Министерство образования Республики Беларусь
Учреждение образования «Витебский государственный
университет имени П.М. Машерова»
Кафедра прикладного и системного программирования

Т.Д. Жук

ПРОГРАММИРОВАНИЕ МОБИЛЬНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Методические рекомендации

Витебск
ВГУ имени П.М. Машерова
2023

УДК 004.42:621.395.721.5(075.8)
ББК 32.973я73+32.884.161я73
Ж85

Печатается по решению научно-методического совета учреждения образования «Витебский государственный университет имени П.М. Машерова». Протокол № 7 от 26.04.2023.

Автор: преподаватель кафедры прикладного и системного программирования ВГУ имени П.М. Машерова, магистр **Т.Д. Жук**

Р е ц е н з е н т :
заведующий кафедрой информационных систем
и автоматизации производства УО «ВГТУ»,
кандидат технических наук, доцент *В.Е. Казаков*

Жук, Т.Д.

Ж85 Программирование мобильных информационных систем : методические рекомендации / Т.Д. Жук. – Витебск : ВГУ имени П.М. Машерова, 2023. – 21 с.

В методических рекомендациях изложены основные принципы применения языка программирования Java с целью создания мобильных информационных систем для операционной системы Android, структура приложений и способы взаимодействия с базами данных.

Предназначаются для студентов первой ступени высшего образования специальностей: «Информационные системы и технологии» (дисциплина «Программирование мобильных информационных систем»); «Прикладная информатика (веб-программирование и компьютерный дизайн). Веб-программирование и компьютерная графика» (дисциплина «Технологии программирования для мобильных приложений»).

УДК 004.42:621.395.721.5(075.8)
ББК 32.973я73+32.884.161я73

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Операционная система Android предназначена для управления различными устройствами, включая мобильные телефоны, планшеты, электронные книги, музыкальные плееры, наручные часы и игровые приставки. Android была создана компанией Android Inc. (ныне принадлежит Google) на основе ядра Linux.

Android является самой медийной мобильной операционной системой в мире. Ее популярность только возрастает с момента его запуска компанией Android еще в 2008 году. На 2022 год в мире насчитывается более 3 миллиардов устройств Android, что составляет 59% рынка.

В данных методических рекомендациях изложены базовые принципы применения языка программирования Java для создания мобильных информационных систем для операционной системы Android, структура приложений и способы взаимодействия с базами данных.

Первая глава позволяет познакомиться с основными компонентами Android приложений и структурой проекта.

Вторая глава помогает освоить систему взаимодействия и передачи данных между разными структурными единицами проекта.

В третьей главе рассматриваются примеры организации запросов к базе данных, способы создания, удаления и обновления записей базы данных.

Материал соответствует темам учебных программ курсов: «Программирование мобильных информационных систем» (специальность «Информационные системы и технологии (в здравоохранении)»); «Технологии программирования для мобильных приложений» (специальность «Прикладная информатика (веб-программирование и компьютерный дизайн). Веб-программирование и компьютерная графика»).

1. СТРУКТУРА ПРИЛОЖЕНИЯ ANDROID

1.1. Компоненты приложения

Компоненты приложения – это блоки, из которых состоит приложение. Каждый компонент представляет собой отдельную точку входа в приложение для системы. Не все компоненты являются точками входа для пользователя. Но каждый компонент является самостоятельной структурной единицей, определяющей поведение приложения в целом. Компоненты приложения можно отнести к одному из пяти типов.

Операция (Activity, активность) представляет собой отдельный экран с пользовательским интерфейсом. Например, в приложении электронной почты одна активити отображает список новых сообщений, другая позволяет пользователю составить сообщение, а третья – прочитать его. Все действия в совокупности формируют взаимодействие между пользователем и приложением, но каждое из них не зависит от других.

Любое активити может быть вызвано другим приложением. Например, приложение камеры может инициировать действие в приложении электронной почты, которое создает новое сообщение, чтобы пользователь мог отправить фотографию. Этот процесс может быть запущен подклассом Activity.

Служба (Service, сервис) – это компонент, работающий в фоновом режиме и выполняющий длительные операции, связанные с работой удаленного процесса. Сервисы не имеют пользовательского интерфейса. Например, они могут проигрывать музыку в фоновом режиме, пока пользователь работает с другим приложением, или получать данные по сети, не прерывая взаимодействия с пользователем. Службы могут быть инициированы другими компонентами, которые, в свою очередь, могут взаимодействовать с сервисом. Службы принадлежат к подклассу Service.

