Permisos

Seguridad

- La seguridad es un aspecto clave de todo sistema. Si nos descargáramos una aplicación maliciosa de Internet o de Google Play Store, esta podría leer nuestra lista de contactos, averiguar nuestra posición GPS, mandar toda esta información por Internet y terminar enviando 50 mensajes SMS.
- Gracias al sistema de permisos, se consigue impedir que las aplicaciones realicen acciones comprometidas, si previamente no han solicitado el permiso adecuado.
- Si queremos que una aplicación tenga acceso a partes del sistema que pueden comprometer la seguridad del sistema o la privacidad/"bolsillo" del usuario hemos de utilizar un modelo de permisos, de forma que el usuario conozca los riesgos antes de instalar la aplicación.

Esquema de permisos

- Para proteger ciertos recursos y características especiales del sistema, Android define un esquema de permisos. Toda aplicación que acceda a estos recursos está obligada a declarar su intención de usarlos. En caso de que una aplicación intente acceder a un recurso del que no ha solicitado permiso, se generará una excepción de permiso y la aplicación será interrumpida inmediatamente.
- Para solicitar un determinado permiso en la aplicación, se tiene que incluir una etiqueta <uses-permission> en el fichero AndroidManifest.xml.
- En el siguiente ejemplo se solicitan dos permisos:

```
<manifest package="org.example.mi_aplicacion" >
   ...
   <uses-permission android:name="android.permission.RECEIVE_SMS"/>
   <uses-permission android:name="android.permission.SEND_SMS"/>
</manifest>
```

Tratamiento de permisos

- Hasta Android 6.0. el usuario concede los permisos a una aplicación en el momento de la instalación. Si no está de acuerdo con algún permiso, la única alternativa para el usuario es no instalar la aplicación. Una vez instalada la aplicación, puede realizar las acciones asociadas a estos permisos tantas veces como desee y cuando desee. Esta forma de trabajar dejaba a los usuarios indefensos ante posibles abusos ya que al final se resigna y acaba aceptando prácticamente cualquier permiso.
- Desde la versión 6 se introducen importantes novedades a la hora de conceder los permisos a las aplicaciones, fijando tipos de permisos. Una medida de seguridad que da al usuario más control y privacidad (y al programador más quebraderos de cabeza!).

Tipos de permisos

- **Normales**:
  - Usuario: En el proceso de instalación el usuario da el visto bueno a los permisos normales (ejemplo, Internet).
  - Programador: Basta con declarar en Manifest
- **Peligrosos**: Se dividen en 9 grupos (10 desde Android 9): almacenamiento, localización, teléfono, SMS, contactos, calendario, cámara, micrófono y sensor de ritmo cardíaco.
  - Usuario: No son concedidos en la instalación. La aplicación consultará al usuario si quiere conceder un permiso peligroso en el momento de utilizarlo por primera vez. Además se recomienda que la aplicación indique para que 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 que permisos concede y cuáles no.
  - Programador: Hay que declarar en Manifest y trabajar código java!
- **De firma**: Se otorga el permiso en el momento de la instalación, pero solo cuando la aplicación que intenta usar el permiso está firmada por el mismo certificado que la aplicación que define el permiso. (Nota: algunos permisos de firma no son para uso de aplicaciones de terceros).
- **Especiales**: Hay un 2 permisos que no se comportan como permisos normales ni peligrosos. SYSTEM_ALERT_WINDOW y WRITE_SETTINGS.

