Каждый проект в Android имеет файл манифеста, называемый **AndroidManifest.xml**, который хранится в корневом каталоге. Файл манифеста является важной частью нашего приложения, поскольку он определяет структуру и метаданные приложения. В частности, манифест приложения выполняет следующие задачи:

* Задаёт имя пакета для приложения. Имя пакета служит уникальным идентификатором для приложения.
* Описывает компоненты приложения (активности, службы, широковещательные приёмники и контент-провайдеры), из которых состоит приложение. Он содержит названия классов, которые реализуют каждый из компонентов, и задаёт им различные свойства (например, какие интенты они могут обрабатывать). Эти объявления позволяют Android знать, какие компоненты и при каких условиях могут быть запущены.
* Определяет, в каких процессах будут размещаться компоненты приложения.
* Объявляет, каких разрешения должно иметь приложение для доступа к защищенным частям API и взаимодействия с другими приложениями.
* Перечисляет классы Instrumentation, которые предоставляют профилирование и другую информацию во время работы приложения. Эти объявления присутствуют в манифестве только тогда, когда приложение находится на стадии разработки и отладки, и удаляются перед публикацией.
* Объявляет минимальный уровень API, который требуется приложению.
* Перечисляет библиотеки, с которыми приложение должно быть связано.

**Структура Манифеста**

<?xml version="1.0" encoding="utf-8"?>

<manifest>

<uses-permission />

<permission />

<permission-tree />

<permission-group />

<instrumentation />

<uses-sdk />

<uses-configuration />

<uses-feature />

<supports-screens />

<compatible-screens />

<supports-gl-texture />

<application>

<activity>

<intent-filter>

<action />

<category />

<data />

</intent-filter>

<meta-data />

</activity>

<activity-alias>

<intent-filter> . . . </intent-filter>

<meta-data />

</activity-alias>

<service>

<intent-filter> . . . </intent-filter>

<meta-data/>

</service>

<receiver>

<intent-filter> . . . </intent-filter>

<meta-data />

</receiver>

<provider>

<grant-uri-permission />

<meta-data />

<path-permission />

</provider>

<uses-library />

</application>

</manifest>

**Элемент <uses-permission>**

Этот элемент позволяет запрашивать у системы разрешения, которые нужны приложению для доступа к различным функциям. Это делается с помощью единственного атрибута **android:name**, в параметры которого нужно передать имя разрешения. Например, для получения доступа к Интернету нужно добавить следующее разрешение:

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

**Элемент <permission>**

Объявляет разрешение, которое может использоваться для ограничения доступа к определенным компонентам или функциям этого или других приложений.

**Элемент <permission-tree>**

Этот элемент объявляет базовое имя дерева разрешений. Приложение получает права собственности на все разрешения в дереве. Оно может динамически добавлять новые разрешения для дерева, вызывая **PackageManager.addPermission()**. Имена внутри дерева разделяются точками. Например, если имя дерева задано как**ru.androidtools.project**, то для него можно добавить следующие разрешения:

ru.androidtools.project.TEST

ru.androidtools.project.ANOTHER\_TEST

Обратите внимание, что этот элемент не объявляет сами разрешения, а только пространство имен, в котором эти разрешения будут размещены.

**Элемент <permission-group>**

Объявляет имя группы для набора разрешений. Отдельные разрешения присоединяются к группе через атрибут android:permissionGroup элемента <permission>.

Обратите внимание, что этот элемент не объявляет само разрешение, а только категорию, в которую разрешения будут помещены.

**Элемент <instrumentation>**

Объявляет класс Instrumentation, который позволяет отслеживать взаимодействие приложения с системой.

**Элемент <uses-sdk>**

Позволяет указать совместимость с одной или несколькими версиями Android через указание уровня **API**. Уровень API, указанный в приложении, будет сравниваться с уровнём API на устройстве.

Несмотря на своё имя, этот элемент используется для указания номера API, а не номера версии SDK или платформы Android.

**Элемент <uses-configuration>**

Указывает, какие аппаратные и программные функции требуются приложению. Например, приложение может указать, что ему требуется физическая клавиатура или определенное навигационное устройство, например, трекбол. Этот элемент используется, чтобы избежать установки приложения там, где он не будет работать.

**Элемент <uses-feature>**

Объявляет определённый функционал, используемый приложением. Цель объявления этого элемента — информировать любой внешний объект о наборе аппаратных и программных функций, от которых зависит приложение. Элемент предлагает атрибут **android:required**, который указывает, требуется ли приложению функционал, без которого он не сможет работать, либо приложение лишь предпочитает этот функционал, но может обойтись без него.

Вы должны указать каждую функцию в отдельном элементе, поэтому если приложению нужно несколько функций, нужно будет создать несколько элементов <uses-feature>. Например, приложение, которое требует Bluetooth и камеру, должно иметь следующие строки в манифестве:

<uses-feature android:name="android.hardware.bluetooth" />

<uses-feature android:name="android.hardware.camera" />

**Элемент <support-screens>**

Позволяет указать размеры экрана, поддерживаемые вашим приложением, и включить режим совместимости с экраном, большим, чем приложение поддерживает.

**Элемент <compatible-screens>**

Указывает конфигурацию экрана, с которой приложение совместимо. В манифесте разрешен только один экземпляр элемента**<compatible-screen>**, но он может содержат несколько элементов **<screen>**. Каждый элемент <screen> указывает конкретную комбинацию плотности экрана, с которой приложение совместимо.

