Permisos en android

Contenido

Nota: **los proyectos de mapas y en general todos los proyectos en este tema** probarlos en **el emulador en Pixel 2 Api 30**. Cread un emulador con estas características

1 Introducción

Vamos a **introducir en estos apuntes los permisos en Android a partir de versión 6 MarshMallow, versión 24 de la SDK.** No lo encontrareis **indicado en los apuntes del curso pero es muy importante, porque a partir de la versión 6**, los permisos **han cambiado,y hace falta una solicitud explícita de cada permiso** que usen nuestras aplicaciones. Vamos a introducirlo sobre uno de los **proyectos del tema 3**, vosotros tendreís que hacerlo sobre el resto de proyectos que usen permisos. **Vamos a modificar el proyecto ServidorBluetooth2020** para que pida **permisos de uso de Bluetooth** y vamos a **explicar el nuevo código.**

1.1 Permisos

Primero vamos a ver como gestiona **Android los permisos a partir de versión 6.** Al final introduciremos **brevemente los cambios en la versión 9 y 10.**

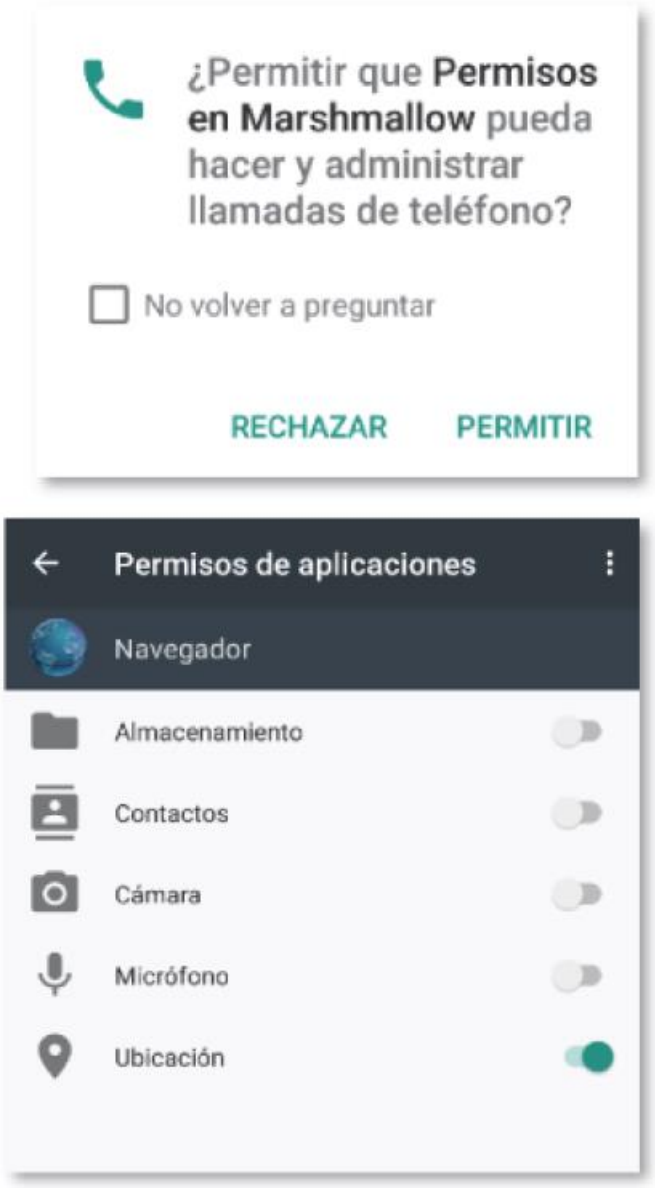
1.1.1 Android 6 en adelante. Grupos de permisos.