Normales

- A partir de Android 9:

ACCESS_LOCATION_EXTRA_COMMANDS
ACCESS_NETWORK_STATE
ACCESS_NOTIFICATION_POLICY
ACCESS_WIFI_STATE
BLUETOOTH
BLUETOOTH_ADMIN
BROADCAST_STICKY
CHANGE_NETWORK_STATE
CHANGE_WIFI_MULTICAST_STATE
CHANGE_WIFI_STATE
DISABLE_KEYGUARD
EXPAND_STATUS_BAR
FOREGROUND_SERVICE
GET_PACKAGE_SIZE
INSTALL_SHORTCUT
INTERNET
KILL_BACKGROUND_PROCESSES
MANAGE_OWN_CALLS
MODIFY_AUDIO_SETTINGS
NFC
READ_SYNC_SETTINGS
READ_SYNC_STATS
RECEIVE_BOOT_COMPLETED
REORDER_TASKS
REQUEST_COMPANION_RUN_IN_BACKGROUND
REQUEST_COMPANION_USE_DATA_IN_BACKGROUND
REQUEST_DELETE_PACKAGES
REQUEST_IGNORE_BATTERY_OPTIMIZATIONS
SET_ALARM
SET_WALLPAPER
SET_WALLPAPER_HINTS
TRANSMIT_IR
USE_FINGERPRINT
VIBRATE
WAKE_LOCK
WRITE_SYNC_SETTINGS

Normales

Comunicaciones

- **INTERNET** – Permite que la aplicación cree sockets de red y use protocolos de red personalizados.
- **ACCESS_NETWORK_STATE** - Ver estado de red. Permite obtener información sobre todas las redes. Por ejemplo para saber si tenemos conexión a internet.
- **CHANGE_NETWORK_STATE** - 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.

- Ej:

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

Normales

Conexión WIFI

- **ACCESS_WIFI_STATE**: Permite conocer las redes Wi-Fi disponibles.
- **CHANGE_WIFI_STATE**: Permite cambiar el estado de conectividad Wi-Fi.
- **CHANGE_WIFI_MULTICAST_STATE**: 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.

Normales

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 nunca.
- **FLASHLIGHT** – Linterna. Permite encender el flash de la cámara.
- **VIBRATE** – Control de la vibración. Permite hacer vibrar al teléfono

Audio

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

Ubicación

- **ACCESS_LOCATION_EXTRA_COMMANDS** – Mandar comandos extras de localización. Permite a una aplicación acceder a comandos adicionales de los proveedores de localización.

Normales

Aplicaciones

- **RECEIVE_BOOT_COMPLETED** – Ejecución automática al encender el teléfono. Permite a una aplicación recibir el 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. 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**– Instalar acceso directo (API 19). Permite a una aplicación añadir un acceso directo a la aplicación en el escritorio.
- **UNINSTALL_SHORTCUT** – Desinstalar acceso directo. OBSOLETO desde Android 9.
- **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

Normales

Configuraciones del sistema

- **CHANGE_CONFIGURATION** –Permite cambiar la configuración del sistema (como la configuración local)
- **SET_WALLPAPER** –Permite establecer fondo de pantalla en el escritorio.
- **SET_WALLPAPER_HITS** –Permite a las aplicaciones establecer sugerencias de fondo de pantalla.
- **SET_ALARM** –Permite a la aplicación enviar una intención para poner una alarma o temporizador en la aplicación Reloj.
- **SET_TIME_ZONE** –Permite cambiar la zona horaria del sistema.
- **ACCESS_NOTIFICATION_POLICY** Permite conocer la política de notificaciones del sistema.

Normales

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**.**

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.

Obligatoriedad de los permisos normales

- No siempre es obligatorio especificar el permiso cuando es un permiso normal.
- Por ejemplo, para visualizar una página web haremos uso del intent implícito

```
Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.iespabloserrano.es"));
```

que abrirá el navegador web, será esa aplicación navegador la que haya fijado el permiso.

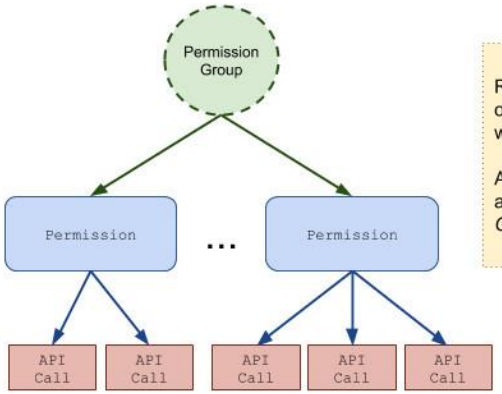
- Pero ante la duda, más vale ponerlo que no que falte!

