

Permisos en Android



Permisos (I)

- ▶ Android **protege los recursos y los datos con permisos.**
- ▶ Son usados para limitar el acceso a:
 - ❖ **Información de usuario** → por ej. *CONTACTOS*
 - ❖ **Operaciones** que impliquen un coste económico → por ej. *SMS/MMS.*
 - ❖ **Recursos del sistema** → Por ej. *Cámara de fotos*
- ▶ Si una **aplicación Android no tiene los permisos asociados con ella,**
 - *no puede hacer nada que tenga efectos adversos sobre el sistema u otras aplicaciones.*

Permisos (II)

- ▶ Los permisos son representados por **STRINGS**
- ▶ Deben ser declarados en el MANIFEST, tanto:
 - ❖ Los permisos que necesita nuestra aplicación,
 - ❖ Como los permisos requeridos a otras apps para usar componentes de la nuestra
- ▶ , **CUALQUIER APLICACIÓN** puede también definir y aplicar sus propios permisos

Permisos (III)

► Permisos que debemos controlar:

- ✂ que nuestras **aplicaciones tengan los permisos adecuados para hacer lo que queramos** con la información de otras aplicaciones: **PEDIR PERMISOS.**
- ✂ que **otras aplicaciones** que, en determinado momento necesiten **consultar información o servicios de nuestra aplicación, puedan o no, según decidamos nosotros:** **DEFINIR PERMISOS.**

Permisos: **CONCESIÓN ANTES DE LA VERSIÓN 6.0 ANDROID** (IV)

- ▶ El usuario **concede los permisos en la instalación**
- ▶ Si no está de acuerdo con algún permiso la única alternativa es no instalar la app.
- ▶ Una vez instalada, la app puede realizar acciones asociadas a estos permisos tantas veces como desee.
- ▶ Algunas apps **abusan** de esta circunstancia **pidiendo permisos innecesarios**.
- ▶ El usuario no puede hacer nada si desea instalar la app.

Permisos: NUEVO PLANTEAMIENTO CON LA VERSIÓN 6.0 ANDROID (V)

▶ Permisos normales:

- Se otorgan en la instalación

▶ Permisos peligrosos:

- Se agrupan en 9 categorías:
 - *Almacenamiento, localización, teléfono, SMS, contactos, calendario, cámara, micrófono y sensor de ritmo cardíaco.*
- Tras la instalación no se concede ninguno de esos permisos.
- Cuando la app tenga que realizar una acción que requiera uno de estos permisos, **ha de solicitarlo al usuario.**
- Aunque se conceda el permiso, **el usuario podrá revocarlo** con posterioridad.
- Si no se concede algún permiso **la app ha de intentar seguir funcionando.**

Permisos: LA APP SOLICITA LOS PERMISION EN VERSIÓN 6.0 ANDROID (VI)

- ▶ En el momento de realizar una acción que necesita un permiso peligroso, la app ha de **justificar por qué necesita el permiso (opcional)**:



Sin permiso para administrar llamadas no puedo borrar llamadas del registro. OK

- ▶ Luego, ha de **solicitarlo**:



Permisos: PERMISOS CONFIGURABLES EN VERSIÓN 6.0 ANDROID (VII)

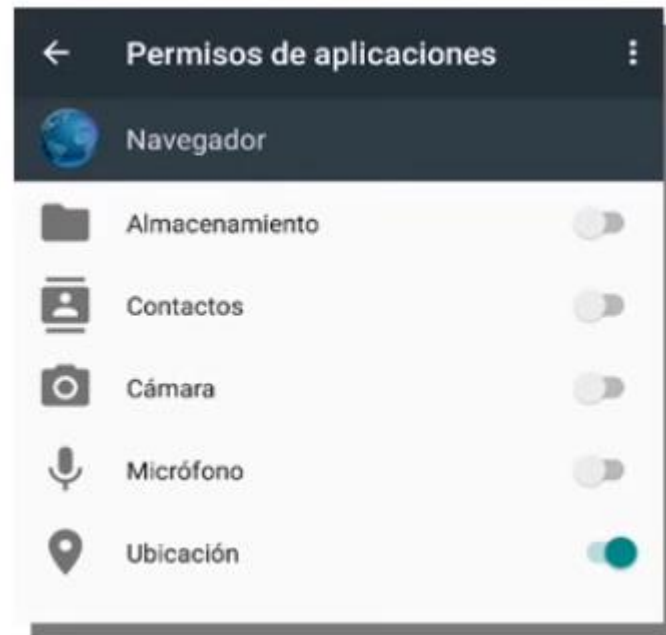
- Los permisos **pueden ser concedidos o retirados desde el administrador de aplicaciones:**