En la **versión 6 se introducen importantes novedades a la hora de conceder los permisos a las aplicaciones.** En primer lugar**, los permisos son divididos en normales y peligrosos**. A su vez los **permisos peligrosos se dividen en 9 grupos: almacenamiento, localización, teléfono, SMS, contactos, calendario, cámara, micrófono y sensor de ritmo cardíaco.**

En el **proceso de instalación el usuario da el visto bueno a los permisos normales**, de la **misma forma como se hacía en la versión anterior**. Por el contrario, **los permisos peligrosos no son concedidos en la instalación.** La **aplicación consultará al usuario si quiere conceder un permiso peligroso en el momento** de utilizarlo:

Además, **se recomienda que la aplicación indique para qué lo necesita**. De esta forma **el usuario tendrá más elementos de juicio para decidir si da o no el permiso**. Si el **usuario no concede el permiso la aplicación ha de tratar de continuar el proceso sin este permiso**. Otro **aspecto interesante es que el usuario podrá configurar en cualquier momento qué permisos concede y cuáles no**. Por ejemplo, **podemos ir al administrador de aplicaciones y seleccionar la aplicación Navegador.** En el apartado Permisos **se nos mostrará los grupos de permisos que podemos concede.**

NOTA: Existen otros permisos que no han sido incluidos en esta lista dado que no podemos solicitarlos en nuestras aplicaciones al estar reservados para aplicaciones del sistema



Observa **como de los grupos de permisos solicitados**, en este momento solo **concedemos el permiso de Ubicación.**

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

1.1.2 Permisos peligrosos

Debemos identificar los permisos definidos como “peligrosos” porque como veremos más adelante los permisos considerados como “peligrosos” obligan al desarrollador a pedirlos explícitamente en tiempo de ejecución además de declararlos en el manifest.

LISTA DE PERMISOS PELIGROSOS:

Almacenamiento Externo:

WRITE_EXTERNAL_STORAGE– Modificar/eliminar almacenamiento USB (API 4). Permite el borrado y la modificación de archivos en la memoria externa. Lo ha de solicitar toda aplicación que necesite escribir un fichero en la memoria externa; por ejemplo, exportar datos en XML. Pero al permitirlo también podrán modificar/eliminar ficheros externos creados por otras aplicaciones.

READ_EXTERNAL_STORAGE– Leer almacenamiento USB (API 16). Permite leer archivos en la memoria externa. Este permiso se introdujo en la versión 4.1. En versiones anteriores todas las aplicaciones pueden leer en la memoria externa. Por lo tanto, has de tener cuidado con la información que dejas en ella.

Ubicación:

ACCESS_COARSE_LOCATION –Localización no detallada (basada en red). Localización basada en telefonía móvil (Cell-ID) y Wi-Fi. Aunque en la actualidad esta tecnología suele ofrecernos menos precisión que el GPS, no siempre es así. Por ejemplo, se está aplicando en el interior de aeropuertos y museos con precisiones similares.

ACCESS_FINE_LOCATION –Localización GPS detallada. Localización basada en satélites GPS. Al dar este permiso también estamos permitiendo la localización basada en telefonía móvil y Wi-Fi (ACCESS_COARSE_LOCATION).

Teléfono:

CALL_PHONE –Llamar a números de teléfono directamente Servicios por los que tienes que pagar. Permite realizar llamadas sin la intervención del usuario. Nunca solicites este permiso en tus aplicaciones, muchos usuarios no instalarán tu aplicación. Si has de realizar una llamada, es mejor realizarla por medio de una intención. A diferencia de la llamada directa, no necesitas ningún permiso, dado que el usuario ha de pulsar el botón de llamada para que comience.

READ_PHONE_STATE –Consultar identidady estado del teléfono. Muchas aplicaciones, como los juegos, piden este permiso para ponerse en pausa cuando recibes una llamada. Sin embargo, también permite el acceso al número de teléfono, IMEI (identificador de teléfono GSM), IMSI (identificador de tarjeta SIM) y al identificador único de 64 bits que Google asigna a cada terminal. Incluso si hay una llamada activa, podemos conocer el número al que se conecta la llamada.