Grupos de permisos peligrosos

Permission Group	Permissions
CALENDAR	• READ_CALENDAR • WRITE_CALENDAR
CALL_LOG	• READ_CALL_LOG • WRITE_CALL_LOG • PROCESS_OUTGOING_CALLS
CAMERA	• CAMERA
CONTACTS	• READ_CONTACTS • WRITE_CONTACTS • GET_ACCOUNTS
LOCATION	• ACCESS_FINE_LOCATION • ACCESS_COARSE_LOCATION
MICROPHONE	• RECORD_AUDIO
PHONE	• READ_PHONE_STATE • READ_PHONE_NUMBERS • CALL_PHONE • ANSWER_PHONE_CALLS • ADD_VOICEMAIL • USE_SIP
SENSORS	• BODY_SENSORS
SMS	• SEND_SMS • RECEIVE_SMS • READ_SMS • RECEIVE_WAP_PUSH • RECEIVE_MMS
STORAGE	• READ_EXTERNAL_STORAGE • WRITE_EXTERNAL_STORAGE

Acceso restringido a registros de llamadas

En Android 9, se introduce el grupo de permisos `CALL_LOG` y se mueven a este 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`.



Related API Calls and access to object properties are associated with a specific permission request.

And related permission requests are rolled up into a *Permissions Group*.

Peligrosos: STORAGE

Almacenamiento Externo

- **READ_EXTERNAL_STORAGE**– Leer almacenamiento. Permite leer archivos en la memoria externa. (Por lo tanto, como usuario has de tener cuidado con la información que dejas en ella). Este permiso se introdujo en Android 4.1 (API 16), antes todas las aplicaciones podían leer en la memoria externa.
- **WRITE_EXTERNAL_STORAGE**– Modificar/eliminar almacenamiento (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. Pero al permitirlo también podrán modificar/eliminar ficheros externos creados por otras aplicaciones.
  - a partir de Android 4.4 (nivel de API 19), ya no es necesario que la app solicite el permiso cuando intente realizar operaciones de escritura en sus propios directorios específicos en el almacenamiento externo. Sin embargo, el permiso es necesario en hasta el nivel de API 18. Por lo tanto, puede declararse que este permiso solo es necesario hasta el nivel de API 18 con una declaración como la siguiente:

```
<uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE"
    android:maxSdkVersion="18" />
```

Peligrosos: LOCATION

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).

Peligrosos - PHONE

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 de marcar nº. 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 identidad y 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.
- **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 (SIP es un protocolo Internet para comunicaciones en vivo utilizado en la configuración de llamadas de voz o video).
- **READ_PHONE_NUMBERS** – Permite el acceso de lectura al número de teléfono del dispositivo. (API 26)
- **ANSWER_PHONE_CALLS** – Permite que la aplicación responda a una llamada telefónica entrante (API 26).

Peligrosos – CALL_LOG

- **READ_CALL_LOG** y **WRITE_CALL_LOG** – Leer y modificar el registro de llamadas telefónicas.
- **PROCESS_OUTGOING_CALLS** –Procesar llamadas salientes. Permite a la aplicación controlar, modificar o abortar las llamadas salientes.

Peligrosos: SMS

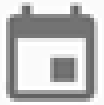
Mensajes de texto (SMS)

- **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 entrantes.
- **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 (ya en desuso).

Peligrosos: CONTACTS y CALENDAR

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** – Permite que la aplicación obtenga la lista de cuentas conocidas por el teléfono. Esto puede incluir cualquier cuenta creada por otras aplicaciones que haya instalado.



Calendario

- **READ_CALENDAR** – Permite leer información del calendario del usuario.
- **WRITE_CONTACTS** – Permite escribir en el calendario, pero no leerlo.

