Приложения под Андроид

- **Основные принципы ОС используемые приложениями**
- **базовые компоненты, которые определяют приложение;**
- **файл манифеста, в котором объявляются компоненты и функции устройства, необходимые для приложения;**
- **ресурсы, которые существуют отдельно от кода приложения и позволяют приложению адаптировать свою работу к устройствам с различными конфигурациями**

Основы создания приложений

- Приложения для Android пишутся на языке программирования Java.
- Инструменты Android SDK (Software Development Kit – комплект разработки программного обеспечения) компилируют написанный вами код — и все требуемые файлы данных и ресурсов — в файл APK – *программный пакет Android*, который представляет собой файл архива с расширением .apk.
- В файле APK находится все, что требуется для работы Android-приложения, и он позволяет установить приложение на любом устройстве под управлением системы Android.
- Каждое приложение Android, установленное на устройстве, работает в собственной "песочнице" (изолированной программной среде):

Операционная система Android

Операционная система Android представляет собой многопользовательскую систему Linux, в которой каждое **приложение является отдельным пользователем;**

- по умолчанию система назначает каждому приложению уникальный идентификатор пользователя Linux (этот идентификатор используется только системой и неизвестен приложению);
- система устанавливает полномочия для всех файлов в приложении, с тем чтобы доступ к ним был разрешен только пользователю с идентификатором, назначенным этому приложению;
- у каждого процесса имеется собственная виртуальная машина (ВМ), так что код приложения выполняется изолированно от других приложений;
- по умолчанию каждое приложение выполняется в собственном процессе Linux.
- Android запускает процесс, когда требуется выполнить какой-либо компонент приложения, а затем завершает процесс, когда он больше не нужен либо когда системе требуется освободить память для других приложений.

Принцип предоставления минимальных прав

Android реализует *принцип предоставления минимальных прав*.

- То есть каждое приложение по умолчанию имеет доступ только к тем компонентам, которые ему необходимы для работы, и ни к каким другим.

Однако у приложения есть варианты предоставления своих данных другим приложениям и доступа к системным службам:

1. двум приложениям можно назначить один идентификатор пользователя Linux. В этом случае каждый из них сможет обращаться к файлам другого приложения. Для экономии ресурсов системы также можно сделать так, чтобы приложения с одинаковым идентификатором пользователя выполнялись в одном процессе Linux и использовали одну ВМ (приложения также должны быть подписаны одним сертификатом);
2. приложение может запросить разрешение на доступ к данным устройства, например к контактам пользователя, SMS-сообщениям, подключаемой карте памяти (SD-карте), камере, Bluetooth и др. Все разрешения должны предоставляться приложению при его установке.

Компоненты приложения

Каждый компонент представляет собой отдельную точку, через которую система может войти в приложение.

Не все компоненты являются точками входа для пользователя, а некоторые из них зависят друг от друга.

При этом каждый компонент является самостоятельной структурной единицей и играет определенную роль — каждый из них представляет собой уникальный элемент структуры, который определяет работу приложения в целом.

Компоненты приложения можно отнести к одному из четырех типов. (**Activity, Service, *Content provider, Broadcast* receiver**)

Компоненты каждого типа предназначены для определенной цели, они имеют собственный жизненный цикл, который определяет способ создания и прекращения существования компонента.

Операции (активности)

- Операция (Activity) *представляет* собой один экран с пользовательским интерфейсом.
- **Например,** в приложении для работы с электронной почтой одна операция может служить для отображения списка новых сообщений, другая – для составления сообщения и третья операция – для чтения сообщений. Несмотря на то что операции совместно формируют связное взаимодействие пользователя с приложением по работе с электронной почтой, каждая из них не зависит от других операций.
- Любые из этих операций **могут быть запущены другим приложением** (если это позволяет приложение по работе с электронной почтой). Например, приложение для камеры может запустить операцию в приложении по работе с электронной почтой, которая составляет новое сообщение, чтобы пользователь мог отослать фотографию.
- Операция относится к подклассу класса Activity.

Службы

Служба (Service) *представляет* собой компонент, который работает в фоновом режиме и выполняет длительные операции, связанные с работой удаленных процессов.

Служба не имеет пользовательского интерфейса. Например, она может воспроизводить музыку в фоновом режиме, пока пользователь работает в другом приложении, или же она может получать данные по сети, не блокируя взаимодействие пользователя с операцией.