READ_PHONE_NUMBERS –Leer los números almacenados en el dispositivo. Está incluido en las capacidades del permiso READ_PHONE_STATE pero se permite su utiliación independiente en las Instant Apps.

READ_CALL_LOG y WRITE_CALL_LOG –Leer y modificar el registro de llamadas telefónicas. Como realizar estas acciones se describe al final del capítulo 9.

ADD_VOICEMAIL –Añadir mensajes de voz. Permite crear nuevos mensajes de voz en el sistema.

USE_SIP –Usar Session Initial Protocol. (API 9). Permite a tu aplicación usar el protocolo SIP.

PROCESS_OUTGOING_CALLS –Procesar llamadas salientes. Permite a la aplicación controlar, modificar o abortar las llamadas salientes.

ANSWER_PHONE_CALLS –Contestar llamadas entrantes.

Mensajes de texto (SMS):

SEND_SMS –Enviar mensaje SMS Servicios por los que tienes que pagar. Permite la aplicación mandar de texto SMS sin la validación del usuario. Por iguales razones que **CALL_PHONE**, a no ser que tu aplicación tenga que mandar SMS sin la intervención del usuario, resulta más conveniente enviarlos por medio de una intención.

RECEIVE_SMS –Recibir mensajes de texto. Permite a la aplicación recibir y procesar SMS. Una aplicación puede modificar o borrar los mensajes recibidos

READ_SMS –Leer mensajes de texto. Permite a la aplicación leer los mensajes SMS entrentes.

RECEIVE_MMS – Recibir mensajes MMS. Permite monitorizar los mensajes multimedia entrantes, pudiendo acceder a su contenido.

RECEIVE_WAP_PUSCH – Recibir mensajes WAP Push. Permite monitorizar los mensajes WAP Push entrantes. Un mensaje WAP PUSH es un tipo de SMS que se usa para acceder de manera sencilla a una página WAP en lugar de teclear su direccion URL en el navegador.

Contactos:

READ_CONTACTS – Leer datos de contactos. Permite leer información sobre los contactos almacenados (nombres, correos electrónicos, números de teléfono). Algunas aplicaciones podrían utilizar esta información de forma no lícita

WRITE_CONTACTS – Escribir datos de contactos. Permite modificar los contactos.

GET_ACCOUNTS – Obtener Cuentas. Permiten acceder a la lista de cuentas en el Servicio de Cuentas.

Calendario:

READ_CALENDAR – Leer datos de contactos. Permite leer información del calendario del usuario.

WRITE_CONTACTS – Escribir datos de contactos. Permite escribir en el calendario, pero no leerlo.

Camara:

CAMARA – Hacer fotos / grabar vídeos. Permite acceso al control de la cámara y a la toma de imágenes y vídeos. El usuario puede no ser consciente.

Microfono:

RECORD_AUDIO – Grabar audio. Permite acceso grabar sonido desde el micrófono del teléfono.

Sensores corporales :

BODY SENSORS_ – Leer sensores corporales. Da acceso a los datos de los sensores que están monitorizando el cuerpo del usuario. Por ejemplo, el lector de ritmo cardiaco.

LISTA PERMISOS NORMALES:

Comunicaciones:

INTERNET_– Acceso a Internet sin límites. Permite establecer conexiones a través de Internet. Este es un permiso muy importante, en el que hay que fijarse a quién se otorga.La mayoría de las aplicaciones lo piden, pero no todas lo necesitan. Cualquier malware necesita una conexión para poder enviar datos de nuestro dispositivo.

ACCESS_NETWORK_STATE –Ver estado de red. Información sobre todas las redes. Por ejemplo para saber si tenemos conexión a internet.

CHANGE_NETWORK_STATE – Cambiar estado de red. Permite cambiar el estado de conectividad de redes.