Поставщик контента (Content provider) управляет общим набором данных приложения. Данные могут храниться в файловой системе, в базе данных SQLite, в Интернете или в другом месте хранения, доступном приложению. Через контент-провайдера другие приложения могут запрашивать и изменять данные (если это разрешено контент-провайдером). Например, в системе Android есть поставщик контента, который управляет информацией контактов пользователя. Приложения с соответствующими правами могут запрашивать чтение и запись частей этого контент-провайдера. Контент-провайдер относится к подклассу ContentProvider и должен реализовывать стандартный набор интерфейсов API.

Приемник широковещательных сообщений (Broadcast receiver) представляет собой компонент реагирующий на общесистемные объявления. Многие из этих сообщений посылаются системой в виде уведомлений о выключении экрана, разрядке аккумулятора, сделанной фотографии и т.д.

Объявления также могут посылаться приложениями. Например, другие приложения могут получать уведомления о том, что определенные данные загружены в устройство и готовы к использованию. Приемники широковещательных сообщений не имеют пользовательского интерфейса, но они могут генерировать уведомление в строке состояния, чтобы сообщить пользователю о событии «рассылка объявления». В большинстве случаев приемник широковещательных сообщений является шлюзом для других компонентов и предназначен для выполнения минимальной работы. Приемники широковещательных сообщений относится к подклассу BroadcastReceiver.

Уникальной особенностью системы Android является то, что любое приложение может запустить компонент другого приложения. Для пользователя это будет выглядеть как одно приложение. Когда система запускает компонент, она запускает процесс для этого приложения (если он еще не был запущен) и создает экземпляры классов, которые требуются этому компоненту. Поэтому в отличие от приложений для большинства других систем в приложениях для Android отсутствует единая точка входа (в них нет функции *main()*). Каждое приложение выполняется системой в отдельном процессе с такими правами доступа к файлам, которые ограничивают доступ другим приложениям. Поэтому одно приложение не может напрямую вызвать компонент из другого приложения. Но это может сделать система Android. Для того чтобы вызвать компонент в другом приложении, необходимо сообщить системе о своем намерении (*Intent*). После чего система активирует этот компонент.

1.2. Структура проекта

Все модули в проекте описываются файлом setting.gradle. Каждый модуль имеет свой файл build.gradle, который определяет конфигурацию построения проекта. Каталоги androidTest и test предназначены для хранения файлов тестов приложения, а каталог java – для хранения исходного кода.

Файл AndroidManifest.xml представляет собой файл манифеста, который описывает фундаментальные характеристики приложения, его конфигурацию и определяет каждый из компонентов данного приложения. Для запуска компонента системе Android необходимо знать, что компонент существует. Для этого она читает файл AndroidManifest.xml приложения, который должен находиться в его корневой папке.

Системе не видны активности, службы и поставщики контента, которые присутствуют в исходном коде, но не объявлены в манифесте, следовательно они не могут быть запущены. Манифест помимо объявления компонентов приложения служит и для других целей. Например, установка всех полномочий пользователя, которые необходимы приложению (разрешения на доступ в интернет или на чтение контактов пользователя), для объявления минимального уровня API, необходимого приложению с учетом того,

какие API-интерфейсы оно использует, для объявления аппаратных и программных функций, которые нужны приложению или используются им, например функции камеры, службы Bluetooth или сенсорного экрана, для указания библиотек API, с которыми нужно связать приложение, например библиотеки Google Maps и др.

Папка res содержит каталоги с ресурсами:

– папка drawable предназначена для хранения изображений, используемых в приложении;

– папка layout предназначена для хранения файлов, определяющих графический интерфейс. По умолчанию здесь есть файл activity_main.xml, который определяет интерфейс для единственной в проекте активности – MainActivity;

– папки mipmap-xxxx содержат файлы изображений, которые предназначены для создания иконки приложения при различных разрешениях экрана. Соответственно для каждого вида разрешения здесь имеется свой каталог;

– папка values хранит различные XML-файлы, содержащие коллекции ресурсов.

Используя ресурсы приложения, возможно без труда менять его характеристики, не изменяя код, и, кроме того, путем предоставления наборов альтернативных ресурсов возможно оптимизировать приложение для работы с различными конфигурациями устройств (например, для различных языков либо разрешений экрана). Для каждого ресурса, используемого в проекте Android, инструменты SDK задают уникальный целочисленный идентификатор, который может использоваться, для ссылки на ресурс из кода приложения или из других ресурсов, определенных в XML.

