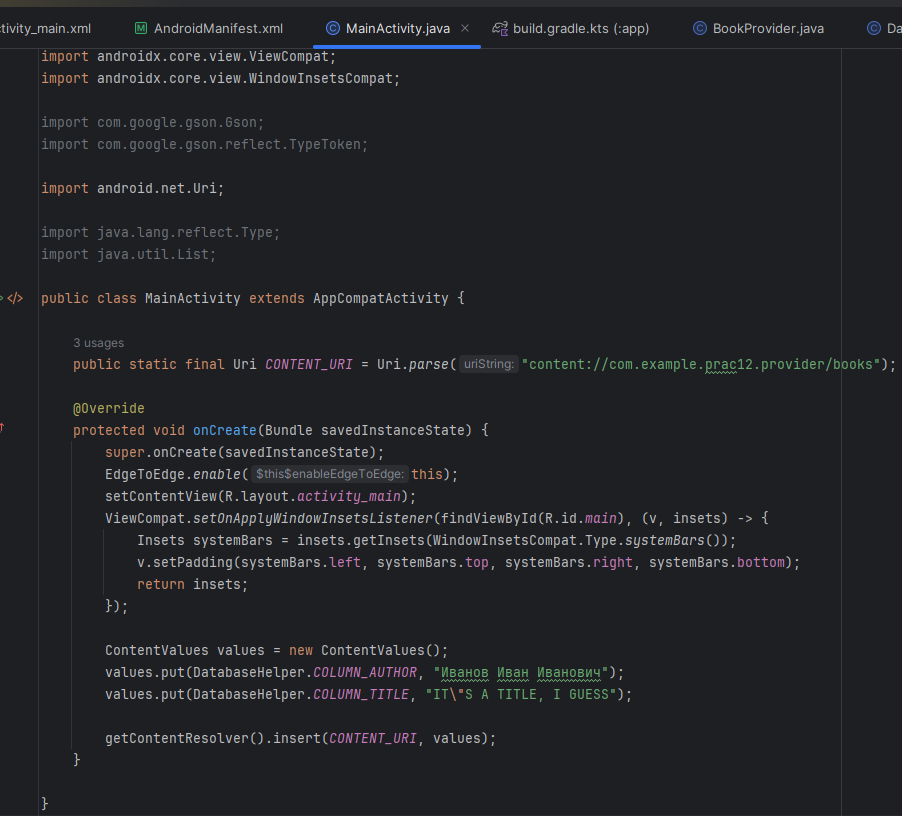


Рисунок 12 – Запись данных в ContentProvider

При запуске приложения произойдёт запись базы данных “books” в провайдер контента.

Далее необходимо описать логику считывания данных из провайдера контента (Рисунок 13).