Peligrosos: CAMERA, MICROPHONE y SENSORS


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.

Ajustes automáticos de permisos

- Con el tiempo, es posible que se agreguen restricciones nuevas a la plataforma a fin de que, para usar determinadas APIs, la app deba solicitar un permiso que antes no necesitaba. Debido a que las apps existentes suponen que el acceso a esas API es libre, Android puede implementar la solicitud del permiso nuevo en el manifiesto de la app para evitar dañarla en la versión nueva de la plataforma. Android toma las decisiones relacionadas con el hecho de que una app pueda necesitar el permiso según el valor indicado para el atributo **targetSdkVersion**. Si el valor es inferior al de la versión en la cual el permiso se agregó, Android agrega el permiso.
- Por ejemplo, el permiso WRITE_EXTERNAL_STORAGE se agregó en el nivel de API 4 para restringir el acceso a los espacios de almacenamiento compartido. Si el atributo targetSdkVersion de la aplicación es 3 o inferior (se hizo hace mucho tiempo y no se ha actualizado por el desarrollador), este permiso se agrega a la app en versiones nuevas de Android.
- **Advertencia:** Si un permiso se agrega automáticamente a la app, en Google Play, la app indicará que necesita estos permisos adicionales aunque podría no necesitarlos realmente.
- Para evitar esto y quitar los permisos predeterminados que no necesita, siempre actualiza el atributo targetSdkVersion para que tenga el valor más alto posible.

Permisos especiales

- Un par de permisos no se comportan como permisos normales ni peligrosos:
  - **SYSTEM_ALERT_WINDOW** - Permite que una aplicación cree ventanas que se muestran sobre todas las demás aplicaciones. Muy pocas aplicaciones deberían usar este permiso; estas ventanas están destinadas a la interacción a nivel del sistema con el usuario.
  - **WRITE_SETTINGS** - Permite que una aplicación lea o escriba la configuración del sistema.
- Son particularmente confidenciales; por ello, la mayoría de las apps no deben usarlos.
- Si una app necesita uno de estos permisos, debe declararlo en el manifiesto y enviar una intent en la que solicite la autorización del usuario. El sistema responde a la intent mostrando una pantalla de administración detallada al usuario.

Permisos de firma

As of Android 8.1 (API level 27), the following permissions that third-party apps can use are classified as `PROTECTION_SIGNATURE`:

- `BIND_ACCESSIBILITY_SERVICE`
- `BIND_AUTOFILL_SERVICE`
- `BIND_CARRIER_SERVICES`
- `BIND_CHOOSER_TARGET_SERVICE`
- `BIND_CONDITION_PROVIDER_SERVICE`
- `BIND_DEVICE_ADMIN`
- `BIND_DREAM_SERVICE`
- `BIND_INCALL_SERVICE`
- `BIND_INPUT_METHOD`
- `BIND_MIDI_DEVICE_SERVICE`
- `BIND_NFC_SERVICE`
- `BIND_NOTIFICATION_LISTENER_SERVICE`
- `BIND_PRINT_SERVICE`
- `BIND_SCREENING_SERVICE`
- `BIND_TELECOM_CONNECTION_SERVICE`
- `BIND_TEXT_SERVICE`
- `BIND_TV_INPUT`
- `BIND_VISUAL_VOICEMAIL_SERVICE`
- `BIND_VOICE_INTERACTION`
- `BIND_VPN_SERVICE`
- `BIND_VR_LISTENER_SERVICE`
- `BIND_WALLPAPER`
- `CLEAR_APP_CACHE`
- `MANAGE_DOCUMENTS`
- `READ_VOICEMAIL`
- `REQUEST_INSTALL_PACKAGES`
- `SYSTEM_ALERT_WINDOW`
- `WRITE_SETTINGS`
- `WRITE_VOICEMAIL`

- El propósito del nivel de permiso "Firma" es que dos aplicaciones del mismo desarrollador puedan compartir datos sin problemas, sin molestar al usuario.
- Para ello el desarrollador podrá definir sus propios permisos.