- Sólo se muestran los permisos que la app lista en el manifiesto.

Permisos: LOS PERMISOS SE CONCEDEN POR GRUPOS EN VERSIÓN 6.0 ANDROID (VIII)

- ▶ El usuario **concede o rechaza los permisos por grupos**.
- ▶ Si en el manifest se ha pedido **leer y escribir en la SD**, concedemos los dos permisos (*permiso Almacenamiento*) o ninguno.
- ▶ Es decir, no podemos conceder permiso de lectura pero denegar el de escritura.



Permisos: **Qué permisos necesita mi app** (IX)

- ▶ Mi app necesitará permisos siempre que:
 - Use **información o recursos que no cree la propia app.**
 - Realice **acciones que afecten al comportamiento del dispositivo u otras apps.**
- ▶ ***Ejemplos de casos que necesita permisos : si necesita...***
 - *acceder a Internet*
 - *Usar la cámara.*
 - *Encender el WI-FI o apagarlo.*

Permisos: ¿Cómo realizar tareas que requieran permisos? (X)

- ▶ Cuando en nuestra app necesitemos realizar tareas que **requieran permisos** podemos llevarlas a cabo de **DOS FORMAS DISTINTAS**:

1. Haciendo que nuestra APP realice todas las tareas y/o proporcione la información.

❑ En este caso, NECESITAMOS PEDIR PERMISOS

2. Haciendo que nuestra APP pida (*mediante el uso de un **INTENT***) que otra APP distinta realice todas las tareas y/o proporcione la información.

❑ En este caso, **NO NECESITAMOS PEDIR PERMISOS**

Permisos: ¿Cómo realizar tareas que requieran permisos? (XI)

- ▶ **EJEMPLO:** Supongamos que nuestra APP necesita leer la libreta de direcciones del usuario (*tarea que requiere permisos*).

1. **FORMA 1:** Si nuestra APP realiza todas las tareas

- ❑ Necesitaremos el permiso, READ_CONTACTS

2. **FORMA 2:** Si nuestra APP usa un INTENT para pedir los datos a la APP CONTACTOS del teléfono

- ❑ En este caso, **NO NECESITAMOS PEDIR PERMISOS**, pero **SI la app CONTACTOS DEL TFNO.**

Permisos: ¿Cómo realizar tareas que requieran permisos? (XII)

- ▶ **OTRO EJEMPLO:** Supongamos que nuestra APP necesita tomar fotos con la cámara (*tarea que requiere permisos*).

1. **FORMA 1:** Si nuestra APP va a controlar ella la CÁMARA directamente

- ❑ Necesitaremos el permiso, CAMERA
- ❑ VENTAJAS:
 - ✓ Tenemos total control sobre el proceso de tomar la foto.
 - ✓ Acceso total a la API de la cámara.
- ❑ DESVENTAJA:
 - ✓ Tenemos que diseñar nuestra propia INTERFAZ DE USUARIO en nuestra APP para tomar y recoger la foto.

2. **FORMA 2:** Si nuestra APP no necesita control completo, puede usar un INTENT ACTION_IMAGE_CAPTURE para pedir una imagen.

- ❑ En este caso, NO NECESITAMOS PEDIR PERMISOS, y ANDROID nos mostrará APPS con las que poder tomar la foto para elegir una.
- ❑ VENTAJAS:
 - ✓ No hay que diseñar el INTERFAZ DE USUARIO.