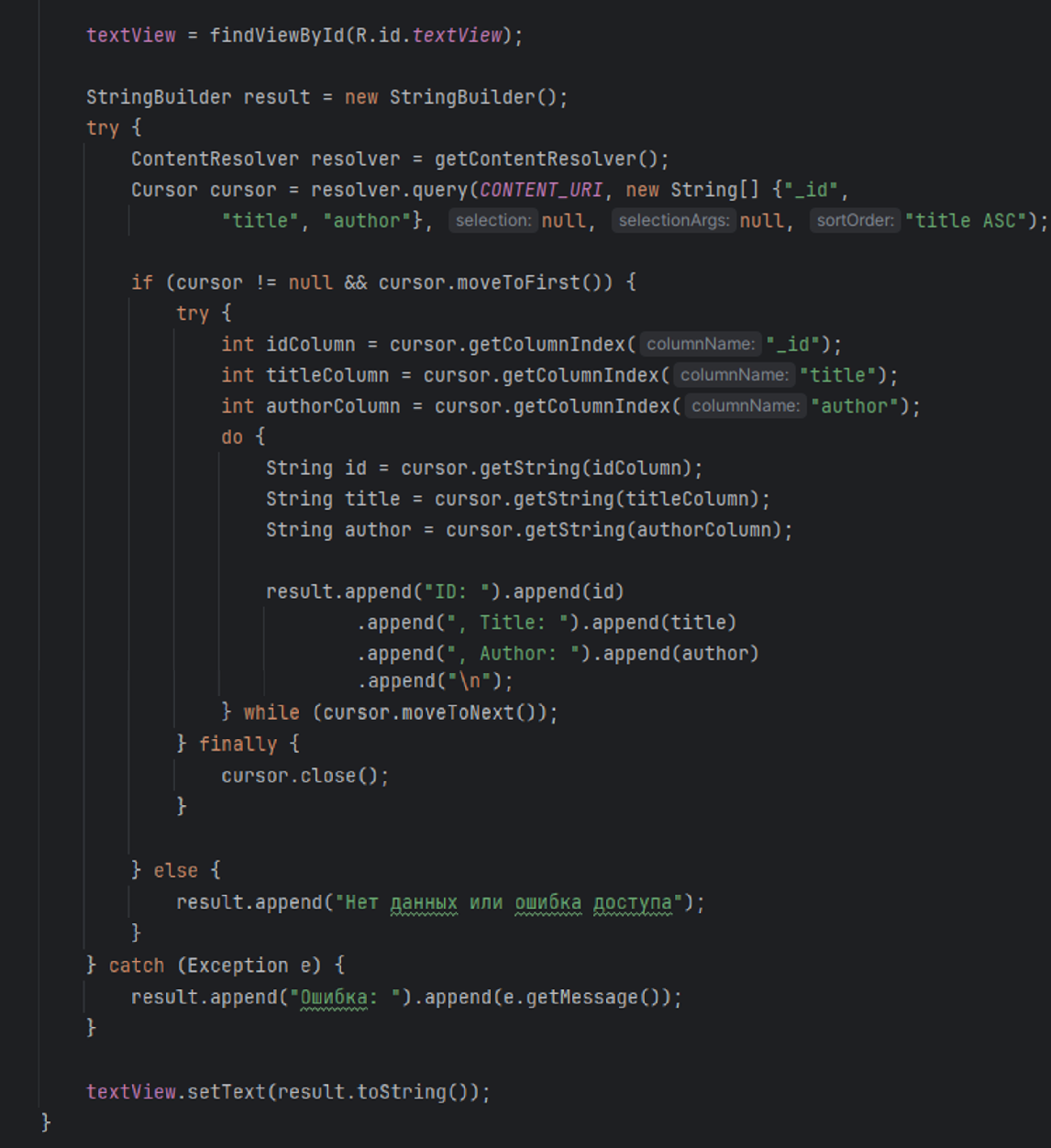


Рисунок 13 – Считывание данных из ContentProvider

На рисунке Рисунок 13 описан процесс считывания данных из ContentProvider, а также отображение считанных данных на экран (Рисунок 14).

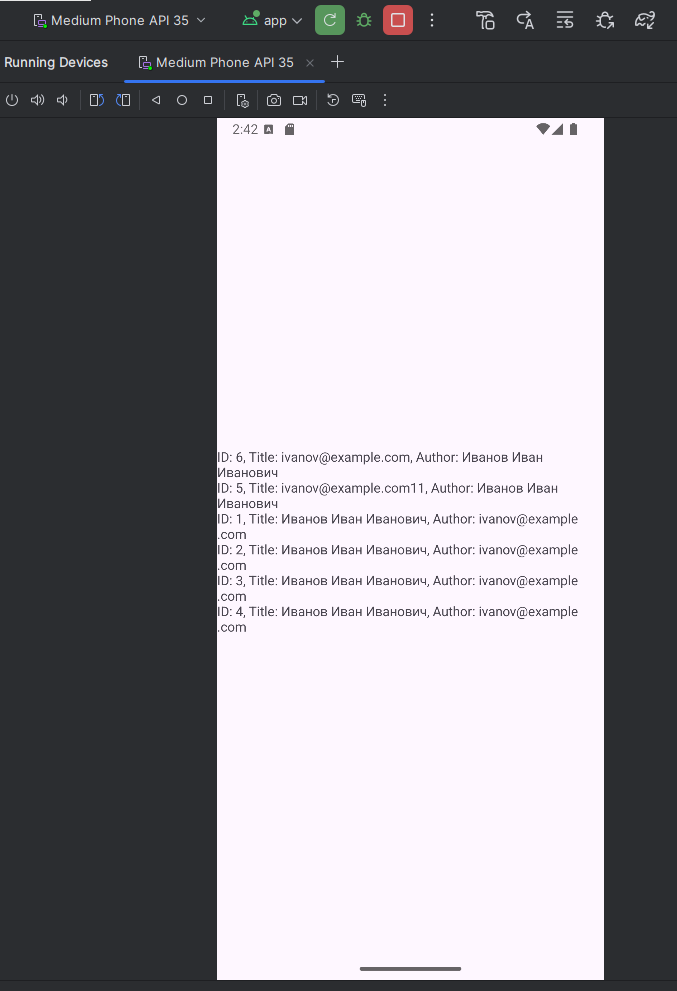


Рисунок 14 – Отображение данных, считанных из ContentProvider

# ПРАКТИЧЕСКОЕ ЗАДАНИЕ

## ContentProvider

### Разрешения

Для передачи файла между двумя приложениями будем использовать класс библиотеки android FileProvider, наследуемый от ContentProvider.

Чтобы передать файл с использованием FileProvider требуется предоставить разрешение на доступ к URI из приложения, передающего файл, вызвав метод grantUriPermission() класса FileProvider. Однако перед этим нужно перечислить все разрешенные пути в файле манифеста.

Добавим тег <provider> в файл манифеста внутри тега <application>, в котором укажем все необходимые параметры (Рисунок 15).