Otros permisos

- 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.
- Ej.: UPDATE_DEVICE_STATS - Permite que una aplicación actualice las estadísticas del dispositivo. No para uso de aplicaciones de terceros.

Permisos del grupo

- Desde Android 6 y hasta Android 8.0, si una app solicitaba un permiso en el tiempo de ejecución y este se otorgaba, el sistema también otorgaba a esta, de manera incorrecta, el resto de los permisos que pertenecían al mismo grupo de permisos y estaban registrados en el manifiesto.
- Desde Android 8.0, se ha corregido este comportamiento. La app solo obtiene los permisos que ha solicitado explícitamente. Sin embargo, una vez que el usuario otorga permiso a la app, todas las solicitudes de permisos posteriores de ese grupo de permisos se otorgan de manera automática.
- Por ejemplo, supón que una aplicación presenta READ_EXTERNAL_STORAGE y WRITE_EXTERNAL_STORAGE en su manifiesto. La aplicación solicita READ_EXTERNAL_STORAGE y el usuario lo otorga. Si la aplicación tiene como objetivo el nivel de API 24 o niveles inferiores, el sistema también otorga WRITE_EXTERNAL_STORAGE al mismo tiempo, porque pertenece al mismo grupo de permisos STORAGE y también está registrado en el manifiesto. Si la app se ejecuta en Android 8.0+, el sistema solo otorga READ_EXTERNAL_STORAGE en ese momento. Sin embargo, si la app luego solicita WRITE_EXTERNAL_STORAGE, el sistema inmediatamente otorga ese privilegio sin notificar al usuario.

Cambios en Android 10

El acceso a la ubicación del dispositivo en segundo plano requiere permiso

Para admitir el control adicional que tienen los usuarios sobre el acceso de la app a la información de ubicación, en Android 10 se introduce el permiso `ACCESS_BACKGROUND_LOCATION`.

A diferencia de los permisos `ACCESS_FINE_LOCATION` y `ACCESS_COARSE_LOCATION`, el permiso `ACCESS_BACKGROUND_LOCATION` solo afecta el acceso de una app a la ubicación cuando esta se ejecuta en segundo plano. Se considera que una app accede a la ubicación en segundo plano siempre y cuando no se cumpla alguna de las siguientes condiciones:

- Se muestra una actividad que pertenece a otra app.
- La app ejecuta un servicio en primer plano que declaró un tipo de servicio en primer plano de `location`.

  Si quieres declarar un tipo de servicio en primer plano para un servicio de tu app, debes configurar la `targetSdkVersion` o `compileSdkVersion` de tu app en `29` o una versión posterior. Obtén más información sobre cómo los servicios en primer plano pueden continuar acciones iniciadas por el usuario que requieren acceso a la ubicación.

El acceso se otorga automáticamente cuando las apps se orientan a Android 9 o versiones anteriores

Si tu app se ejecuta en Android 10 o versiones posteriores, pero está orientada a Android 9 (API nivel 28) o versiones anteriores, la plataforma se comportará de la siguiente manera:

- Si tu app declara un elemento `<uses-permission>` para `ACCESS_FINE_LOCATION` o `ACCESS_COARSE_LOCATION`, el sistema agregará automáticamente un elemento `<uses-permission>` en `ACCESS_BACKGROUND_LOCATION` durante la instalación.
- Si tu app solicita `ACCESS_FINE_LOCATION` o `ACCESS_COARSE_LOCATION`, el sistema automáticamente agrega `ACCESS_BACKGROUND_LOCATION` a la solicitud.

Consejos de Google sobre los permisos

- Solicitar solo los permisos necesarios.
- No solicitar todos los permisos a la vez, sino cuando hagan falta.
- Explicar la causa de la solicitud.
- Prestar atención a los permisos requeridos por las bibliotecas. Cuando se incluye una biblioteca, también hereda sus requisitos de permiso. Debe tenerse en cuenta lo que está incluyendo, los permisos que requieren y para qué se usan esos permisos.