**Элемент <supports-gl-texture>**

Объявляет одну GL текстуру сжатия, поддерживаемую приложением.

**Элемент <application>**

Является одним из основных элементов. В нём описываются метаданные приложения, такие как иконка, название, стиль и прочее. Включает в себя подэлементы, которые описывают компоненты в приложении.

В манифесте может быть только один элемент **<application>**.

**Элемент <activity>**

Объявляется активность (подкласс **Activity**), которая реализует часть визуального пользовательского интерфейса приложения. Все активности должны иметь элемент <activity> в манифесте. Если активность не будет иметь соответствующего элемента, она не будет видна системе и никогда не будет запущена.

**Элемент <intent-filter>**

Указывает типы интентов, на которые может реагировать активность, сервис или широковещательный приемник. Фильтр интентов объявляет возможности своего родительского компонента — что может делать активность или сервис и какие типы широковещательных сообщений может получать приемник. Он открывает интентам нужного типа доступ к компоненту и отфильтровывает те, которые не нужны.

**Элемент <action>**

Добавляет действие в фильтр намерений. Элемент **<intent-filter>** должен содержать как минимум один элемент **<action>**. Если он не содержит никаких действий, интенты не пройдут через фильтр.

**Элемент <category>**

Добавляет название категории в фильтр намерений.

**Элемент <data>**

Добавляет спецификацию данных в фильтр намерений. Спецификация может быть только типом данных (атрибут **android:mimeType**), только URI, либо типом данных вместе с URI. URI определяется отдельными атрибутами для каждой из его частей:

<scheme>://<host>:<port>/[<path>|<pathPrefix>|<pathPattern>]

Эти атрибуты являются необязательными, но взаимозависимыми.

Если для фильтра не указана схема, все остальные атрибуты URI игнорируются.

Если для фильтра не указан **<host>**, атрибут **<port>** и все пути игнорируются.

Все элементы <data>, содержащиеся в одном**<intent-filter>**, относятся к этому фильтру. Например, следующий фильтр

<intent-filter . . . >

<data android:scheme="something" android:host="project.example.com" />

. . .

</intent-filter>

аналогичен этому

<intent-filter . . . >

<data android:scheme="something" />

<data android:host="project.example.com" />

. . .

</intent-filter>

Вы можете поместить любое количество элементов **<data>** внутрь фильтра, чтобы предоставить ему несколько данных. Ни один из атрибутов не имеет значения по умолчанию.

**Элемент <meta-data>**

Пара «**имя-значение**» для дополнительных произвольных данных, которые могут быть переданы родительскому компоненту. Элемент компонента может содержать любое количество элементов **<meta-data>**. Значения из всех них собираются в отдельном объекте Bundle и становятся доступными для компонента в качестве поля **PackageItemInfo.metaData**.

**Элемент <activity-alias>**

Псевдоним для активности, указанной в атрибуте **android:targetActivity**. Активность должна быть в том же приложении, что и псевдоним, и должна быть объявлена в манифесте перед псевдонимом.

Псевдоним представляет целевую активность как независимый объект. Он может иметь свой собственный набор фильтров интентов, и они, а не фильтры интентов целевой активности, определяют, какие интенты могут запустить активность через псевдоним и как система обрабатывает псевдоним.

Например, фильтр интентов в псевдониме может содержать  флаги **«android.intent.action.MAIN»**и **«android.intent.category.LAUNCHER»**, заставляя его отображаться в списке приложений, хотя ни один из фильтров целевой активности не устанавливал эти флаги.

**Элемент <service>**

Объявляет сервис (подкласс **Service**) как один из компонентов приложения. В отличие от активности, сервис не имеет визуального пользовательского интерфейса. Сервисы используются для реализации длительных фоновых операций или крупного API, который может быть вызван другими приложениями.

Все сервисы должны быть представлены элементами**<service>** в файле манифеста. Те сервисы, которые не будут объявлены, не будут видны системе и не могут быть запущены.

**Элемент <receiver>**

Объявляет широковещательный приемник (подкласс **BroadcastReceiver**) как один из компонентов приложения. Широковещательные приемники позволяют приложению получать интенты, которые посылаются системой или другими приложениями, даже если другие компоненты приложения не работают.

Существует два способа создать широковещательный приемник:

1. Объявить его в коде манифеста с помощью этого элемента.
2. Создать динамически в коде и зарегистрировать приемник с помощью метода Context.registerReceiver().

**Элемент <provider>**

Объявляет компонент контент-провайдера. Контент-провайдер — это подкласс **ContentProvider**, который обеспечивает структурированный доступ к данным, управляемым приложением. Все контент-провайдеры должны быть определены в элементе **<provider>**, в противном случае система не будет знать о них.

**Элемент <grant-uri-permission>**

Указывает, для кого могут быть предоставлены разрешения на подмножество данных контент-провайдера.

**Элемент <path-permission>**

Определяет путь и необходимые разрешения для определенного подмножества данных внутри контент-провайдера. Этот элемент может быть указан несколько раз для передачи нескольких путей.

**Элемент <uses-library>**

Задает общедоступную библиотеку, с которой приложение должно быть связано. Этот элемент указывает системе включить код библиотеки в загрузчик классов для пакета.