Класс MainActivity имеет следующую структуру:

```
    package by.myapplication;

    import androidx.appcompat.app.AppCompatActivity;
    import android.os.Bundle;

     public class MainActivity extends AppCompatActivity {

     @Override
     protected void onCreate(Bundle savedInstanceState) {

          super.onCreate(savedInstanceState);
          setContentView(R.layout.activity_main);
    }
}
```

2. ПЕРЕДАЧА ДАННЫХ МЕЖДУ ACTIVITY

2.1. Intent

Интент (intent, намерение) – объект, который может использоваться компонентом для взаимодействия с ОС. Интенты представляют собой универсальные средства передачи информации, а класс Intent предоставляет разные конструкторы в зависимости от того, для чего должен использоваться интент.

При создании объекта Intent с объектом Context и Class создается ***явный*** (explicit) интент. Явные интенты применяются для запуска активностей в приложениях. Когда активность в вашем приложении должна запустить активность в другом приложении, вы создаете ***неявный*** (implicit) интент.

Рассмотрим сведения, которые содержит Intent.

Имя компонента, который необходимо запустить. Эта информация является необязательной, но именно она и делает объект Intent явным. При отсутствии названия компонента объект Intent является неявным. Имя можно задать через объект ComponentName(), setComponent(), setClass(), setClassName() или конструктор Intent.

Действие – это строка, характеризующая стандартное действие, которое необходимо выполнить (view, pick, …). Действие в значительной степени определяет, как будет структурирована остальная часть объекта Intent. Для использования Intent в пределах своего приложения можно указать собственные действия. Обычно следует использовать константы действий, определенные классом Intent или другими классами платформы (например, ACTION_VIEW, ACTION_SEND и др.) Задается методом setAction() или конструктором Intent.

Данные – это объект URI, ссылающийся на данные, с которыми будет выполняться действие и (или) тип MIME этих данных. Тип данных определяется действием объекта Intent. Например, если действием является ACTION_EDIT, в данных долженнаходиться URI документа, который необходимо отредактировать. Чтобы задать только URI данных, вызывается setData(). Чтобы задать только тип MIME – setType(). При необходимости установить оба этих параметра – setDataAndType().

Категория – строка, содержащая прочие сведения о том, каким компонентом должна выполняться обработка этого объекта Intent. В объект Intent можно поместить любое количество описаний категорий, но большинству объектов Intent категория не требуется. Например, стандартные категории:

- CATEGORY_BROWSABLE (целевая активность позволяет запускать себя веб-браузером для отображения данных, указанных по ссылке);
- CATEGORY_LAUNCHER (активность является начальной для задачи). Указать категорию можно с помощью addCategory().

Дополнительные данные – это пары «ключ – значение», содержащие прочую информацию. Добавлять дополнительные данные можно с помощью методов putExtra(), каждый из которых принимает два параметра: имя и значение ключа. Можно создать объект Bundle со всеми дополнительными данными, затем вставить объект Bundle в объект Intent с помощью метода putExtras().

Флаги, определенные в классе Intent, действуют как метаданные. Флаги должны указывать ОС, каким образом следует запускать активность (например, к какой задаче должна принадлежать активность и как с ней обращаться после запуска).

2.2. Запуск активности

Чтобы запустить одну активность из другой, проще всего воспользоваться методом startActivity():

```
public void startActivity(Intent intent);
```

Когда активность вызывает startActivity(), этот вызов передается компоненту ОС, который называется ActivityManager. ActivityManager создает экземпляр Activity и вызывает его метод onCreate().

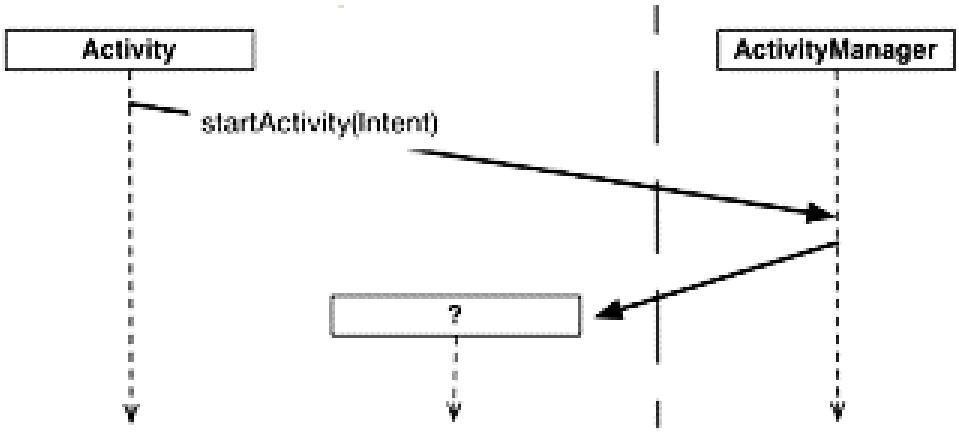

Рис. Запуск активности

Информацию о том, какую активность следует запустить, ActivityManager получает из параметра Intent.

```
Intent intent = new Intent(this, Target.class);
startActivity(intent);
```