Tratamiento de los permisos peligrosos

- A partir de Android 6 trabajar con acciones que necesiten de un permiso va a suponer un esfuerzo adicional para el programador.
- Antes de realizar la acción tendremos que verificar si tenemos el permiso.
- En caso negativo hay que exponer al usuario para qué lo queremos y pedírselo.
- Si el usuario no nos diera el permiso, tendremos qué decidir qué hacer.

Comprobar permiso

- Hay que llamar al método ContextCompat.checkSelfPermission().
- Si la aplicación tiene el permiso, el método devuelve PackageManager.PERMISSION_GRANTED, y la aplicación puede continuar con la operación.
- Si la aplicación no tiene el permiso, el método devuelve PERMISSION_DENIED y la aplicación tiene que pedir explícitamente permiso al usuario.
- Ejemplo:

```
// ¿Tengo el permiso para hacer la accion?
int chequeoPermiso = ContextCompat.checkSelfPermission(getApplicationContext(), Manifest.permission.CALL_PHONE);
if  (chequeoPermiso== PackageManager.PERMISSION_GRANTED) {
        hacerLlamada();
} else {
        pedirPermiso();
}
```

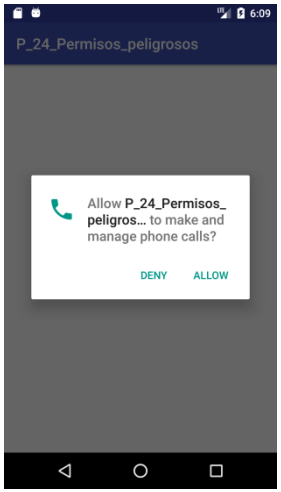
Pedir permiso

- En algunas circunstancias, es posible que sea preciso ayudar al usuario a comprender por qué la aplicación necesita un permiso.
- Por ejemplo, si un usuario inicia una aplicación de fotografía, probablemente no se sorprenda si la aplicación le pide permiso para utilizar la cámara, pero el usuario puede no entender por qué la aplicación quiere acceder a la ubicación o los contactos del usuario.
- Antes de solicitar un permiso, es conveniente proporcionar una explicación para el usuario pero sin abrumar; si hay demasiadas explicaciones, el usuario podría encontrar la aplicación frustrante y cancelarla.
- Podríamos proporcionar una explicación sólo si el usuario ya ha rechazado la solicitud de permiso. Si un usuario sigue tratando de utilizar la funcionalidad que requiere un permiso, pero mantiene rechazar la solicitud de permiso, probablemente indica que el usuario no entiende por qué la aplicación necesita el permiso para proporcionar esa funcionalidad. En una situación así, es probable que sea una buena idea para mostrar una explicación.
- Para ayudar a encontrar situaciones en las que el usuario pueda necesitar una explicación, Android proporciona el método shouldShowRequestPermissionRationale(Activity activity, String permission) que devuelve **true** si la aplicación ha solicitado este permiso previamente y el usuario ha denegado la solicitud.
- Ej.:

```
if (ActivityCompat.shouldShowRequestPermissionRationale(this, Manifest.permission.CALL_PHONE)) {
        //explicar motivo y volver a solicitar
}
else
        //solicitar permiso
```

Solicitar permiso

- Hay que llamar al método requestPermissions(Activity activity, String[] permissions, int requestCode) para solicitar el permiso correspondiente.



  - Aparece un diálogo estándar de Android que no se puede personalizar.
  - Parámetros: contexto de la actividad, array de permisos solicitados y un número entero *de código de solicitud* que sirva para identificar esta solicitud de permiso (es una constante declarada para la actividad).

