|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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 «Проектирование и разработка мобильных приложений»

|  |  |
| --- | --- |
| **Выполнил:**  Студент группыИКБО-21-23 | Муравьев А.О. |
| **Проверил:**  Старший преподаватель кафедры МОСИТ | Шешуков Л.С. |

Москва 2025 г.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc198900692)

[1 ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ 4](#_Toc198900693)

[1.1 Работа с JSON 4](#_Toc198900694)

[1.2 Провайдеры контента 6](#_Toc198900695)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 10](#_Toc198900696)

[2.1 Реализация передачи данных в другое приложение через провайдер контента 10](#_Toc198900697)

[2.2 Преобразование данных в JSON и сохранение в отдельный файл, преобразование из JSON в различные поля 13](#_Toc198900698)

[ЗАКЛЮЧЕНИЕ 17](#_Toc198900699)

# ВВЕДЕНИЕ

В данной работе мы сосредоточились на реализации двух важных аспектов взаимодействия и обработки данных в Android-приложениях. Во-первых, мы настроили передачу данных между приложениями с помощью провайдера контента (Content Provider) — стандартного механизма Android, позволяющего безопасно и структурировано предоставлять доступ к данным другим приложениям. Во-вторых, мы выполнили преобразование информации в формат JSON, а затем реализовали сохранение этой информации во внешний файл. Далее мы научились читать данные из JSON-файла и заполнять соответствующие поля в приложении. Эти задачи позволили нам укрепить навыки межприложного взаимодействия и обработки структурированных данных.

# 1 ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ

## Работа с JSON

JSON (JavaScript Object Notation) — это компактный и удобный формат представления данных, основанный на подмножестве синтаксиса JavaScript. Мы используем его для обмена структурированной информацией между клиентом и сервером. JSON широко применяется в веб-разработке, мобильных приложениях и других системах, где важно быстро и понятно передавать данные.

Формат JSON представляет данные в виде пар "ключ-значение", где ключи, как правило, являются строками, а значения могут быть строками, числами, булевыми значениями, массивами, объектами или null. Благодаря своей читаемости и простоте, JSON хорошо воспринимается как человеком, так и машиной, что делает его удобным стандартом для передачи данных.

В процессе Android-разработки мы часто используем JSON для обмена данными между приложением и сервером, а также для локального хранения информации.

Хотя в Android нет встроенных инструментов для работы с JSON, мы можем воспользоваться сторонними библиотеками. Одной из самых популярных является библиотека Gson из пакета com.google.code.gson.

Чтобы подключить её к проекту и упростить работу с JSON-объектами, необходимо добавить зависимость Gson в файл build.gradle нашего проекта.

На рисунке 1 показано добавление необходимой зависимости в Gradle.



Рисунок 1 – Добавление зависимости в Gradle

Теперь рассмотрим несколько примеров работы с JSON и классом User, который содержит три поля: имя, возраст и электронную почту. Создаём объект из JSON-строки (десериализация) и покажем это на рисунке 2.

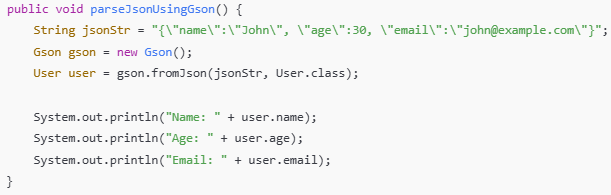


Рисунок 2 – Десериализация

В этом примере (рисунок 2) мы создаём объект Gson, после чего вызываем метод fromJson(), чтобы преобразовать JSON-строку в объект класса User. Мы передаём в метод строку с сериализованными данными и указываем класс, в который нужно выполнить преобразование. Таким образом, мы легко получаем доступ к значениям из JSON.

Далее преобразуем объект в JSON-строку (сериализация) и покажем это на рисунке 3.



Рисунок 3 – Сериализация

Здесь (рисунок 3) мы вручную создаём объект User и заполняем его данными. С помощью метода toJson() объекта Gson сериализуем его в строку в формате JSON. Полученный результат можно сохранить, например, в файл или передать по сети.

Далее чтение JSON-массива объектов. Код покажем на рисунке 4.



Рисунок 4 – Чтение массива объектов

В этом случае (рисунок 4) мы работаем с JSON-массивом, содержащим несколько объектов User. Чтобы корректно его обработать, мы определяем тип List<User> с помощью TypeToken. Затем десериализуем строку и перебираем полученные объекты в цикле, выводя информацию о каждом пользователе.