NFC – Near field communication. (API 19) Algunos dispositivos disponen de un trasmisor infrarrojo para el control remoto de electrodomésticos.

TRASMIT_R – Trasmitir por infrarrojos. (API 19) Algunos dispositivos disponen de un trasmisor infrarrojo para el control remoto de electrodomésticos.

Conexión WIFI:

ACCESS_WIFI_STATE – Ver estado de Wi-Fi. Permite conocer las redes Wi-Fi disponibles.

CHANGE_WIFI_STATE – Cambiar estado de Wi-Fi. Permite cambiar el estado de conectividad Wi-Fi.

CHANGE_WIFI_MULTICAST_STATE – Cambiar estado multicast Wi-Fi (API 4). Permite pasar al modo Wi-Fi Multicast.

Bluetooth:

BLUETOOTH – Crear conexión Bluetooth. Permite a una aplicación conectarse con otro dispositivo Bluetooth. Antes ambos dispositivos han de emparejarse

BLUETOOTH_ADMIN – Emparejar Bluetooth. Permite descubrir y emparejarse con otros dispositivos Bluetooth.

Consumo de batería:

WAKE_LOCK –Impedir que el teléfono entre en modo de suspensión. Para algunas aplicaciones, como un navegador GPS, puede ser importante que no sean suspendidas nuca. Realmente, a lo único que puede afectar es a nuestra batería.

FLASHLIGHT – Linterna. Permite encender el flash de la cámara.

VIBRATE – Control de la vibración. Permite hacer vibrar al teléfono. Los juegos suelen utilizarlo.

Aplicaciones:

RECEIVE_BOOT_COMPLETED – Ejecución automática al encender el teléfono. Permite a una aplicación recibir el anuncio broadcast ACTION_BOOT_COMPLETED enviado cuando el sistema finaliza un inicio. Gracias a esto la aplicación pondrá ponerse en ejecución al arrancar el teléfono.

BROADCAST_STICKY – Enviar anuncios broadcast permanentes. Un broadcast permanente llegará a los receptores de anuncios que actualmente estén escuchando, pero también a los que se instancien en un futuro. Por ejemplo, el sistema emite el anuncio broadcast ACTION_BATTERY_CHANGED de forma permanente. De esta forma, cuando se llama a registerReceiver() se obtiene la intención de la última emisión de este anuncio. Por lo tanto, puede usarse para encontrar el estado de la batería sin necesidad de esperar a un futuro cambio en su estado. Se ha incluido este permiso dado que las aplicaciones mal intencionadas pueden ralentizar el dispositivo o volverlo inestable al demandar demasiada memoria.

KILL_ BACKGROUND_PROCESSES – Matar procesos en Background(API 9). Permite llamar a killBackgroundProcesses(String). Al hacer esta llamada el sistema mata de inmediato a todos los procesos de fondo asociados con el paquete indicado. Es el mismo método que usa el sistema cuando necesita memoria. Estos procesos serán reiniciados en el futuro, cuando sea necesario.

REORDER_TASKS – Reordenar tareas. Permite a una aplicación cambiar el orden de la lista de tareas.

INSTALL_SHORTCUT y UNINSTALL_SHORTCUT– Instalar y desinstalar acceso directo(API 19). Permite a una aplicación añadir o eliminar un acceso directo a nuestra aplicación en el escritorio.

GET_PACKAGE_SIZE – Obtener tamaño de un paquete. Permite a una aplicación conocer el tamaño de cualquier paquete.

EXPAND_STATUS_BAR – Expandir barra de estado. Permite a una aplicación expandir o contraer la barra de estado

FOREGROUND_SERVICE – Crear servicios en primer plano. (API 28). Permite a una aplicación crear servicios en primer plano.

Configuraciones del sistema:

SET_WALLPAPER – Poner fondo de pantalla. Permite establecer fondo de pantalla en el escritorio.