При создании Intent используется конструктор с двумя параметрами. Первый параметр – это Context. Здесь Activity является подклассом Context (объект, который предоставляет доступ к базовым функциям приложения, таким как доступ к ресурсам и файловой системе, вызов Activity и т.д.). Второй параметр – имя класса.

2.3. Передача и получение значений из Activity

При переходе от одной активности к другой можно передавать различную информацию с помощью метода putExtra():

```
intent.putExtra("имя", значение);
```

Фактически это словарь, который состоит из пар «ключ – значение». Можно передавать не только строковые значения, но и другие, например числовые, логические. Вызывая метод getIntent(), можно получить объект Intent, а с помощью его метода getExtras() – те данные, которые ранее были переданы с объектом Intent. Для получения строковых данных используется метод getStringExtra(), для получения данных типа float – getFloatExtra(), а в качестве параметра используется ключ:

```
Intent intent = getIntent();
...
String string = intent.getStringExtra("имя");
int intNum = intent.getIntExtra("имя", 0);
```

После запуска активности может потребоваться получить результат. Для этого вызывается метод startActivityForResult() вместо startActivity(). Чтобы получить результат после выполнения последующей активности, необходимо реализовать метод обратного вызова onActivityResult(). По завершении она возвращает результат в объекте Intent в вызванный метод onActivityResult().

2.4. Задание неявного объекта Intent

В приложении может потребоваться выполнить некоторое действие, например отправить письмо, текстовое сообщение или обновить статус, используя данные из текущей Activity. В этом случае можно воспользоваться Activity из других приложений, имеющихся на устройстве. Например, если пользователю требуется предоставить возможность отправить электронное письмо, можно создать следующий Intent:

```
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, textMessage);
sendIntent.setType("text/plain");
if(sendIntent.resolveActivity(getPackageManager())
                                          != null) {
      startActivity(sendIntent);
}
```

Дополнительный компонент EXTRA_, добавленный в Intent, представляет собой текст письма. Когда почтовая программа реагирует на Intent, она считывает дополнительно добавленную строку и помещает ее в поле текста письма. При этом запускается активность почтовой программы, а после того, как пользователь завершит требуемые действия, возобновляется текущая активность.

Возможна ситуация, когда на устройстве пользователя не будет никакого приложения, которое может откликнуться на неявный объект Intent. В этом случае вызов закончится неудачей, а работа приложения аварийно завершится. Чтобы проверить, будет ли получен объект Intent, необходимо вызвать метод resolveActivity() для объекта Intent. Если результатом будет значение, отличное от null, значит, имеется хотя бы одно приложение, которое способно откликнуться на объект Intent, поэтому можно вызывать startActivity().

Рассмотрим еще один пример определения неявного объекта Intent:

```
Button sbut = (Button) findViewById(R.id.bSend);
sbut.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View view) {
     Intent sendIntent = new Intent();
     sendIntent.setAction(Intent.ACTION_SEND);
     sendIntent.putExtra(Intent.EXTRA_TEXT,
                                 "Send something here");
     sendIntent.setType("text/plain");
     Intent chooser = Intent.createChooser(sendIntent,
                                 "Choose the activity");
     if(sendIntent.resolveActivity(getPackageManager()
                                                  !=null){
          startActivity(chooser);
     }
    }
});
```

2.5. Настройка фильтров и разрешения для объектов Intent

Чтобы указать, какие неявные объекты Intent может принимать приложение, следует объявить один или несколько фильтров Intent для каждого компонента приложения с помощью элемента intent-filter в файле манифеста. Система передаст неявный объект Intent приложению, только если он может пройти через один из фильтров.

Явный объект Intent всегда доставляется целевому компоненту, без учета любых фильтров Intent, объявленных компонентом.

Компонент приложения должен объявлять отдельные фильтры для каждой уникальной работы, которую он может выполнить. Каждый фильтр Intent определяется элементом intent-filter в файле манифеста приложения, указанном в объявлении соответствующего компонента приложения.