Рисунок 15 – Описание тега <provider> в файле AndroidManifest.xml

Атрибуты этого тега:

* authorities="com.example.fileprovider" – уникальный идентификатор полномочия, необходимый для составления URI к передаваемому файлу;
* name="androidx.core.content.FileProvider" – указание на использование класса FileProvider в качестве провайдера;
* exported="false" – отключение доступа программ к данному провайдеру, если у них нет прямого разрешения;
* grantUriPermissions="true" – предоставляет возможность создать разрешение на использование данного провайдера.

Кроме того, в данном теге также присутствует внутренний тег <meta-data>, необходимый для предоставления всех разрешенных для чтения/записи папок и файлов. Атрибут **name** зарезервирован и обязан быть "android.support.FILE\_PROVIDER\_PATHS", атрибут **resource** указывает на расположение файла с разрешенными путями.

На рисунке Рисунок 16 изображено содержание файла shared\_files.xml.

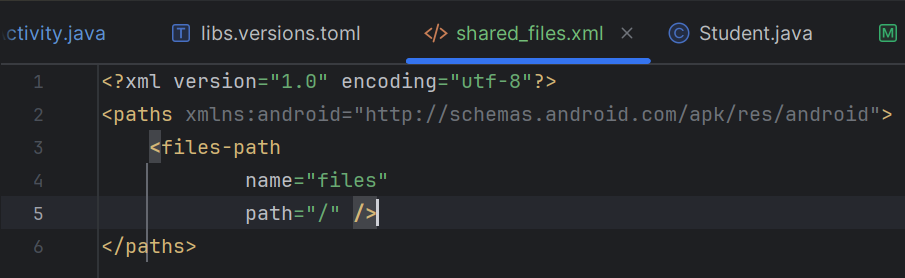


Рисунок 16 – Содержание файла shared\_files.xml

Внутри файла shared\_files.xml коренным тегом является <paths>, а внутри располагаются все разрешенные пути. В данном случае разрешенным путем было выбрано все внутреннее хранилище приложения.

### Реализация

Для предоставление доступа к файлу нужно вызвать метод grantUriPermission() класса FileProvider, в котором указывается пакет, приложению которого требуется предоставить разрешение, URI файла и флаги. Для получения URI файла нужно вызвать метод getUriForFile(), в который передать ранее указанную строку в атрибуте authority и класс File, указывающий на необходимый для передачи файл (Рисунок 17).

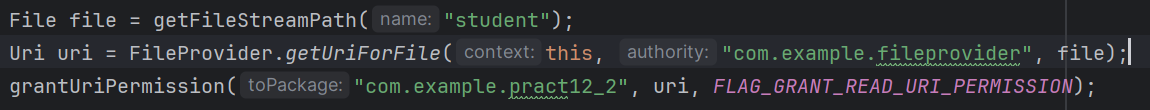


Рисунок 17 – Предоставление разрешения на чтение файла с помощью FileProvider

Для чтения файла из другого приложения требуется воспользоваться методом openInputStream() класса ContentResolver (Рисунок 18).

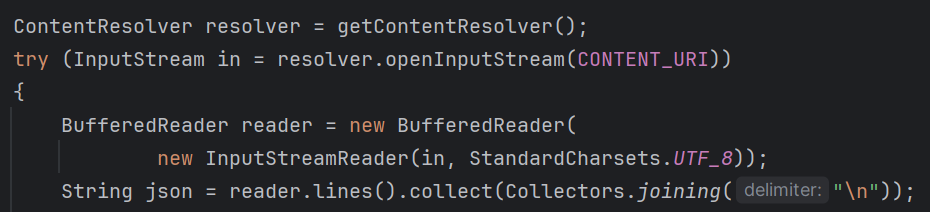


Рисунок 18 – Получение содержимого файла из другого приложения

## Чтение и запись JSON файлов

### Зависимости

Для конвертирования между JSON и строковым представлениями воспользуемся классом Gson библиотеки com.google.code.gson. Для этого добавим необходимую зависимость в файл build.gradle.kts приложения, читающего файл и приложения передающего файл (Рисунок 19).

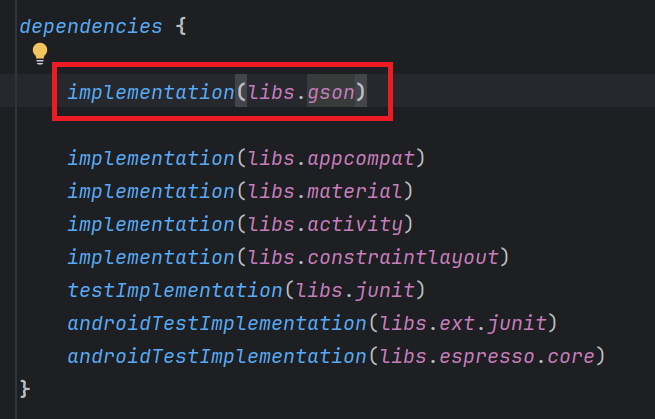


Рисунок 19 – Модуль dependencies файла build.gradle.kts

В данном случае libs.gson указывает на библиотеку com.google.code.gson версии 2.8.8, что указано в файле libs.versions.toml (Рисунок 20).