SET_WALLPAPER_HITS – Sugerencias de fondo de pantalla. Permite a las aplicaciones establecer sugerencias de fondo de pantalla.

SET_ALARM – Establecer Alarma. Permite a la aplicación enviar una intención para poner una alarma o temporizador en la aplicación Reloj.

SET_TIME_ZONE – Cambiar zona horario. Permite cambiar la zona horaria del sistema.

ACCESS_NOTIFICATION_POLICY – Acceso a política de notificaciones (API 23). Permite conocer la política de notificaciones del sistema.

Audio:

MODY_AUDIO_SETTINGS– Cambiar ajustes de audio. Permite cambiar ajustes globales de audio, como el volumen.

Sincronización:

READ_SYNC_SETTINGS – Leer ajustes de sincronización. Permite saber si tienes sincronización en segundo plano con alguna aplicación (como con un cliente de Twitter o Gmail).

WRITE_SYNC_SETTINGS – Escribir ajustes de sincronización. Permite registrar tu aplicación como adaptador de sincronización (SyncAdapter).

READ_SYNC_STATS – Leer estadísticas de sincronización.

Ubicación:

ACCESS_LOCATION_EXTRA_COMMANDS – Mandar comandos extras de localización. Permite a una aplicación acceder a comandos adicionales de los proveedores de localización. Por ejemplo, tras pedir este permiso podríamos enviar el siguiente comando al GPS, con el método: sendExtraCommand("gps", "delete_aiding_data", null);.

Seguridad:

USE_FINGERPRINT– Usar huella digital(API 23). Permite usar el hardware de reconocimiento de huella digital.

DISABLE_KEYGUARD – Deshabilitar bloqueo de teclado. Permite a las aplicaciones desactivar el bloqueo del teclado si no es seguro.

NOTA: Los permisos peligrosos pertenecen a uno de los 9 grupos anteriores. Estos grupos son importantes dado que el usuario concede o deniega el permiso a un grupo entero. Por el contrario, a partir de la versión 6.0 los permisos normales ya no se clasifican en grupos. Se han organizado en este texto por grupos para una mejor organización.

1.1.3 Permisos para funciones de hardware opcionales

Para **acceder a algunas funciones de hardware (como Bluetooth o la cámara), se necesita un permiso de la app**. Sin embargo, no **todos los dispositivos Android cuentan con estas funciones de hardware**. De modo que, **si tu app solicita el permiso de CAMERA, es importante que también incluyas la etiqueta <uses-feature>** en tu manifiesto a fin de declarar si se requiere o no esta función. Por ejemplo:

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

Si declaras **android:required="false" para la función, Google Play permitirá que tu app se instale en dispositivos que no cuentan con esa función**. Luego, para verificar si **el dispositivo actual tiene la función durante el tiempo de ejecución, llama a PackageManager.hasSystemFeature()** e inhabilita esa función de manera fluida, si no está disponible.

Si **no proporcionas la etiqueta <uses-feature>, cuando Google Play detecte que tu app solicita el permiso correspondiente, asumirá que requiere esta función**. De este modo, **filtra tu app desde dispositivos que no cuentan con la función**, como si **declararas android:required="true"** en la etiqueta <uses-feature>.

No **lo hacemos en el curso, solicitaremos los permisos** , normalmente pero es **información adicional.**

1.1.4 Añadiendo permisos al manifest

Añadiremos los siguientes permisos al comienzo del manifest.xml. Son los que vamos a usar en la **aplicación para añadir fotos de la galería y almacenar fotos** tomadas en **nuestro espacio de memoria externa de la aplicación**. Es **obligatorio si la versión mínima de API** es **anterior a 24**, vamos a tener que pedir permiso para leer ficheros de la memoria externa. En *AndroidManifest.xml* añade dentro de la etiqueta <manifest …> </manifest> el siguiente código. **Hacerlo en cualquier caso**.