Внутри элемента intent-filter можно указать тип объектов Intent с помощью одного или нескольких элементов:

– action – объявляет принимаемое действие, заданное в объекте Intent, в атрибуте name;

– data – объявляет тип принимаемых данных;

– category – объявляет принимаемую категорию, заданную в объекте Intent, в атрибуте name:

```
<intent-filter>
      <action>
      </action>

      <data>
      </data>

      <category>
      </category>
</intent-filter>
```

Можно создавать фильтры, в которых будет несколько экземпляров action, data, category. В этом случае просто нужно убедиться в том, что компонент может справиться с любыми сочетаниями этих элементов фильтра.

Чтобы объект Intent был доставлен компоненту, он должен пройти все три теста. Если он не будет соответствовать хотя бы одному из них, ОС не доставит этот объект Intent компоненту. Поскольку у компонента может быть несколько фильтров, объект Intent, который не проходит ни через один из них, может пройти через другой фильтр.

Когда система получает неявный объект Intent для запуска активности, она выполняет поиск путем сравнения объекта Intent с фильтрами по трем критериям: действие объекта Intent; данные объекта (структура URI и тип данных); категория объекта.

Для указания принимаемых действий объекта Intent фильтр может объявлять любое (в том числе нулевое) число элементов action:

```
<intent-filter>
 <action> android:name=”android.intent.action.EDIT” />
 <action> android:name=”android.intent.action.VIEW” />
…
</intent-filter>
```

Чтобы пройти через этот фильтр, действие, указанное в объекте Intent, должно соответствовать одному или нескольким действиям, перечисленным в фильтре. Если в фильтре не перечислены действия, объекту Intent будет нечему соответствовать, поэтому все объекты Intent не пройдут этот тест. Если в объекте Intent не указано действие, он пройдет тест. Действия задаются константами действий:

– ACTION_ANSWER – открывает активность, которая связана с входящими звонками;

– ACTION_CALL – инициализирует звонок по телефону;

– ACTION_DELETE – запускает активность, с помощью которой можно удалить данные, указанные в пути URI внутри Intent;

– ACTION_EDIT – отображает данные с возможностью редактирования пользователем;

– ACTION_INSERT – открывает активность для вставки в Cursor нового элемента, указанного с помощью пути URI;

– ACTION_HEADSET_PLUG – подключение наушников;

– ACTION_MAIN – запускается как начальная активность задания;

– ACTION_PICK – загружает дочернюю Activity, позволяющую выбрать элемент из источника данных, указанный с помощью URI;

– ACTION_SEARCH – запускает активность для выполнения поиска;

– ACTION_SEND – загружает экран для отправки данных, указанных в намерении;

– ACTION_SENDTO – открывает активность для отправки сообщений контакту, указанному в пути URI, который передается через Intent;

– ACTION_SYNC – синхронизирует данные сервера с данными мобильного устройства;

– ACTION_VIEW – наиболее распространенное общее действие.

Аналогичным образом происходит тестирование категории.

```
<intent-filter>
 <category>android:name=”android.intent.
                                    category.DEFAULT” />
 <category>android:name=”android.intent.
                                    category.BROWSABLE” />
…
</intent-filter>
```

Можно задать собственные категории или же брать стандартные значения, предоставляемые системой:

– ALTERNATIVE – действие должно быть доступно в качестве альтернативного тому, которое выполняется по умолчанию для компонента этого типа данных. Например, если действие по умолчанию для контакта – просмотр, то в качестве альтернативы его также можно редактировать;

– SELECTED_ALTERNATIVE – то же самое, что и ALTERNATIVE, но вместо одиночного действия применяется в тех случаях, когда нужен список различных возможностей;

– BROWSABLE – действие доступно из браузера;

– DEFAULT – делает компонент обработчиком по умолчанию для действия, выполняемого с указанным типом данных внутри фильтра;

– GADGET – активность может запускаться внутри другой активности;

– LAUNCHER – помещает Activity в окно для запуска приложений.

2.6. Лабораторная работа «Обработка событий»

Класс Dialog

Класс Dialog является базовым для всех классов диалоговых окон. Каждое диалоговое окно должно быть определено внутри активности, в которой будет использоваться. Для его отображения нужно вызвать метод showDialog() и передать в качестве параметра идентификатор диалога(константу, которую необходимо объявить в коде программы).

Метод dismissDialog() скрывает диалоговое окно (но не удаляет), не отображая его на экране. Окно остается в пуле диалоговых окон текущей активности. При повторном отображении с помощью метода showDialog() будет использована кешированная версия окна.