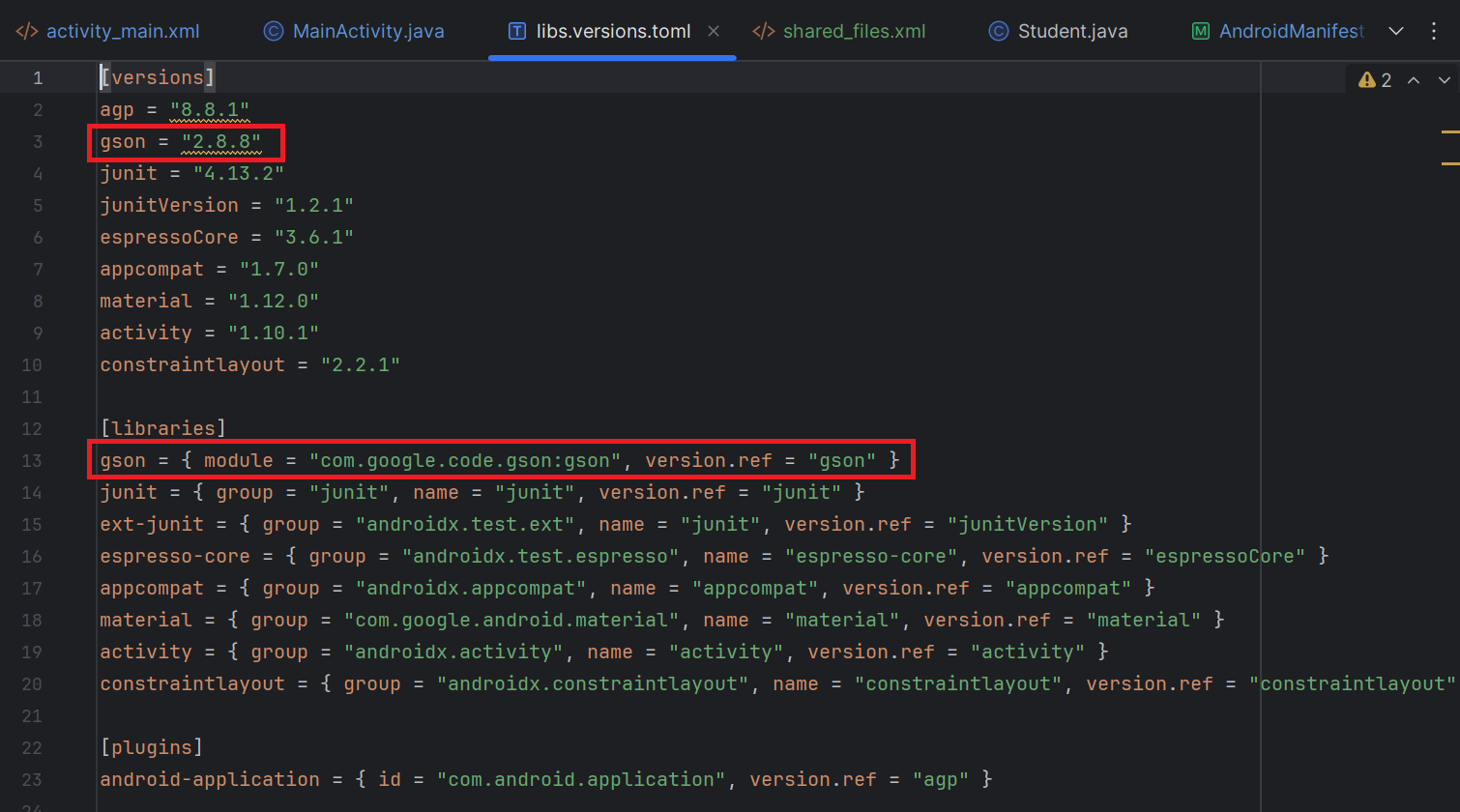


Рисунок 20 – Содержание файла libs.versions.toml

### Разметка

Добавим в разметку приложения, записывающего файл, несколько полей для ввода текста, а также кнопку для записи данных в файл (Рисунок 21).