Служба может быть запущена другим компонентом, который затем будут взаимодействовать с ней, – например операцией.

Служба относится к подклассу класса Service.

Поставщики контента

- Поставщик *контента (Content provider)* управляет общим набором данных приложения. Данные можно хранить в файловой системе, базе данных SQLite, в Интернете или любом другом постоянном месте хранения, к которому у вашего приложения имеется доступ. Посредством поставщика контента другие приложения могут запрашивать или даже изменять данные (если поставщик контента позволяет делать это).
- В Android есть поставщик контента, который управляет информацией контактов пользователя. Любое приложение, получившее соответствующие разрешения, может запросить часть этого поставщика контента (например ContactsContract.Data), для чтения и записи сведений об определенном человеке.
- Поставщики контента также используются для чтения и записи данных, доступ к которым внешним компонентам приложение не предоставляет. Например, в образце приложения Note Pad с помощью поставщика контента выполняется сохранение заметок.
- Поставщик контента относится к подклассу класса ContentProvider. Он должен реализовывать стандартный набор API-интерфейсов, с помощью которых другие приложения будут выполнять транзакции.

Приемники широковещательных сообщений

Приемник широковещательных сообщений (Broadcast receiver) *представляет* собой компонент, который реагирует на объявления распространяемые по всей системе.

- Многие из этих объявлений рассылает система — например объявление о том, что экран выключился, аккумулятор разряжен или был сделан фотоснимок.
- Объявления также могут рассылаться приложениями, — например, чтобы сообщить другим приложениям о том, что какие-то данные были загружены на устройство и теперь готовы для использования.
- Несмотря на то что приемники широковещательных сообщений не имеют пользовательского интерфейса, они могут создавать уведомления в строке состояния, чтобы предупредить пользователя о событии "рассылка объявления".
- Однако чаще всего они являются просто "шлюзом" для других компонентов и предназначены для выполнения минимального объема работы.
- Приемник широковещательных сообщений относится к подклассу класса BroadcastReceiver , а каждое такое сообщение предоставляется как объект Intent

Запуск компонентов другого приложения

- Уникальной особенностью системы Android является то, что любое приложение может запустить компонент другого приложения - вы можете просто запустить операцию фотографирования из приложения для камеры. По завершении этой операции фотография будет возвращена в ваше приложение, и ее можно будет использовать. 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
- В отличие от приложений для большинства других систем, в приложениях для Android отсутствует единая точка входа (например, в них нет функции main()).
- Поскольку система выполняет каждое приложение в отдельном процессе с такими правами доступа к файлам, которые ограничивают доступ в другие приложения, ваше приложение не может напрямую вызвать компонент из другого приложения. Это может сделать сама система Android.
- Поэтому, чтобы вызвать компонент в другом приложении, необходимо сообщить системе о своем намерении (*Intent*) запустить определенный компонент. После этого система активирует для вас этот компонент.

Активация компонентов

- Компоненты трех из четырех возможных типов — операции, службы и приемники широковещательных сообщений — активируются асинхронным сообщением, которое называется ***Intent*** (намерение).
- **Объекты Intent связывают друг с другом** отдельные компоненты во время выполнения, будь то это компоненты вашего или стороннего приложения (эти объекты Intent можно представить себе в виде мессенджеров, которые посылают другим компонентам запрос на выполнение действий).
- **Объект Intent создается** с помощью объекта Intent, который описывает запрос на активацию либо конкретного компонента, либо компонента конкретного *типа* — соответственно, намерение **Intent может быть явным или неявным.**
- Для операций и служб **Объект Intent определяет действие**, которое требуется выполнить (например, просмотреть (view) или отправить (send) что-то), а также **может указывать URI** (Uniform Resource Identifier – унифицированный идентификатор ресурса) данных, с которыми это действие нужно выполнить (помимо прочих сведений, которые нужно знать запускаемому компоненту).

Примеры использования Intent

1. Объект Intent может передавать запрос на выполнение операции "показать изображение" или "открыть веб-страницу".
2. В некоторых ситуациях операцию можно запустить, чтобы получить результат. В этом случае операция возвращает результат также в виде объекта Intent (например, можно отправить сообщение Intent, чтобы дать пользователю возможность выбрать контакт и вернуть его вам — в ответном сообщении Intent будет содержаться URI, указывающий на выбранный контакт).
3. Для приемников широковещательных сообщений Intent просто определяет передаваемое объявление (например, широковещательное сообщение о низком уровне заряда аккумулятора содержит только строку "аккумулятор разряжен").