Метод removeDialog() удаляет окно из пула окон текущей активности. При повторном вызове метода showDialog() диалоговое окно придется создавать снова.

Рассмотрим пример создания диалогового окна на основе класса Dialog. Можно создать элементарную разметку для диалогового окна – текстовое поле внутри LinearLayout. А в разметку главной активности можно добавить кнопку для вызова диалогового окна:

```
public void onClick(View v) {
 dialog.show();
}
```

В коде для главной активности нужно определить:

```
public class MainActivity extends FragmentActivity {
 Dialog dialog;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 …
      dialog = new Dialog(MainActivity.this);
      // Устанавливаем заголовок
      dialog.setTitle("Dialog window");
```

```
        // Передаем ссылку на макет
        dialog.setContentView(R.layout.dialog);
        // Находим элемент TextView внутри разметки
        // и устанавливаем ему соответствующий текст
        TextView text = (TextView) dialog.findViewById
                                    (R.id.dialogTextView);
        text.setText("Delete list?");
    }
```

Метод onCreateDialog() вызывается один раз при создании окна. После изначального создания при каждом вызове метода showDialog() будет срабатывать обработчик onPrepareDialog(). Переопределив этот метод, можно изменять диалоговое окно при каждом его выводе на экран. Это позволит вложить контекст в любое из отображаемых значений. Если необходимо перед каждым вызовом диалогового окна менять его свойства (например, текстовое сообщение или количество кнопок), то это можно реализовать внутри метода.

Задание 1. Создайте приложение в соответствии с вариантом.

Варианты задания:

1. Вычисление диапазона ip-адресов подсети по адресу сети.
2. Побитовое И, ИЛИ, НЕ двух двоичных чисел.
3. Сложение и умножение двух матриц фиксированной размерности.
4. Калькулятор элементарных операций.
5. Нахождение количества вхождений подстроки в строке (с учётом регистра и без).

Задание 2. Создайте приложение, содержащее несколько полей разных типов и кнопку для сохранения данных. После нажатия кнопки, должно выводиться сообщение «Данные сохранены», если все поля заполнены верно, и «Данные не могут быть сохранены. Проверьте правильность заполнения полей» – в противном случае.

Указания: На форме должно присутствовать не менее 10 полей, минимум 5 из которых должны быть различных типов. Используйте класс Dialog (или класс наследующий Dialog) для вывода сообщения.

Варианты задания:

1. Полная форма регистрации пользователя в соц. сети.
2. Форма оформления заказа в интернет-магазине одежды.
3. Форма для соискателя вакансии на должность разработчика.
4. Формуляр читателя в библиотеке.
5. Форма для регистрации авто для продажи.
6. Форма для регистрации абитуриента при поступлении.
7. Форма заявления для регистрации брака.

8. Форма для подачи заявки на получение кредита в банке.
9. Форма для добавления товара в каталог.
10. Форма для подачи заявки на ремонт ноутбука.
11. Форма для заявки на получение визы.
12. Форма для покупки авиабилетов.

2.7. Лабораторная работа «Передача данных между Activity»

Задание 1. Вывод всех событий из календаря, запланированных на введенную пользователем дату.

Указания: Используйте Calendar Provider API.

Задание 2. Разработайте приложение, которое позволяет добавлять данные и просматривать их в табличном виде. Создайте класс (набор классов) для представления данных. Реализуйте проверку правильности ввода данных.

Указания: Добавление и просмотр должны быть реализованы в разных Activity и связываться посредством Intent.

Таблица 1

Вариант	Описание	Обязательные данные
1.	Компания. Список сотрудников компании	Имя, Филиал, Отдел, Номер, Должность, Город, Стаж
2.	Магазин. Учёт товаров магазина	Название, Категория, Остаток на складе, Номер поставщика, Стоимость, Закупочная стоимость
3.	Транспортная компания. Информация о перевозках	Уникальный идентификатор, Пункт отправления, Пункт прибытия, Дата отправления, Автомобиль, Водитель, Номер контактного лица
4.	Информационный ресурс. Список пользователей некоторого ресурса	Логин, Пароль, Почта, Информация «О себе», Дата регистрации, Дата последнего входа, Рейтинг
5.	Адресная книга	Имя, Номер, Адрес (город, улица, дом, корпус, квартира, индекс)
6.	Ателье. Список заказов ателье	Имя заказчика, Телефон заказчика, Вид изделия, Исполнитель, Дата приёма заказа, Дата выдачи заказа, Стоимость
7.	Риэлтерское агентство. Список недвижимости	Уникальный идентификатор, Тип недвижимости, Адрес, Менеджер, Номер продавца, Площадь, Стоимость
8.	Поликлиника. Список пациентов, наблюдающихся у участковых врачей	Имя, Адрес, Номер участка, Имя врача, Номер медицинской карты, Телефон
9.	Склад. Учёт товаров на складе	Уникальный идентификатор, Название, Количество, Поставщик, Единица измерения, Вес, Объём.
10.	Компания. Список проектов	Уникальный идентификатор проекта, Заказчик, Номер контактного лица, Менеджер проектов, Дата начала, Предполагаемая дата окончания, Бюджет