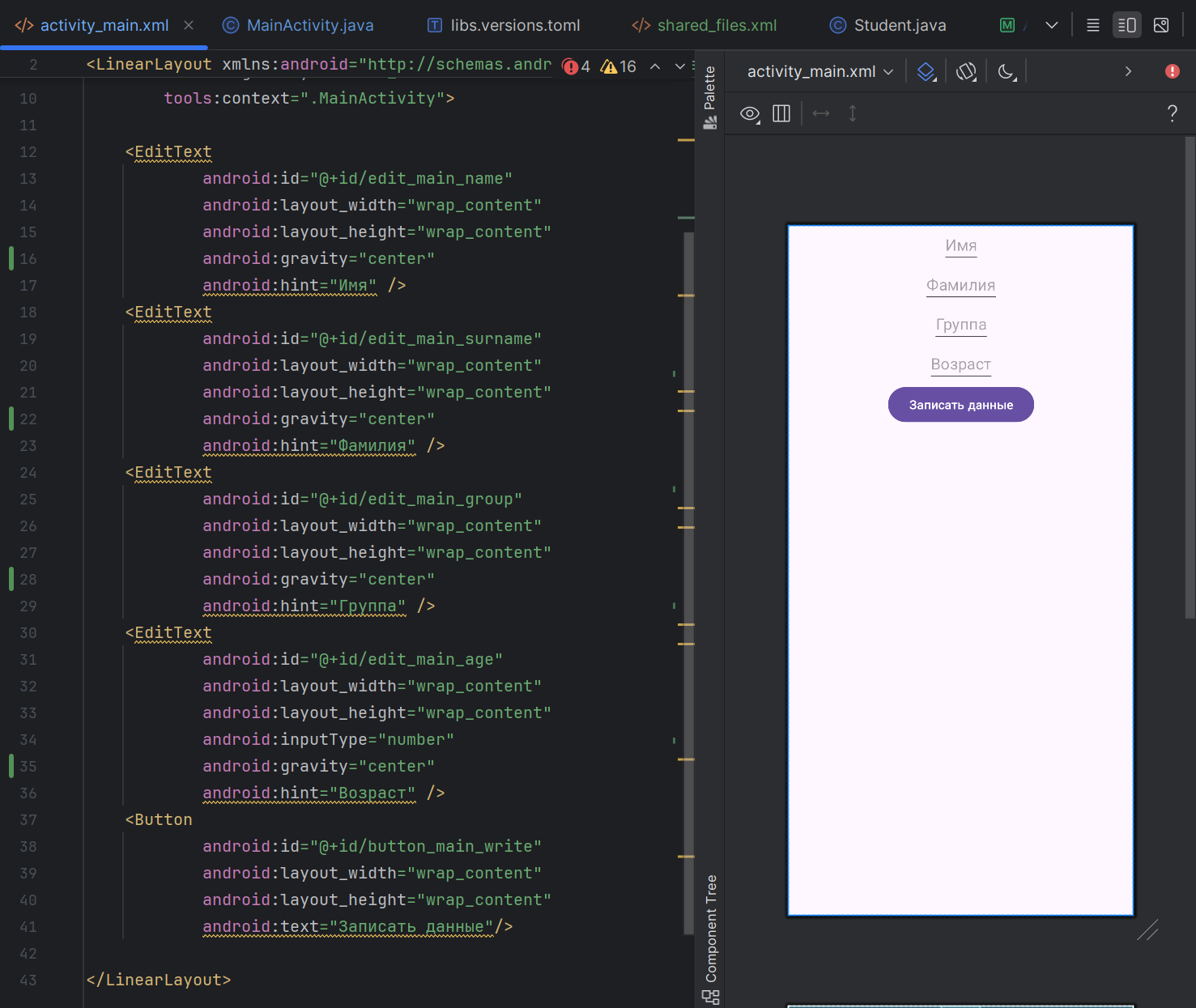


Рисунок 21 – Содержание файла acitvity\_main.xml приложения, передающего файл

Добавим в разметку приложения, считывающего файл, несколько текстовых полей для отображения полученных данных, а также кнопку для чтения файла (Рисунок 22).