Permisos: sólo pedir los permisos que necesitamos (XIII)

- ▶ Cada vez que preguntamos por un permiso, el usuario debe **tomar una decisión**.
- ▶ Debemos minimizar el nº de veces que hacemos esas peticiones.
- ▶ En Android 6.0 (API 23) o superior.
 - Cada vez que el usuario intenta alguna nueva acción de la APP, que requiera permiso,
 - La APP tiene que interrumpir el trabajo del usuario y mostrarle el mensaje **pidiéndole permiso**. **NO AGOBIES AL USUARIO**
- ▶ En versiones más viejas de Android.
 - El usuario tiene que CONCEDER cada uno de los permisos de la APP cuando instala la APP.
 - Si la lista es muy larga o parece no adecuada, el usuario puede decidir NO INSTALAR NUESTRA APP.
- ▶ Muchas veces, **PODEMOS EVITAR PEDIR PERMISOS** usando INTENTS en su lugar para que otras APP nos hagan ese trabajo, que NO ES PRIMORDIAL en nuestra APP.

PEDIR PERMISOS EN EL MANIFEST

Permisos: ¿CÓMO AÑADIR LOS PERMISOS EN EL MANIFEST? (I)

- ▶ Usar la etiqueta **<uses-permission>** en el *AndroidManifest.xml*.
- ▶ Podemos tener **cero** o muchos elementos **<uses-permission>**.
- ▶ Habrá uno por cada permiso que se necesite.
- ▶ Elementos **<uses-permission>** se definen todos como **hijos directos del elemento raíz del archivo: etiqueta <manifest>**.
- ▶ *Ej:*

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.snazzyapp">

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

    <application ...>
        ...
    </application>

</manifest>
```


Permisos: **<uses-permission>** (II)

► ATRIBUTOS de <uses-permission> :

⊙ *android:name:*

Define el nombre del permiso que requiere nuestra aplicación.

Puede ser :

- un **permiso definido por la aplicación** con **<permission>**,
- un **permiso definido por otra aplicación**, o
- uno de los **permisos estándar de Android**, como:

android.permission.CAMERA

android.permission.READ_CONTACTS

- Las aplicaciones de terceros **pueden tener sus propios permisos**, para los casos en los que **deberás consultar su documentación para poder utilizarlos** dentro de tus aplicaciones Android.

Permisos (III)

► Ejemplo del MANIFEST de una app que pide usar permisos:

```
<manifest ... >
...
<uses-permission android:name=
    "android.permission.CAMERA"/>
<uses-permission android:name=
    "android.permission.INTERNET"/>
<uses-permission android:name=
    "android.permission.ACCESS_FINE_LOCATION"/>
...
</manifest >
```

Permiso para acceder a la cámara

Permiso para acceder a información precisa de localización (GPS)

Permiso para abrir un socket para acceder a INTERNET

PEDIR PERMISOS EN TIEMPO DE EJECUCIÓN Android 6.0

EL PROBLEMA

- ▶ En Android 6.0 un usuario puede denegar alguno de los permisos considerados como peligrosos.
- ▶ **Antes de realizar una acción peligrosa has de verificar si tienes el permiso.**
- ▶ Si no lo hacemos, Android Studio muestra un error directamente en el código de nuestra app:

```
getContentResolver().delete(CallLog.Calls.CONTENT_URI,  
                             "number='55555555'", null);
```

Call requires permission which may be rejected by user: code should explicitly check to see if permission is available (with 'checkPermission') or explicitly handle a potential 'SecurityException' [more...](#) (Ctrl+F1)

VERIFICAR SI TENEMOS PERMISO

```
if (ContextCompat.checkSelfPermission(this,  
    Manifest.permission.CAMERA) == PackageManager.PERMISSION_GRANTED) {  
    //permiso concedido  
    lanzarIntentCamara();  
  
} else {  
    //permiso no concedido  
    solicitarPermisoCamara();  
}
```

SOLICITAR EL PERMISO