3. РАБОТА С БАЗОЙ ДАННЫХ

Механизм работы с базами данных в Android позволяет хранить и обрабатывать структурированную информацию. Любое приложение может создавать свои собственные базы данных, над которыми оно будет иметь полный контроль.

База данных SQLite доступна на любом Android-устройстве, ее не нужно устанавливать. Для работы большинства СУБД необходим специальный процесс сервера базы данных. SQLite обходится без сервера и представляет собой обычный файл.

Android хранит базы данных в каталоге /data/data/«имя_пакета»/databases на эмуляторе, на устройстве путь может отличаться. Метод Environment.getDataDirectory() возвращает путь к каталогу DATA.

По умолчанию все базы данных закрытые, доступ к ним могут получить только те приложения, которые их создали. Каждая база данных состоит из двух файлов. Имя первого файла базы данных соответствует имени базы данных. Это основной файл, в нем хранятся все данные. Второй файл – файл журнала. Его имя состоит из имени базы данных и суффикса *-journal*. В нем хранится информация обо всех внесенных изменениях. Если при работе с данными возникнет проблема, Android использует информацию журнала для отмены (или отката) последних изменений. Для просмотра можно воспользоваться Device File Explorer.

В большинстве случаев работа с базой данных происходит через специальные объекты Cursor, которые требуют наличия в таблице колонки с именем *_id*. Можно создать столбец вручную в коде, а можно использовать BaseColumn, который создаст столбец с нужным именем автоматически. Если не работать с курсорами, то можно использовать и стандартное наименование id или вообще не пользоваться данным столбцом. Но это не является рекомендуемой практикой.

3.1. Классы для работы с SQLite

Система Android содержит набор классов для работы с базами данных. Основная часть этой работы выполняется объектами трёх типов. Класс SQLiteDatabase позволяет выполнять запросы к базе данных и различные манипуляции с ней. Рассмотрим методы этого класса:

– query() – осуществляет чтение данных;
– insert() – производит вставку записей;
– delete() – отвечает за удаление записей;
– update() – обновляет записи таблицы;
– execSQL() – выполнение кода SQL;
– rawQuery() – выполнение «сырого» запроса.

Для чтения и записи данных используется класс SQLiteCursor. Класс SQLiteQueryBuilder предназначен для выпонения SQL-запросов. Сами SQL-выражения представлены классом SQLiteStatement, которые позволяют

вставлять в выражения динамические данные. Помощник SQLite создается расширением класса SQLiteOpenHelper. Он предоставляет средства для создания и управления базами данных и содержит два обязательных абстрактных метода:

– onCreate() – данный метод вызывается при первом создании базы данных;

– onUpgrade() – данный метод вызывается при модификации базы данных.

Также используются другие методы класса: onDowngrade(…), onOpen(…), getReadableDatabase(), getWritableDatabase().

Пример выражений для создания и удаления данных:

```
private static final String SQL_CREATE_ENTRIES =
        "CREATE TABLE " + DBEntry.TABLE_NAME + " (" +
        DBEntry._ID + " INTEGER PRIMARY KEY," +
        DBEntry.COLUMN_NAME_NAME +"TEXT" + "," +
        DBEntry.COLUMN_NAME_INFO +"TEXT" + "," +
        DBEntry.COLUMN_NAME_RATE + "INTEGER)";
private static final String SQL_DELETE_ENTRIES =
        "DROP TABLE IF EXISTS " + DBEntry.TABLE_NAME;
```

И класса помощника:

```
public class DbHelper extends SQLiteOpenHelper {

 public static final int DATABASE_VERSION = 2;
 public static final String DATABASE_NAME =
                                        "DBSimple.db";

 public DbHelper(Context context) {
      super(context, DATABASE_NAME,
                                null, DATABASE_VERSION);
 }

public void onCreate(SQLiteDatabase db) {
      db.execSQL(SQL_CREATE_ENTRIES);
}
public void onUpgrade(SQLiteDatabase db, int oldVersion,
                                        int newVersion){
      db.execSQL(SQL_DELETE_ENTRIES);
      onCreate(db);
}
}
```

Если на устройстве нет базы данных и класс должен создать ее, то вызывается метод onCreate(). У метода есть параметр db, относящийся к классу

SQLiteDatabase. У класса есть специальный метод execSQL(), которому нужно передать запрос (SQL-скрипт) для создания таблицы.