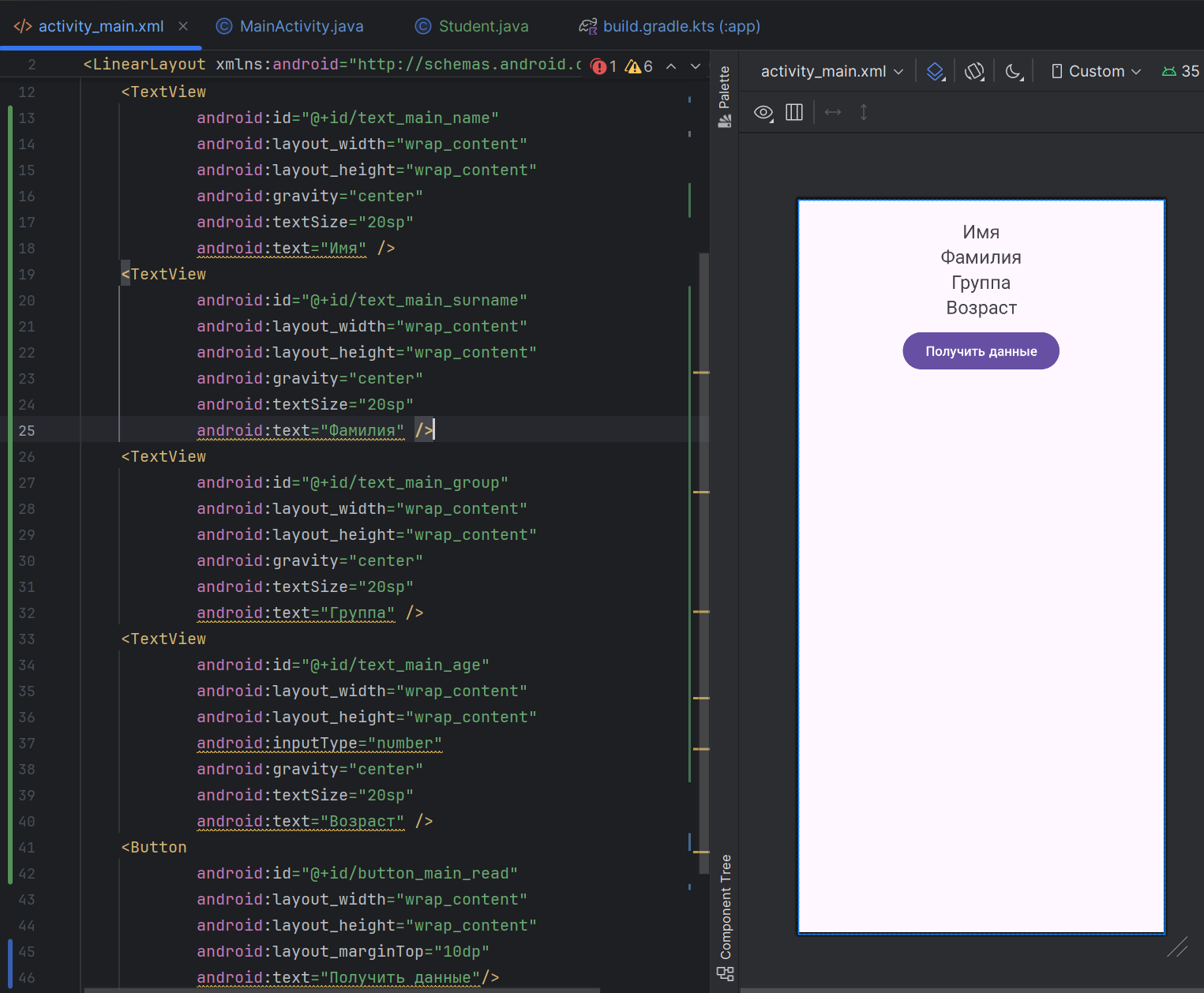


Рисунок 22 – Содержания файла activity\_main.xml приложения, получающего файл

### Реализация

Сначала в обоих приложениях создадим класс Student, в котором будут содержаться все данные для конвертации в JSON строку. В данном классе содержатся поля name, surname, group и age, а также конструктор и getter-методы для получения значения полей (Рисунок 23).



Рисунок 23 – Описание класса Student

В методе onCreate() класса MainActivity приложения-отправителя получим все данные из элементов разметки, конвертируем их в JSON строку с помощью класса Gson и запишем их в файл "student" при нажатии кнопки «Записать данные». После этого предоставим разрешение на чтение файла второму приложению при помощи ранее упомянутого класса FileProvider (Рисунок 24).

