ОГЛАВЛЕНИЕ

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ, ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ЕЕ ВЫПОЛНЕНИЯ

Целью выполнения лабораторной работы является формирование навыков создания собственных источников данных.

Основными задачами выполнения лабораторной работы являются:

1. Научиться создавать собственные источники данных (ContentProvider).
2. Научиться использовать собственные источники данных.
3. Научиться работать со стандартными источниками данных: аудио файлами, графическими файлами, списком контактов.
4. Разработать эффективные приложения с учетом аппаратных ограничений мобильных устройств.
5. Научиться реализовывать логику работы приложения с учетом специфики платформы Android

Результатами работы являются:

- Программа с несколькими Activity, использующая собственные источники данных (ContentProvider) и стандартными источниками данных для решения задачи согласно варианту задания
- Подготовленный отчет

CONTENT PROVIDER

Контент-провайдер или "Поставщик содержимого" (ContentProvider) - это оболочка (wrapper), в которую заключены данные. Если приложение использует базу данных SQLite, то только оно имеет к ней доступ. Но бывают ситуации, когда данные желательно сделать общими. Простой пример - контакты из телефонной книги тоже содержатся в базе данных, но желательно иметь доступ к данным, чтобы приложение тоже могло выводить список контактов. Так как нет доступа к базе данных чужого приложения, был придуман специальный механизм, позволяющий делиться своими данными всем желающим.

Поставщик содержимого применяется лишь в тех случаях, когда нужно использовать данные совместно с другими приложениями, работающих в устройстве. Но даже если не планируется делиться данными, то всё равно можно подумать об реализации этого способа на всякий случай.

В Android существует возможность выражения источников данных (или поставщиков данных) при помощи передачи состояния представления - REST, в виде абстракций, называемых поставщиками содержимого. Базу данных SQLite можно заключить в поставщик содержимого. Чтобы получить данные из поставщика содержимого или сохранить в нём новую информацию, нужно использовать набор REST-подобных идентификаторов URI. Например, если бы было нужно получить набор книг из поставщика содержимого, в котором заключена электронная библиотека, следовало бы использовать такой URI (по сути запрос к получению всех записей таблицы books):

content://com.android.book.bookprovider/books

Чтобы получить из библиотеки конкретную книгу (например, книгу №23), будет использоваться следующий URI (отдельный ряд таблицы):

content://com.android.book.bookProvider/books/23

Любая программа, работающая в устройстве, может использовать такие URI для доступа к данным и осуществления с ними определенных операций. Следовательно, поставщики содержимого играют важную роль при совместном использовании данных несколькими приложениями.

Создание собственного контент-провайдера

Для создания собственного контент-провайдера нужно унаследоваться от абстрактного класса ContentProvider:

```
class MyContentProvider : ContentProvider()
    {
    }
```

В классе необходимо реализовать абстрактные методы query(), insert(), update(), delete(), getType(), onCreate(). Прослеживается некоторое сходство с созданием обычной базы данных.

А также его следует зарегистрировать в манифесте с помощью тега provider с атрибутами name и authorities. Тег authorities служит для описания базового пути URI, по которому ContentResolver может найти базу данных для взаимодействия. Данный тег должен быть уникальным, поэтому рекомендуется использовать имя вашего пакета, чтобы не произошло путаницы с другими приложениями, например:

```
<provider
android:name:".MyContentProvider"
android:authorities="ru.anything.provider.notepad" />
```

Источник поставщика содержимого аналогичен доменному имени сайта. Если источник уже зарегистрирован, эти поставщики содержимого будут представлены гиперссылками, начинающимися с соответствующего префикса источника:

content://ru.anything.provider.notepad/

Итак, поставщики содержимого, как и веб-сайты, имеют базовое доменное имя, действующее как стартовая URL-страница.

Необходимо отметить, что поставщики содержимого, используемые в Android, могут иметь неполное имя источника. Полное имя источника рекомендуется использовать только со сторонними поставщиками содержимого. Поэтому вам иногда могут встретиться поставщики содержимого, состоящие из одного слова, например contacts, в то время как полное имя такого поставщика содержимого - com.google.android.contacts.