Los **permisos normales sólo hace falta añadidlos en el manifest.xml.** Los **permisos peligrosos** o sensibles **además de añadirlos en el manifest.xml**, hay que **pedirlos a la aplicación** como se os muestra en el código posterior.

Este apartado es más complejo porque **debemos asegurarnos de que tenemos permisos** para guardar las fotografías en Android. Para ello debemos añadir unos cuantos métodos a nuestro código.

Podéis veer los enlaces al curso para intentar seguir esta explicación también

http://www.androidcurso.com/index.php/282

http://www.androidcurso.com/index.php/626

A partir de **la versión 6 de Android los permisos cambian tenemos que solicitar los permisos no vale con declararlos sólo en el manifest.** Hemos **de solicitar el permiso**

explícitamente al usuario si es del grupo de peligrosos. Para ello vamos a usar el proyecto EnvioSMS2021.

1.1.5 Añadiendo permisos a EnvioSMS2021

Lo primero es **añadir los permisos al manifest**. Los colocamos junto antes de la etiqueta application. Como veis **no todos son peligrosos .INTERNET y VIBRATE no lo son. El proyecto esta disponible en el aula virtual.**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.enviosms2021">

    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.READ_CONTACTS"
/>
    <uses-permission
android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.RECEIVE_SMS" />
    <uses-permission android:name="android.permission.READ_SMS" />
    <uses-permission android:name="android.permission.WRITE_SMS" />
    <uses-permission android:name="android.permission.RECEIVE_MMS" />
    <uses-permission android:name="android.permission.WRITE" />
    <uses-permission android:name="android.permission.VIBRATE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <application
```

En la actividad principal **SMSActivity, creamos un array de permisos y llamamos a un método solicitar permiso.** Los **permisos los pasamos por medio de una Array de Strings**. En el paquete **Manifest.permission.* encontrareis todos los permisos.**

```
String permisos[] = { Manifest.permission.READ_CONTACTS,
        Manifest.permission.READ_PHONE_STATE,
        Manifest.permission.RECEIVE_SMS,
        Manifest.permission.READ_SMS,
        Manifest.permission.SEND_SMS,
        Manifest.permission.RECEIVE_MMS,
        Manifest.permission.WRITE_EXTERNAL_STORAGE};


solicitarPermiso(permisos, "Necesitamos permisos de CONTACTOS Y SMS
para que funcione la APP", PERMISO_MULTIPLES_SMS_REQUEST_CODE);
```

Para **comprobar si tenemos los permisos de** almacenamiento implementamos los métodos **solicitarPermiso**, hayPermisoContactos y hayPermisoMensajes

```
public void solicitarPermiso(final String[] permiso, String
justificacion,
                             final int requestCode) {

    for(String perm : permiso) {

        if (ActivityCompat.shouldShowRequestPermissionRationale(this,
perm)) {
            new AlertDialog.Builder(this)
                    .setTitle("Solicitud de permiso")
                    .setMessage(justificacion)
                    .setPositiveButton("Ok", new
DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog,
int whichButton) {

ActivityCompat.requestPermissions(SMSActivity.this,
                                    new String[]{perm}, requestCode);
                        }
                    })
                    .show();
        } else {
            ActivityCompat.requestPermissions(this,
                    new String[]{perm}, requestCode);
        }

    }
}


public boolean hayPermisoContactos() {
    //La función checkselfpermission pregunta si el permiso se ha
concedido previamente
    return (ActivityCompat.checkSelfPermission(
            this, Manifest.permission.READ_CONTACTS)
            == PackageManager.PERMISSION_GRANTED);
}

public boolean hayPermisoMensajes() {
    return (ActivityCompat.checkSelfPermission(
            this, Manifest.permission.SEND_SMS)
            == PackageManager.PERMISSION_GRANTED);
}