Таким образом, используя библиотеку Gson, мы можем эффективно выполнять как сериализацию, так и десериализацию как одиночных объектов, так и коллекций, что делает работу с JSON в Android простой и удобной.

## Провайдеры контента

Провайдеры контента (Content Providers) в Android — это компоненты, которые мы можем использовать для безопасного обмена данными между приложениями. С их помощью мы инкапсулируем данные внутри приложения и предоставляем унифицированный механизм доступа, соблюдая при этом контроль над тем, какие данные и кому открываются. Это особенно важно, когда несколько приложений нуждаются в доступе к одной и той же информации или когда одно приложение должно делиться данными с другим.

Провайдеры контента дают нам несколько ключевых преимуществ:

1. безопасность доступа: Мы можем предоставлять другим приложениям только нужные данные, скрывая внутреннюю структуру базы,
2. унифицированный интерфейс: Мы получаем стандартный механизм работы с данными, который упрощает интеграцию между разными приложениями,
3. удобное управление данными: С помощью методов query(), insert(), update(), delete() мы централизованно обрабатываем все операции,
4. интеграция с системой: Провайдеры легко взаимодействуют с такими компонентами, как загрузчики (Loaders) или система глобального поиска.

Допустим, у нас есть приложение, работающее с базой данных SQLite, в которой хранятся книги пользователя. Мы хотим разрешить другим приложениям обращаться к этим данным.

Шаг 1. Определим URI для доступа (рисунок 5).



Рисунок 5 – Определение URI

Шаг 2. Реализуем контент-провайдер (рисунок 6).



Рисунок 6 – Контент-провайдер

Шаг 3. Добавим провайдер в AndroidManifest.xml (рисунок 7).

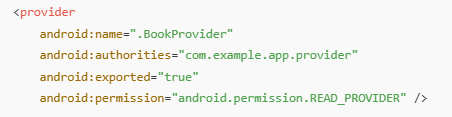


Рисунок 7 – Добавление провайдера в манифест

Шаг 4. В другом приложении запрашиваем разрешение на чтение (рисунок 8).



Рисунок 8 – Запрос на чтение

Шаг 5. Получаем данные о книгах из внешнего приложения (рисунок 9).



Рисунок 9 – Получение данных о книгах

Таким образом, мы предоставили доступ к данным из SQLite-базы одного приложения другим приложениям через стандартизированный механизм Content Provider. Это позволяет централизованно управлять безопасностью, форматом данных и политикой доступа, не раскрывая внутреннюю реализацию базы.

# 2 ПРАКТИЧЕСКАЯ ЧАСТЬ

## 2.1 Реализация передачи данных в другое приложение через провайдер контента

На рисунке 10 показан код класса UserProvider.java.