В поставщиках содержимого также встречаются REST-подобные гиперссылки, предназначенные для поиска данных и работы с ними. В случае описанной выше регистрации унифицированный идентификатор ресурса, предназначенный для обозначения каталога или коллекции записей в базе данных NotePadProvider, будет иметь имя:

content://ru.anything.provider.notepad/notes

URI для идентификации отдельно взятой записи будет иметь вид:

content://ru.anything.provider.notepad/notes/#

Символ # соответствует конкретной записи (ряд таблицы). Ниже приведено еще несколько примеров URI, которые могут присутствовать в поставщиках содержимого:

content://media/internal/images
content://media/external/images
content://contacts/people/
content://contacts/people/23

Обратите внимание – здесь поставщики содержимого content://media и content://contacts имеют неполную структуру. Это обусловлено тем, что данные поставщики содержимого не являются сторонними и контролируются Android.

Встроенные поставщики

Android предоставляет доступ к нескольким стандартным базам данных, используя Источники данных. Так, можно использовать эти Источники данных напрямую с помощью методов, описанных ранее в этой главе. Однако есть возможность воспользоваться пакетом android.provider, который содержит классы, упрощающие доступ ко многим из наиболее полезных источников.

- Browser. Используйте этот Источник данных для чтения или изменения закладок, истории посещений или использования поиска в обозревателе.
- CallLog. Выводит или обновляет историю звонков (входящие, исходящие, пропущенные), открывает доступ к детальной информации, такой как номер звонившего и продолжительность разговора.
- ContactsContract. Используйте этот Источник данных для получения, изменения или хранения информации о контактах. Он заменяет старый класс Contacts.
- MediaStore. Этот Источник данных предоставляет централизованный, управляемый доступ к файлам мультимедиа на устройстве, включая аудио, видео и изображения. Отсюда имеется возможность хранить в нем собственные файлы мультимедиа и делать их общедоступными.
- Settings. Вы можете получить доступ к настройкам устройства с помощью этого Источника данных. Предоставляется возможность просматривать большинство системных

настроек, а некоторые из них даже изменять. Класс android.provider.

- Settings содержит коллекцию действий для Намерений, которые могут использоваться для открытия соответствующего экрана с настройками, чтобы пользователь мог их поменять.
- UserDictionary. Обеспечивает доступ к пользовательским наборам слов, добавленных в словарь для интеллектуального ввода текста. Нужно использовать эти стандартные Источники данных везде, где возможно, чтобы обеспечить гармоничную интеграцию приложения с другими программами.

Использование Источника данных MediaStore

MediaStore в Android можно назвать хранилищем аудио-, видеофайлов, а также изображений. Каждый раз, когда на файловую системузаписывается файл мультимедиа, он также должен быть добавлен в MediaStore. Это откроет доступ к нему для других приложений, включая стандартный медиапроигрыватель.

Чтобы получить доступ к медиафайлам из MediaStore, нужно запрашивать их через Источники данных. Класс MediaStore включает подклассы Audio, Video и Images, которые, в свою очередь, содержат подклассы, обеспечивающие доступ к именам столбцов и путям URI, ссылающимся на содержимое каждого Источника данных.

MediaStore упорядочивает файлы мультимедиа, хранящиеся на внутренних и внешних носителях устройства. Каждый из его подклассов предоставляет путь URI ко внутренним или внешним файлам, используя шаблоны вида:

- MediaStore..Media.EXTERNAL_CONTENT_URI;
- MediaStore..Media.INTERNAL_CONTENT_URI.