```
// Mostrar una explicación de por qué necesito el permiso???  
if (ActivityCompat.shouldShowRequestPermissionRationale(this,  
    Manifest.permission.CAMERA)) {  
    mostrarDialogo(this, " Sin el permiso de la cámara no se pueden tomar fotos",  
        Manifest.permission.CAMERA);  
}  
else {  
    ActivityCompat.requestPermissions(this,  
        new String[]{Manifest.permission.CAMERA},  
        MY_PERMISSIONS_REQUEST_CAMERA);  
}
```

Solicitud de permiso

Sin el permiso de la cámara no se
pueden tomar fotos

ACEPTAR



Allow 07-3-
AndroidManifest-Ge...
to take pictures and
record video?



Never ask again

DENY

ALLOW

PROCESAR LA RESPUESTA DEL USUARIO

```
@Override
public void onRequestPermissionsResult(int requestCode,
                                     String permissions[], int[] grantResults) {
    switch (requestCode) {
        case MY_PERMISSIONS_REQUEST_CAMERA: {
            // If request is cancelled, the result arrays are empty.
            if (grantResults.length > 0
                && grantResults[0] == PackageManager.PERMISSION_GRANTED) {

                //REALIZAMOS LA ACCIÓN

            } else {
                //a) Seguimos con el proceso sin esta acción
                //b) Abortamos el proceso actual
                //c) Salimos de la app
            }
            return;
        }
    }
}
```


DEFINIR NUESTROS PROPIOS PERMISOS

Permisos: DEFINIR PERMISOS(I)

- ▶ Para **definir un permiso propio**, *por ejemplo, para controlar quién puede arrancar nuestras activities*, utilizamos el elemento ***<permission>***.
- ▶ Elementos ***<permission>*** se definen todos como **hijos directos del elemento raíz del archivo**: *etiqueta <manifest>*.
- ▶ *Enlace a un ejemplo muy claro de uso.*

Protegiendo el acceso a nuestras aplicaciones :

definir permisos con **<permission>** (II)

► Los atributos para definir un permiso personalizado son:

Android:name:

- **Nombre simbólico del permiso.**
- Debe seguir un esquema **<paquete de nuestra app>.<nombre del permiso>.**
- Para evitar que nuestro permiso coincida con el de alguna otra aplicación, se recomienda utilizar el espacio de nombres de las clases Java de nuestra aplicación. Por ejemplo: *com.ejemplosdam.*

Android:permissionGroup:

- **Asigna este permiso a un grupo. Ej: permisos con coste,...**
- El valor de este atributo es el **nombre del grupo.**
- Si este atributo no está establecido, el permiso no pertenece a un grupo.
- Es opcional.

Android:label:

- **Un nombre corto para el permiso**, que sea entendible por el usuario.
- Extraída del fichero de recursos cadena “strings.xml”.

Protegiendo el acceso a nuestras aplicaciones :

definir permisos con `<permission>` (III)

Android:description:

- Descripción larga del permiso que sea entendible por el usuario,.
- Extraída del fichero de recursos cadena “strings.xml”.
- Es posible que aparezca **para explicar el permiso para el usuario** – por ejemplo, *cuando se le pregunta al usuario si desea conceder el permiso a otra aplicación.*

Android:icon:


- **Icono que representa al permiso.**
- Referencia a un drawable o mipmap recurso.

Android:protectionLevel:

- Indica la “**peligrosidad**” del permiso, y como debe reaccionar el sistema ante una solicitud de este permiso.
- Cambiará la forma de notificar este permiso al usuario en la interfaz, según su peligrosidad.
- Puede valer: “normal” | “dangerous” | “signature” | “signatureOrSystem

Protegiendo el acceso a nuestras aplicaciones : *definir permisos con* <permission> (IV)

- ▶ Ejemplo de aplicación que define un permiso y cómo usarlo luego



DEFINING PERMISSIONS

SUPPOSE YOUR APPLICATION PERFORMS A PRIVILEGED/DANGEROUS OPERATION

YOU MIGHT NOT WANT TO ALLOW JUST ANY APPLICATION TO INVOKE YOURS

SO YOU CAN DEFINE & ENFORCE YOUR OWN PERMISSION

PERMISSIONS

EXIGIR PERMISOS PARA ACCEDER A COMPONENTES CONCRETOS:

- Activity
- Service

- ...

Permisos para los componentes

- ▶ Android permite que **los propios componentes requieran permisos para acceder a ellos, para protegerse.**
- ▶ Estos permisos tienen prioridad sobre los permisos a nivel de APLICACIÓN.
- ▶ Puede utilizar cualquier permiso definido por Android o puede definir uno propio.
- ▶ Para hacer esto, se incluye el atributo **android:permission** en el componente deseado,
 - Indicando como valor el **nombre del permiso que controla el acceso** a ese componente.
- ▶ Ejemplo de uso:

```
<activity
    android:name=".SecureApp"
    android:label="Secure App"
    android:permission="com.androideity.SEE_APP">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category
            android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```


Permisos para los componentes

Activity Permissions

- ▶ Se aplican sobre la etiqueta `<activity>`
- ▶ Restringe qué componentes pueden arrancar esa ACTIVITY asociada.
- ▶ Ese permiso es chequeados cuando se ejecutan los métodos:
 - `**startActivity()`
 - `**startActivityForResult()`
- ▶ Si el que llama a esa activity no tiene el **permiso requerido**, **Android lanza** una **SecurityException** para indicar que falla el permiso.

Permisos para los componentes

Services Permissions

- ▶ Se aplican sobre la etiqueta `<service>`
- ▶ Restringe qué componente puede arrancar o conectarse al SERVICIO asociado.
- ▶ Eses permiso es chequeado cuando se ejecutan los métodos:
 - `**startService()`
 - `**stopService()`
 - `**bindService()`
- ▶ Si el que llama a ese servicio no tiene el **permiso requerido**, **Android lanza** una **SecurityException** para indicar que falla el permiso.

Permisos para los componentes BroadcastReceiver Permissions

- ▶ Se aplican sobre la etiqueta `<receiver>`
- ▶ Restringe qué componentes pueden enviar y recibir broadcasts.
- ▶ Esos permisos son comprobados de diferentes formas y en diferentes sitios.
- ▶ Los veremos más adelante cuando veamos a fondo los BroadcastReceiver.

Permisos para los componentes ContentProvider Permissions

- ▶ Se aplican sobre la etiqueta `<provider>`
- ▶ Restringe qué componentes pueden leer y escribir los datos en un ContentProvider.
- ▶ Los ContentProvider tienen también otro tipo de permisos llamados **URI PERMISOS**
- ▶ Los veremos más adelante cuando veamos a fondo los ContentProvider.

OTROS PERMISOS

Otros permisos

- ▶ El sistema Android requiere que **todas las aplicaciones instaladas sean firmadas digitalmente con un certificado** con un clave privada que es proporcionada por el desarrollador de la aplicación.
- ▶ El sistema Android utiliza el certificado como una forma de **identificar al autor de una aplicación** y establecer relaciones con confianza entre las aplicaciones.
- ▶ El **certificado no se utiliza para controlar qué aplicaciones puede instalar** el usuario.
- ▶ El **certificado no necesita estar firmado por una autoridad certificadora**: se permite , y es típico, que **las apps Android utilicen certificados auto-firmados**.

Otros permisos

- ▶ Los puntos principales para entender sobre firmado de aplicaciones Android son:
 - Todas las aplicaciones deben ser firmadas.
 - El sistema no instalará una aplicación en un emulador o en un dispositivo si no está firmada.
 - Para probar y depurar tu app, las herramientas de compilación firman tu app con una **debug key** especial que es **creada por el Android SDK build tools**.
 - Cuando estás listo para publicar tu app para usuarios finales, debes firmarla con un clave privada conforme. **No puedes publicar una app que está firmada con la debug key de las SDK tools**.
 - **Puedes utilizar certificados auto-firmados** para firmar tus aplicaciones. No se necesita ninguna autoridad certificadora.
 - El sistema comprueba la fecha de caducidad del certificado solo en tiempo de instalación. **Si el certificado caduca después de haber sido instalada la app, la app seguirá funcionando normalmente.**