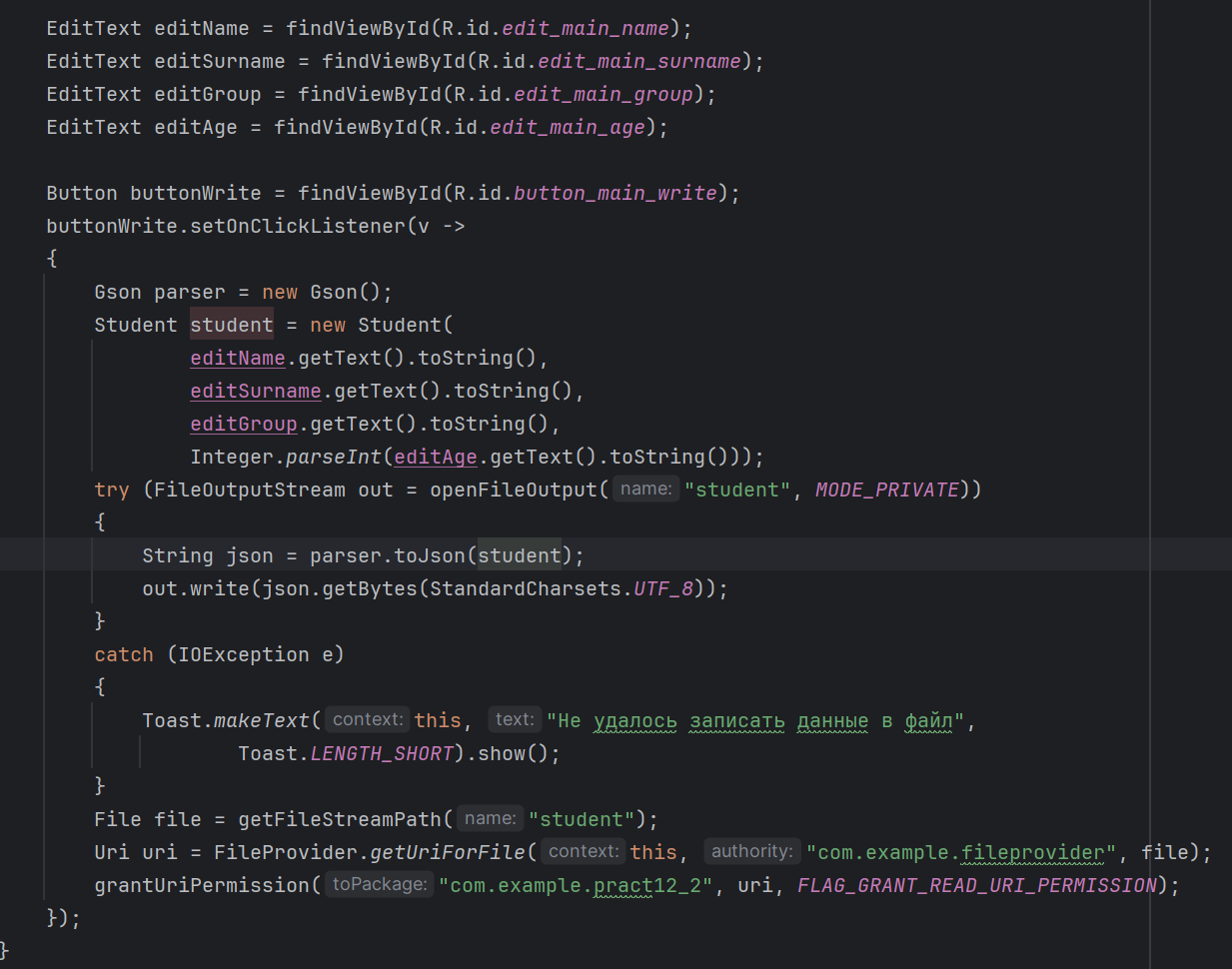


Рисунок 24 – Описание метода onCreate() класса MainActivity передающего приложения

В методе onCreate() класса MainActivity приложения-получателя получим доступ к файлу "student" при помощи класса ContentResolver, упомянутого ранее и прочитаем из него JSON строку с необходимыми данными. После этого, используя класс Gson, получим исходный объект класса Student и отобразим все полученные данные в текстовых полях при нажатии кнопки «Получить данные» (Рисунок 25).