Далее показан фрагмент кода, в котором идет поиск названия песни и альбома в каждом аудиофайле, хранящемся на внутреннем или внешнем носителе.

```
    // получить курсор, ссылающийся на все аудиофайлы, хранящиеся
    // на внутреннем или внешнем носителе.
var cursor = getContentResolver().query(
   MediaStore.Audio
      .Media.EXTERNAL_CONTENT_URI, null, null, null, null
)
    // дать возможность Активности управлять жизненным циклом курсора.
    startManagingCursor(cursor);
    // использать удобные свойства для получения индексов столбцов
    val albumIdx = cursor.getColumnIndexOrThrow(MediaStore.Audio
        .Media. ALBUM)
    val titleIdx = cursor. getColumnIndexOrThrow(MediaStore.Audio
        .Media. TITLE)
    String[] result = new String[cursor.getCount()];
    if (cursor.moveToFirst())
      do {
            // извлечь название песни.
            String title = cursor.getString(titleIdx);
            // Извлечь название альбома.
            String album = cursor.getString(albumIdx);
            result[cursor.getPosition()] = title + " (" + album + ")";
      }
      while(cursor.moveToNext());
```

РАБОТА С КОНТАКТАМИ

Контакты в Android обладают встроенным API, который позволяет получать и изменять список контактов. Все контакты хранятся в базе данных SQLite, однако они не представляют единой таблицы. Для контактов отведено три таблицы, связанных отношением один-ко-многим: таблица для хранения информации о людях, таблица их телефонов и таблица адресов их электронных почт. Но благодаря Android API мы можем абстрагироваться от связей между таблицами.

Общая форма получения контактов выглядит следующим образом:

```
ArrayList<String> contacts = new ArrayList<String>();
ContentResolvercontentResolver = getContentResolver();
Cursor cursor = contentResolver.query(ContactsContract.Contacts
    .CONTENT_URI, null, null, null, null);
if(cursor!=null){
while (cursor.moveToNext()) {
       // получить каждый контакт
       String contact = cursor.getString(cursor.getColumnIndex
          (ContactsContract.Contacts.DISPLAY_NAME_PRIMARY));
// добавить контакт в список
contacts.add(contact);
   }
cursor.close();
}
```

Все контакты и сопутствующий функционал хранятся в специальных базах данных SQLite. Но нам не надо напрямую работать с ними. Мы можем воспользоваться объектом класса Cursor. Чтобы его получить, сначала вызывается метод getContentResolver(), который возвращает объект ContentResolver. Затем по цепочке вызывается метод query(). В этот метод передается ряд параметров, первый из которых представляет URI - ресурс, который мы хотим получить. Для

обращения к базе данных контактов используется константа ContactsContract.Contacts.CONTENT_URI

Метод contactsCursor.moveToNext() позволяет последовательно перемещаться по записям контактов, считывая по одному контакту через вызов contactsCursor.getString().

Таким образом, получать контакты не сложно. Главная сложность в работе с контактами, да и с любыми другими провайдерами контента, заключается в установке разрешений. До Android API 23 достаточно было установить соответствующее разрешение в файле манифеста приложения. Начиная же с API 23 (AndroidMarshmallow) Google изменил схему работы с разрешениями. И теперь пользователь сам должен решить, будет ли он давать разрешения приложению. В связи с чем разработчики должны добавлять дополнительный код.

Чтение контактов

Чтобы получить доступ к любой контактной информации, необходимо добавить полномочие READ_CONTACTS к манифесту вашего приложения:

```
<uses-permission
android:name="android.permission.READ_CONTACTS"/>
```

Добавление контактов

Кроме создания запросов к базе данных контактов можно использовать эти Источники данных для изменения, удаления или вставки записей о контактах, добавив полномочие android.permission.WRITE_CONTACTS в манифест приложения.

Благодаря принципу расширяемости ContactsContract вы можете добавлять произвольные строки в таблице Data к любой учетной записи, которая хранится в виде RawContacts. На деле это не лучший способ расширения сторонних учетных записей, так как нельзя синхронизировать новую информацию с удаленным сервером.

Более удачное решение — создание собственного Адаптера для синхронизации, объединенного с другими сторонними данными об учетной записи.