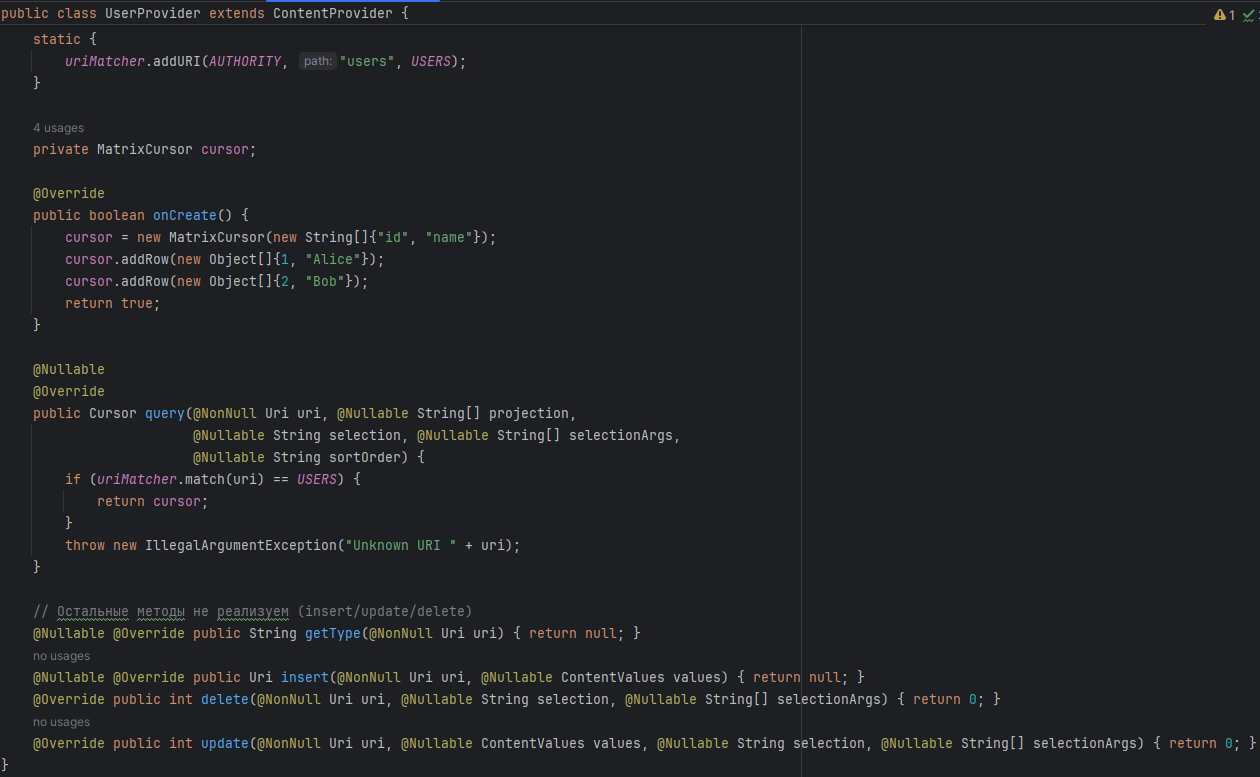


Рисунок 10 – Класс UserProvider

На рисунке 11 показаны изменения в AndroidManifest.

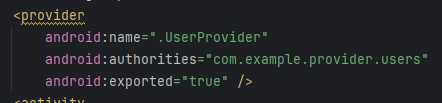


Рисунок 11 – Изменения в AndroidManifest

На рисунке 12 показан код MainActivity из второго приложения.



Рисунок 12 – MainActivity второго приложения

На рисунке 13 показан результат работы приложения.



Рисунок 13 – Работоспособность приложения

## 2.2 Преобразование данных в JSON и сохранение в отдельный файл, преобразование из JSON в различные поля

На рисунке 14 покажем разметку для activity\_main.xml.