}
```

1.1.6 Diseccionando el Código

La parte de **permisos en Android es bastante complicada**. Intentaré ser breve y explicar le funcionamiento en detalle sobre el código. Vamos a empezar con la petición de permisos.

Tenemos tres opciones:

1. El permiso ya se ha concedido, y no se vuelve a preguntar
2. El permiso se ha denegado, con la opción no se vuelve a preguntar. En este caso se debería indicar al usuario que conceda el permiso manualmente.
3. El permiso se ha concedido o denegado una vez, pero se vuelve a preguntar siempre.

Es habitual solicitar los permisos llamando al **método solicitar permisos en el onCreate de la actividad que los necesita, en este caso SMSActivity.**. Vemos el método solicitarPermiso.

```
public void solicitarPermiso(final String[] permiso, String
justificacion,
                             final int requestCode) {

    for(String perm : permiso) {

        if (ActivityCompat.shouldShowRequestPermissionRationale(this,
perm)) {
            new AlertDialog.Builder(this)
                    .setTitle("Solicitud de permiso")
                    .setMessage(justificacion)
                    .setPositiveButton("Ok", new
DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog,
int whichButton) {

ActivityCompat.requestPermissions(SMSActivity.this,
                                    new String[]{perm}, requestCode);
                        }
                    })
                    .show();
        } else {
            ActivityCompat.requestPermissions(this,
                    new String[]{perm}, requestCode);
        }

    }
}
```

El método **shouldShowRequestPermissionRationale va a devolver verdadero** si al **usuario nunca le ha solicitado el permiso la aplicación**, o **se le ha solicitado pero ha indicado que se** pregunte cada vez acerca del permiso. Devolverá falso, si ya se ha **solicitado y el usuario ha contestado que no se vuelva a preguntar**. En este caso, se le debería indicar al usuario que debe dar los permisos manualmente. En nuestro caso y para que no os falle la aplicación volveremos a solicitar el permiso.

En caso de que **shouldShowRequestPermissionRationale, nos devuelva verdadero** creamos un **dialogo con la clase AlertDialog y el método builder**. En este se indica el nombre **del permiso, el título, la justificación** de porque debemos pedir el permiso y un botón de ok. A ese botón ok le añadimos un listener con la petición de permisos en caso de

```
 new AlertDialog.Builder(fragment.getActivity())
                    .setTitle("Solicitud de permiso")
                    .setMessage(justificacion)
                    .setPositiveButton("Ok", new
DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int whichButton) {
      fragment.requestPermissions(new String[] {perm}, requestCode);
                }})
            .show();
     } else {
       fragment.requestPermissions(new String[] {perm},      requestCode);
     }
```

El método **hayPermisoContactos() y hayPermisoMensajes(),** nos va a comprobar **si los permisos para contactos y mensajes han sido concedido o no**. Si no lo hacemos e intentamos almacenar fotografías en nuestra aplicación, el programa producirá una excepción y fallará.

El método ActivityCompat.*checkSelfPermission* nos comprueba si el permiso ya ha sido concedido a la aplicación.

```
public boolean hayPermisoContactos() {
    //La función checkselfpermission pregunta si el permiso se ha
concedido previamente
    return (ActivityCompat.checkSelfPermission(
            this, Manifest.permission.READ_CONTACTS)
            == PackageManager.PERMISSION_GRANTED);
}

public boolean hayPermisoMensajes() {
    return (ActivityCompat.checkSelfPermission(
            this, Manifest.permission.SEND_SMS)
```