Можно добавлять новые записи в Источник Data, которые будут привязаны к учетной записи контакта. После добавления они объединятся с информацией, предоставляемой стандартными и сторонними Адаптерами, и станут доступны при запросах к Источнику ContactsContract.

ТРЕБОВАНИЯ К РЕАЛИЗАЦИИ

Программа должна быть реализована на языке высокого уровня Kotlin.

ВАРИАНТЫ ЗАДАНИЙ

1. В отдельном приложении создать собственный источник данных: Студенты: ФИО, факультет, группа. Создать приложение-клиент для работы с созданным источником данных. Предусмотреть возможность: добавления новой информации, обновления имеющихся данных и удаления данных. Параметры для добавления, удаления и обновления вводятся на одном Activity, а итоговый список отображается на другом Activity.

2. В отдельном приложении создать собственный источник данных: Машины: Марка, модель, год выпуска.Создать приложение-клиент для работы с созданным источником данных. Предусмотреть возможность: добавления новой информации, обновления имеющихся данных и удаления данных. Параметры для добавления, удаления и обновления вводятся на одном Activity, а итоговый список отображается на другом Activity.

3. В отдельном приложении создать собственный источник данных: Телефоны: Производитель, модель, год выпуска.Создать приложение-клиент для работы с созданным источником данных. Предусмотреть возможность: добавления новой информации, обновления имеющихся

данных и удаления данных. Параметры для добавления, удаления и обновления вводятся на одном Activity, а итоговый список отображается на другом Activity.

4. В отдельном приложении создать собственный источник данных: Книги: Автор, название, год выпуска.Создать приложение-клиент для работы с созданным источником данных. Предусмотреть возможность: добавления новой информации, обновления имеющихся данных и удаления данных. Параметры для добавления, удаления и обновления вводятся на одном Activity, а итоговый список отображается на другом Activity.

5. В отдельном приложении создать собственный источник данных: Страны: Название, столица, форма правления.Создать приложение-клиент для работы с созданным источником данных. Предусмотреть возможность: добавления новой информации, обновления имеющихся данных и удаления данных. Параметры для добавления, удаления и обновления вводятся на одном Activity, а итоговый список отображается на другом Activity.

6. В виде списка вывести информацию об аудио файлах, а именно: Название, исполнитель, длительность, жанр. Элементы списка отсортировать по исполнителю. Длительность преобразовать к стандартному виду(минуты: секунды).

7. Вывести список названий композиций данного исполнителя. Исполнитель вводится на одном Activity. Список выводится на другом Activity.

8. Вывести информацию о композициях данного исполнителя (название, длительность). Длительность преобразовать к стандартному виду (минуты:секунды). Исполнитель выбирается из компонента Spinner.

9. Вывести список композиций (исполнитель, название) длительность которых находится в определенном диапазоне. Диапазон выбирается с помощью компонентов EditText, находящихся на одной Activity, а список выводится на втором. Диапазон указывается в минутах и секундах.

10. В первом Activity вывести список исполнителей. По щелчку на один из элементов списка, открывать второй Activity со списком названий композиций данного исполнителя.

11. Вывести список контактов: имя, ФИО, e-mail, домашний и мобильный телефон. Элементы списка отсортировать по имени.

12. На первом Activity вводится искомая фамилия (или часть фамилии), на втором Activity выводится искомый список (Имя, Фамилия, моб. телефон). Фамилии во втором списке соответствуют запросу.

13. Вывести в виде списка изображения и их названия.

14. На первом Activity вводится необходимая фамилия (или часть фамилии), на втором Activity выводится искомый список (Имя, Фамилия, моб. телефон). Фамилии во втором списке соответствуют запросу. Второй список отсортировать по имени.

15. Вывести, выбранное из галереи, изображение. Указать также его отображаемое название и размер (в кбайтах).

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Раскройте значение терминаContentProvider.
2. Дайте определение URI.
3. Опишите механизмсоздания собственногоконтент-провайдера.
4. Перечислите методы, которые необходимо реализовать в классе, унаследованном от ContentProvider.
5. Приведите пример встроенных контент-провайдеров.
6. Опишите использование источникаданныхMediaStore.
7. Опишите принцип работы с контактами.
8. Приведите примеры основных операций с контактами.