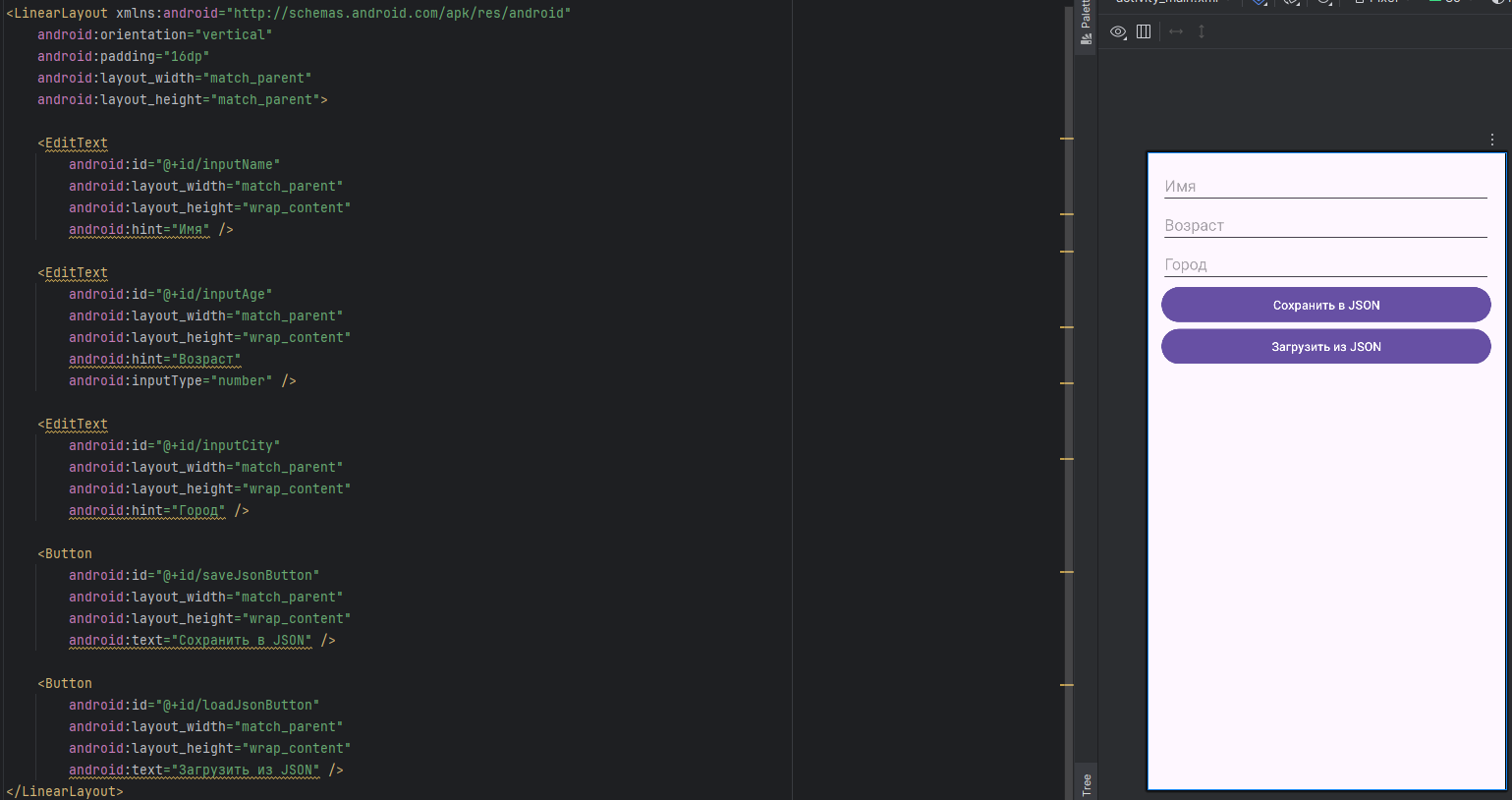


Рисунок 14 – Разметка activity\_main.xml

На рисунке 15 покажем первую часть MainActivity.java.



Рисунок 15 – MainActivity ч.1

На рисунке 16 покажем вторую часть класса MainActivity.



Рисунок 16 – MainActivity ч.2

На рисунке 17 покажем работоспособность приложения.

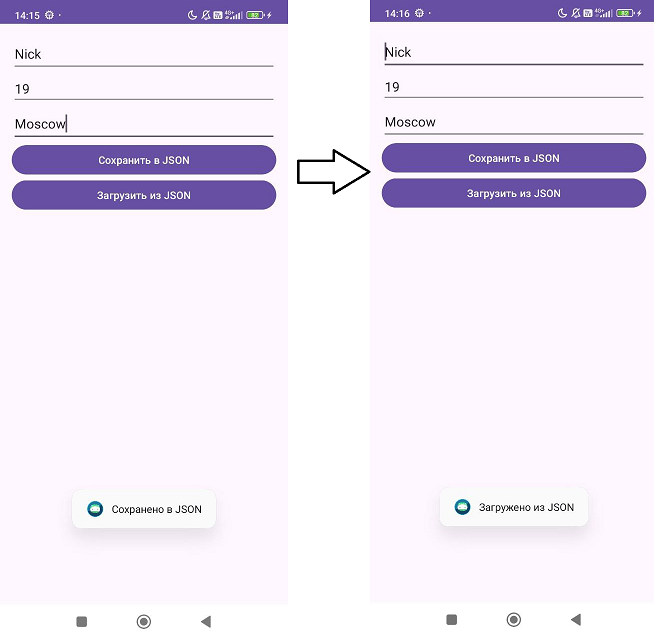


Рисунок 17 – Работоспособность приложения

# ЗАКЛЮЧЕНИЕ

В результате выполнения работы мы реализовали полноценный обмен данными между приложениями на Android с помощью контент-провайдера. Мы научились создавать URI, управлять запросами и настраивать безопасный доступ к базе данных. Помимо этого, мы освоили сериализацию объектов в формат JSON с использованием библиотеки Gson, сохранили данные во внешний файл, а затем успешно выполнили обратную операцию — десериализацию данных и отображение их в пользовательском интерфейсе. Таким образом, мы закрепили знания, необходимые для построения надёжных, гибких и взаимодействующих между собой Android-приложений.