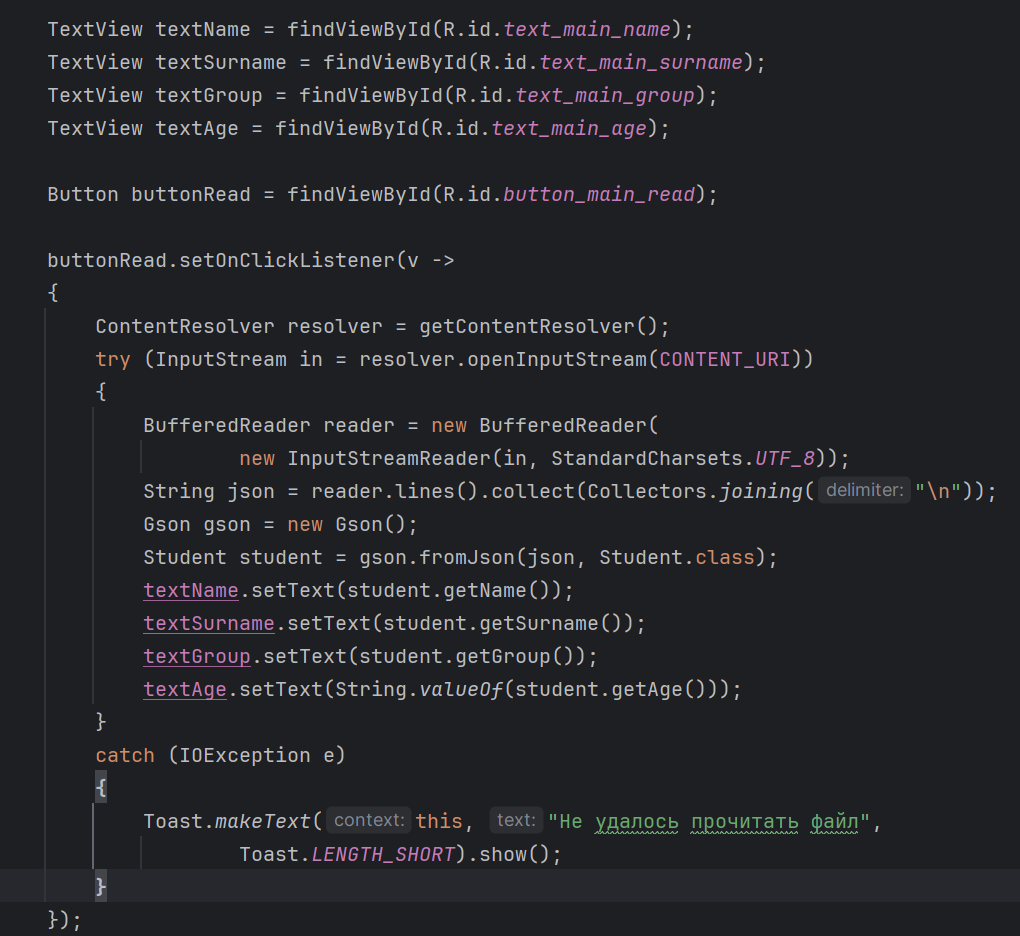


Рисунок 25 – Описание метода onCreate() класса MainActivity получающего приложения

### Тестирование

Откроем приложение для передачи данных (Рисунок 26).

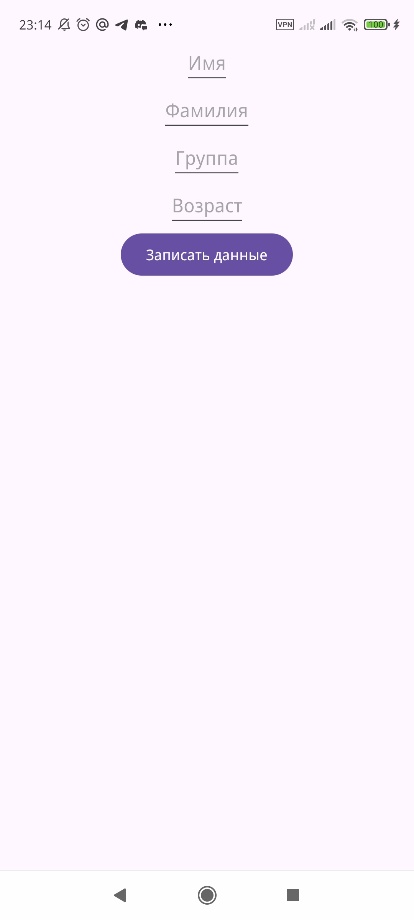


Рисунок 26 – Начальный экран приложения для передачи данных

Запишем данные в текстовые поля ввода и нажмем кнопку «Записать данные» (Рисунок 27).

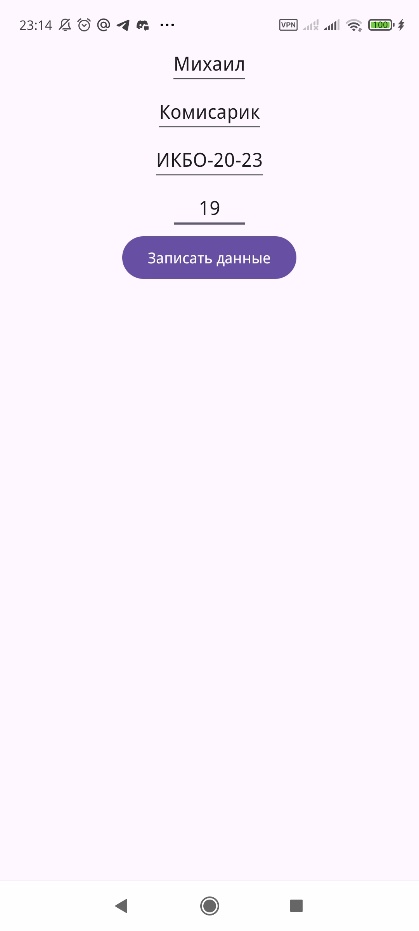


Рисунок 27 – Приложение для передачи данных после заполнения полей

Войдем во второе приложение (Рисунок 28).



Рисунок 28 – Начальный экран приложения для получения данных

Нажмем на кнопку «Получить данные», после чего текстовые поля над кнопкой заполняются переданными данными (Рисунок 29).



Рисунок 29 – Получение данных во втором приложении

Тестирование прошло успешно.

ЗАКЛЮЧЕНИЕ

В ходе работы были получены знания по работе с классом ContentProvider, а также Gson в Android Studio. Полученные знания были закреплены путём конвертации данных в JSON строку и записи в файл, а затем последующим чтением этого файла из другого приложения и конвертации полученной JSON строки обратно в переданные данные.