ФОРМА ОТЧЕТА ПО ЛАБОРАТОРНОЙ РАБОТЕ

На выполнение лабораторной работы отводится 2 занятия (4 академические часа: 3 часа на выполнение и сдачу лабораторной работы и 1 час на подготовку отчета).

Номер варианта студенту выдается преподавателем.

Отчет на защиту предоставляется в печатном виде.

Структура отчета (на отдельном листе(-ах)): титульный лист, формулировка задания (вариант), листинг (xml код разметки экрана, графическое представление, соответствующее разметке экрана, код программы и при необходимости наличие кода дополнительных классов), результаты выполнения работы (графические изображения примеров работы приложения), выводы.

ОСНОВНАЯ ЛИТЕРАТУРА

1. Семакова, А. Введение в разработку приложений для смартфонов на ОС Android / А. Семакова. - 2-е изд., испр. - М. : Национальный Открытый Университет «ИНТУИТ», 2016. - 103 с. : ил. ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=429181
2. Введение в разработку приложений для ОС Android / Ю.В. Березовская, О.А. Юфрякова, В.Г. Вологдина и др. - 2-е изд., испр. - М. : Национальный Открытый Университет «ИНТУИТ», 2016. - 434 с. : ил. - Библиогр. в кн. ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=428937
3. Разработка приложений для смартфонов на ОС Android / Е.А. Латухина, О.А. Юфрякова, Ю.В. Березовская, К.А. Носов. - 2-е изд., исправ. - М. : Национальный Открытый Университет «ИНТУИТ», 2016. - 252 с. : ил. - Библиогр. в кн. ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=428807

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

4. Дэвид, Х. Разработка приложений Java EE 6 в NetBeans 7. [Электронный ресурс] — Электрон. дан. — М. : ДМК Пресс, 2013. — 330 с. — Режим доступа: http://e.lanbook.com/book/58693 — Загл. с экрана.
5. Сильвен, Р. Android NDK. Разработка приложений под Android на C/C++. [Электронный ресурс] — Электрон. дан. — М. : ДМК Пресс, 2012. — 496 с. — Режим доступа: http://e.lanbook.com/book/9126 —Загл. с экрана.
6. Ретабоуил, С. Android NDK: руководство для начинающих. [Электронный ресурс] — Электрон. дан. — М. : ДМК Пресс, 2016. — 518 с. — Режим доступа: http://e.lanbook.com/book/82810 — Загл. с экрана.

7. Соколова, В.В. Разработка мобильных приложений: учебное пособие / В.В. Соколова ; Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Томский государственный университет», Министерство образования и науки Российской Федерации. - Томск: Издательство Томского политехнического университета, 2015. - 176 с.: ил., табл., схем. - Библиогр. в кн.. - ISBN 978-5-4387-0369-3; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=442808 (15.06.2017).
8. Баженова, И.Ю. Язык программирования Java / И.Ю. Баженова. - М.: Диалог-МИФИ, 2008. - 254 с. : табл., ил. - ISBN 5-86404-091-6 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=54745 (15.06.2017).
9. Джошуа Блох Java. Эффективное программирование [Электронный ресурс]/ Джошуа Блох— Электрон. текстовые данные. — Саратов: Профобразование, 2017.— 310 с.— Режим доступа: http://www.iprbookshop.ru/64057.html .— ЭБС «IPRbooks»

Электронные ресурсы:

10. Научная электронная библиотека http://eLIBRARY.RU
11. Электронно-библиотечная система http://e.lanbook.com
12. Электронно-библиотечная система «Университетская библиотека онлайн» http://biblioclub.ru
13. Электронно-библиотечная система IPRBookhttp://www.iprbookshop.ru
14. Базовый сайт о системе Android - https://www.android.com/intl/ru_ru
15. Разработка приложения на базе Android - https://developer.android.com/index.html