Для того чтобы использовать реализацию вспомогательного класса, нужно создать новый экземпляр класса, передать его конструктору контекст, имя базы данных, можно текущую (опционально), вызвать метод getReadableDatabase() или getWritableDatabase(), чтобы открыть и вернуть экземпляр базы данных.

```
DbHelper mDbHelper = new DbHelper(this);
…
// Получаем репозиторий в режиме записи
SQLiteDatabase db = mDbHelper.getWritableDatabase();
// Создаем новую запись
ContentValues values = new ContentValues();
values.put(DBContract.DBEntry.COLUMN_NAME_NAME,
                                              "Ivanov");
values.put(DBContract.DBEntry.COLUMN_NAME_INFO, "VSU");
values.put(DBContract.DBEntry.COLUMN_NAME_RATE, "8");
// Вставляем данные, и возвращается primary key
long newRowId;
newRowId = db.insert( DBContract.DBEntry.TABLE_NAME,
                                          null, values);
```

Метод insert() возвращает идентификатор _id вставленной строки или –1 в случае ошибки. Существует также второй способ вставки через метод execSQL(), когда подготавливается нужная строка и запускается скрипт. В этом варианте используется стандартный SQL-запрос INSERT INTO.

3.2. Лабораторная работа «Работа с базой данных»

Задание 1. Добавьте возможность авторизации в системе. Наложите ограничение на добавление информации для одной из ролей (должно быть минимум две).

Задание 2. Создайте базу данных и таблицы для хранения информации. Реализуйте добавление и удаление данных из базы.

Задание 3. Реализуйте запросы в соответствии с вариантом:

Таблица 2

Вариант	Описание	Запросы
1.	Компания. Список сотрудников компании	Вывести список всех сотрудников из заданного пользователем филиала Вывести список имен и номеров сотрудников со стажем больше заданного пользователем

2.	Магазин. Учёт товаров магазина	Вывести список товаров из категории заданной пользователем Вывести названия и номера поставщиков товаров с закупочной стоимостью меньшей, чем заданная пользователем
3.	Транспортная компания. Информация о перевозках	Вывести список перевозок с заданным пунктом прибытия Вывести все автомобили (и соответствующих им водителей), отбывающие в заданную дату
4.	Информационный ресурс. Список пользователей некоторого ресурса	Вывести список пользователей с заданным рейтингом Вывести все логины и почтовые адреса пользователей, зарегистрированных ранее указанной даты
5.	Адресная книга	Вывести адреса контактов, начинающихся на заданную букву Вывести имена и телефоны контактов из одного города
6.	Ателье. Список заказов ателье	Вывести все заказы исполнителя на заданную дату Вывести имена и телефонные номера заказчиков, чьи заказы не выданы
7.	Риэлтерское агентство. Список недвижимости	Вывести адреса и номера продавцов недвижимости, закреплённой за определенным менеджером Вывести информацию о недвижимости из заданного ценового промежутка
8.	Поликлиника. Список пациентов, наблюдающихся у участковых врачей	Вывести информацию о пациентах заданного врача Вывести имена и номера пациентов, закрепленных за конкретным участком
9.	Склад. Учёт товаров на складе	Вывести все товары конкретного поставщика Вывести название и количество товаров, превышающих заданный вес
10.	Компания. Список проектов	Вывести все проекты определенного менеджера проектов Вывести заказчика и номер контактного лица проектов с бюджетом выше заданного

СПИСОК РЕКОМЕНДОВАННЫХ ИСТОЧНИКОВ

1. Дейтел, П. Android для разработчиков / П. Дейтел, Х. Дейтел, А. Уолд. – 3-е изд. – СПб.: Питер, 2016. – 512 с.

2. Android. Программирование для профессионалов / Б. Харди [и др.]. – 2-е изд. – СПб.: Питер, 2016. – 640 с.

3. Гриффитс, Д. Head First. Программирование для Android / Д. Гриффитс. – СПб.: Питер, 2016. – 704 с.