```
            == PackageManager.PERMISSION_GRANTED);
}
```

1.1.7 Ejercicio.

Comprobar en el código donde se mandan o reciben SMS y se leen los contactos, **porque antes de realizar la funcionalidad debemos comprobar si tenemos permisos**. En caso de no tener permisos, con la clase TOAST mandar un mensaje indicando que no se pueden leer los contactos o no se puede enviar SMS, en su caso.

1.1.8 Android 9. Permisos

Con el **objetivo de mejorar la privacidad de usuario**, **Android 9 introduce varios cambios de comportamiento**, entre los que se **incluyen nuevas reglas de permisos y grupos de permisos relacionados con las llamadas telefónicas,** el **estado del teléfono y análisis de WiFi.** Estos cambios **afectan a todas las aplicaciones que se ejecuten en Android 9**, sin importar la versión de SDK a la que se orienten.

Entre las nuevas medidas, **Android 9 restringe el acceso a los registros de llamadas**. Para ello, se **introduce el grupo de permisos CALL_LOG**, y **se mueven al mismo los permisos READ_CALL_LOG, WRITE_CALL_LOG y PROCESS_OUTGOING_CALLS.** En **versiones anteriores de Android, estos permisos se encontraban en el grupo de permisos PHONE**.

Este nuevo g**rupo de permisos brinda a los usuarios más control y visibilidad respecto de las aplicaciones** que necesiten acceder a **información confidencial sobre llamadas telefónicas**, como puede ser la **lectura de registro de llamadas y la identificación de los números telefónicos**.

Al mismo tiempo, **se ha modificado la política de Google Play y solo las aplicaciones por defecto tendrán acceso al registro de SMS y de llamadas**, esto es, aplicaciones **cuya única finalidad sea el envío y recepción de SMS o llamadas. De** acuerdo a esto, **aplicaciones como WhatsApp, que piden acceso a los SMS y al teléfono del usuario de cara a su activación, tendrían que eliminar dichos permisos** ya que no los utilizan como **base de su funcionamiento**.

Eso sí, al tratarse **de una política de la Google Play, aquellas aplicaciones que se descarguen fuera de Google Play no** tendrán que **asumir la restricción**. Incluso aunque se descarguen desde dentro del mismo, **al no tratarse de una medida implementada en el sistema,** los **desarrolladores podrían continuar con las peticiones**, bajo **el riesgo de ser expulsados de Google Play.**

1.1.9 Permisos Android 10

En la **versión 10 también se introducen importantes novedades relacionadas con la privacidad**. La más **importante es el ámbito de almacenamiento (Scoped Storage**), que **consiste en que las aplicaciones solo pueden ver el contenido de las carpetas creadas por ellas**. Gracias a **esto ya no va a ser necesario pedir permiso** para **acceder al almacenamiento externo**, si accedemos a ficheros creados por la aplicación.

Otra novedad es que **se añaden capas encima del sistema de permisos de Android** 9. Se pretende **que permisos sensibles como micrófono y localización solo sean válidos cuando la app está en primer plano**. Por ejemplo, si queremos que **una aplicación pueda localizarnos en segundo plano, hemos de darle el nuevo permiso ACCESS_BACKGROUND_LOCATION.**

Se **añaden nuevas restricciones. Las apps deben tener permiso de firma READ_PRIVILEGE_STATE_STORAGE** para acceder a los **identificadores del dispositivo que no se pueden restablecer, incluido IMEI** y **número de serie**.

Tampoco **podrán habilitar/deshabilitar WiFi. WifiManager.setWifiEnabled**() **siempre mostrará false.** Para esto será obligatorio usar el **panel de configuración**.

Se **debe tener el permiso ACCESS_FINE_LOCATION para** poder usar varios métodos **dentro de las API de WiFi y Bluetooth, relacionadas con localización**.

Se introduce **el permiso en tiempo de ejecución ACTIVITY_RECOGNITION para apps que necesitan detectar el recuento de pasos del usuario o clasificar su actividad física** (caminar, ir en bicicleta o en un vehículo). Así los **usuarios** tienen **mayor visibilidad sobre cómo se usan los datos del sensor del dispositivo.**

2 Referencias

El gran libro de Android, 8ª edición

https://developer.android.com/training/permissions/requesting?hl=es-419

http://www.androidcurso.com/index.php/626