Активация поставщиков контента

Компоненты четвертого типа – поставщики контента – сообщениями Intent не активируются.

Они активируются по запросу от ContentResolver. Процедура определения контента (content resolver) обрабатывает все прямые транзакции с поставщиком контента, с тем чтобы этого не пришлось делать компоненту, который выполняет транзакции с поставщиком.

Вместо этого он вызывает методы для объекта ContentResolver. Это формирует слой, абстрагирующий (в целях безопасности) поставщика контента от компонента, запрашивающего информацию

Для активации компонентов каждого типа имеются отдельные методы:

1. Можно запустить операцию (или определить для нее какое-то новое действие), передав объект Intent методу startActivity() или startActivityForResult() (если требуется, чтобы операция вернула результат).
2. Можно запустить службу (либо выдать работающей службе новые инструкции), передав объект Intent методу startService(). Либо можно установить привязку к службе, передав объектIntent методу bindService().
3. Можно инициировать рассылку сообщений, передав объект Intent таким методам, как sendBroadcast(), sendOrderedBroadcast() и sendStickyBroadcast().
4. Можно выполнить запрос к поставщику контента, вызвав метод query() для объекта ContentResolver.

Файл манифеста

- Для запуска компонента приложения системе Android необходимо знать, что компонент существует. Для этого она читает файл AndroidManifest.xml приложения (файл манифеста). В этом файле, который должен находиться в корневой папке приложения, должны быть объявлены все компоненты приложения.
- Помимо объявления компонентов приложения, манифест служит и для других целей, среди которых:
- указание всех полномочий пользователя, которые требуются приложению, например разрешения на доступ в Интернет или на чтение контактов пользователя;
- объявление минимального уровня API, требуемого приложению, с учетом того, какие API-интерфейсы оно использует;
- объявление аппаратных и программных функций, которые нужны приложению или используются им, например камеры, службы Bluetooth или сенсорного экрана;
- указание библиотек API, с которыми необходимо связать приложение (отличные от API-интерфейсов платформы Android), например библиотеки Google Maps ;
- и многое другое.

Объявление компонентов в файле манифеста

Основная задача манифеста – это информировать систему о компонентах приложения.

Например, файл манифеста может объявлять операцию следующим образом:

- ```
  <?xml version="1.0" encoding="utf-8"?>
  <manifest ... >
    <application android:icon="@drawable/app_icon.png" ... >
      <activity android:name="com.example.project.ExampleActivity"
              android:label="@string/example_label" ... >
      </activity>
      ...
    </application>
  </manifest>
  ```
- Атрибут android:icon в элементе <application> указывает на ресурсы для значка, который обозначает приложение.
- Атрибут android:name в элементе <activity> указывает полное имя класса подкласса Activity, а атрибут android:label указывает строку, которую необходимо использовать в качестве метки операции, отображаемой для пользователя.
  ```

Все компоненты приложения необходимо объявлять следующим образом:

- элементы <activity> для операций;
- элементы <service> для служб;
- элементы <receiver> для приемников широковещательных сообщений;
- элементы <provider> для поставщиков контента
- Системе не видны операции, службы и поставщики контента, которые имеются в исходном коде, но не объявлены в манифесте, поэтому они не могут быть запущены.
- Приемники широковещательных сообщений можно либо объявить в манифесте, либо создать динамически в коде (как объекты BroadcastReceiver) и зарегистрировать в системе путем вызова registerReceiver().

Объявление возможностей компонентов

С помощью объекта Intent можно запускать операции, службы и приемники широковещательных сообщений.

Для этого в объекте **Intent следует явно указать имя** целевого компонента (с помощью имени класса компонента).

Однако в полной мере возможности объектов Intent раскрываются при использовании концепции **неявных Intent.**

В неявном сообщении **Intent просто описывается тип действия**, которое требуется выполнить (а также, хотя это и не обязательно, данные, в отношении которых вы бы хотели выполнить это действие).

Системе же предоставляется возможности **найти** на устройстве компонент, который может выполнить это действие, и запустить его.

При наличии нескольких компонентов, которые могут выполнить действие, описанное в **сообщении Intent,** пользователь выбирает, какой из них будет

Фильтры объектов Intent

- Система определяет компоненты, которые могут ответить на сообщение Intent**, путем сравнения полученного сообщения Intent** с *фильтрами объектов Intent,* указанными в файле манифеста других приложений, имеющихся на устройстве.
- При объявлении операции в манифесте своего приложения по желанию можно указать фильтры объектов Intent, которые указывают возможности операции, с тем чтобы она могла реагировать на сообщения Intent от других приложений.
- Чтобы объявить фильтр Intent для своего компонента, необходимо добавить элемент <intent-filter> в качестве дочернего для элемента объявления компонента.

Пример фильтра для ответа на сообщение Intent

- Если вы создали приложение для работы с электронной почтой с операцией составления нового сообщения, вы можете объявить фильтр для ответа на сообщения Intent типа "send" (для отправки нового сообщения электронной почты) следующим образом:

```
<manifest ... >
    ...
    <application ... >
      <activity android:name="com.example.project.ComposeEmailActivity">
        <intent-filter>
          <action android:name="android.intent.action.SEND" />
          <data android:type="*/*" />
          <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
      </activity>
    </application>
</manifest>
```

- Затем, если другое приложение создаст объект Intent с действием ACTION_SEND и передаст его в startActivity(), система сможет запустить вашу операцию, дав пользователю возможность написать и отправить сообщение электронной почты.

Объявление требований приложения

Если вашему приложению требуется камера и оно использует API-интерфейсы из Android 2.1 (уровень API 7), эти параметры следует объявить в файле манифеста в качестве требований следующим образом:

```
<manifest ... >
    <uses-feature android:name="android.hardware.camera.any"
                  android:required="true" />
    <uses-sdk android:minSdkVersion="7" android:targetSdkVersion="19" />
    ...
</manifest>
```

Теперь ваше приложение нельзя будет установить из Google Play на устройствах, в которых *нет* камеры, а также на устройствах, работающих под управлением Android версии *ниже* 2.1.

Однако можно также объявить, что приложение использует камеру, но для его работы она не является *непременно необходимой*. В этом случае в приложении атрибуту required необходимо задать значение "false", а во время работы оно должно проверять, имеется ли на устройстве камера, и при необходимости отключать свои функции, которые используют камеру.

Ресурсы приложения

Приложение Android состоит не только из кода — ему необходимы такие существующие отдельно от исходного кода ресурсы, как изображения, аудиофайлы и все, что связано с визуальным представлением приложения.

Для каждого ресурса, включаемого в проект Android, инструменты SDK задают уникальный целочисленный идентификатор, который может использоваться, чтобы сослаться на ресурс из кода приложения или из других ресурсов, определенных в XML.

Например, если в вашем приложении имеется файл изображения с именем logo.png (сохраненный в папке res/drawable/), инструменты SDK сформируют идентификатор ресурса под именем R.drawable.logo, с помощью которого на изображение можно будет ссылаться и вставлять его в пользовательский интерфейс.

Предоставление ресурсов

Один из наиболее важных аспектов предоставления ресурсов отдельно от исходного кода заключается в **возможности использовать альтернативные ресурсы для различных конфигураций устройств**.

Например, определив строки пользовательского интерфейса в XML, вы сможете перевести их на другие языки и сохранить эти переводы в отдельных файлах.

Затем по ***квалификатору* языка** , добавленному к имени каталога ресурса (res/values-fr/ для строк на французском языке), и выбранному пользователем языку система Android применит к вашему пользовательскому интерфейсу строки на соответствующем языке.

Android поддерживает разные ***квалификаторы*** для соответствующих ресурсов.

Квалификатор представляет собой короткую строку, которая включается в имена каталогов ресурсов с целью определения конфигурации устройства, для которой эти ресурсы следует использовать.

В качестве другого примера можно сказать, что для операций следует создавать разные макеты, которые будут соответствовать размеру и ориентации экрана устройства.

Чтобы при изменении ориентации экрана изменялся макет, можно определить два разных макета и применить **соответствующий квалификатор к имени каталога** каждого макета.

После этого система будет автоматически применять соответствующий макет в зависимости от ориентации устройства.

Microsoft Edge
Xbox
Музыка
Кино и ТВ
OneDrive
Камера
Параметры
Карты
Калькулятор
Будильники и часы
Все приложения

Литература

- https://developer.android.com/guide/components/fundamentals.html
- https://habr.com/ru/post/109944/
- https://habr.com/ru/post/336434/