- Ej:

```
ActivityCompat.requestPermissions(this,new String[]{Manifest.permission.CALL_PHONE},SOLICITUD_PERMISO_LLAMADA);
```

Manejar la respuesta

- Cuando el usuario responde, el sistema invoca al método onRequestPermissionsResult() pasándole dicha respuesta del usuario.
- Debemos sobreescribir ese método con las acciones a realizar según lo que haya elegido el usuario.
- El método recibe el mismo código de solicitud que se pasó en requestPermissions().

```
@Override
public void onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults) {
    switch (requestCode) {
        case MI_CONSTANTE_PARA_EL_PERMISO : {
            // Si el diálogo ha sido cancelado, el array  está vacío.
            if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                // Permiso concedido
                // Acciones a realizar
            } else {
                // Permiso rechazado
                // Acciones a realizar. Por ejemplo, mostrar un Toast explicativo de que no funciona
            }
        // otros 'case' para chequear otros posibles permisos que necesita la aplicación
    }
}
```

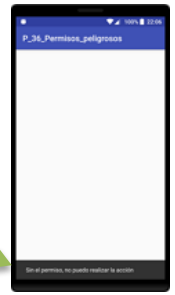
Ejemplo


```
@Override
   public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {
      if (requestCode == SOLICITUD_PERMISO_LLAMADA) {
         if (grantResults.length == 1 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            //Realizamos la acción
            hacerLlamada();
         } else {
            Toast.makeText(this,"Sin el permiso, no puedo realizar la acción",Toast.LENGTH_LONG).show();
         }
         return;
      }
   }
```

SnackBar

- Para explicar la necesidad del permiso, se puede utilizar SnackBar.
- Snackbar es un componente de notificación que forma parte de Material Design (por tanto, hay que añadir la librería correspondiente implementation 'com.google.android.material:material:1.0.0' si hemos escogido plantilla Empty, por eso es recomendable usar plantilla Basic que ya la tiene ☺).
- Estas notificaciones se muestran en la parte inferior de la pantalla, son semejantes a los Toast

```
vista = findViewById(R.id.activity_main); //layout donde queremos SnackBar
…
Snackbar.make(vista, "Mensaje 2", Snackbar.LENGTH_SHORT).show();
```



- Opcionalmente ofrecen al usuario la posibilidad de realizar alguna acción.

```
Snackbar.make(vista, R.string.mensaje, Snackbar.LENGTH_INDEFINITE)
        .setAction("OK", new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                        …
            }
        })
        .show();
```

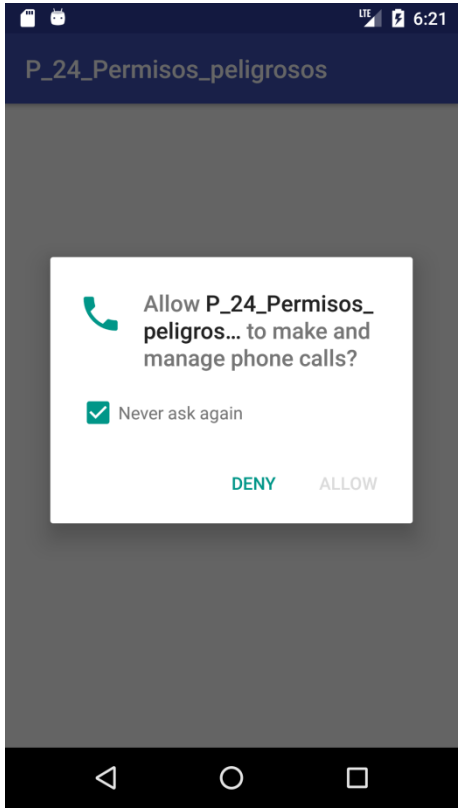
- Sólo habrá una Snackbar visible en pantalla y nunca mostrarán imágenes.

Ejemplo solicitar permiso


```
vista = findViewById(R.id.activity_main); //layout donde queremos SnackBar, HAY QUE IDENTIFICARLO
…
 private void pedirPermiso() {
      if (ActivityCompat.shouldShowRequestPermissionRationale(this, Manifest.permission.CALL_PHONE)) {
         final Activity activity=this;
         Snackbar.make(vista, R.string.mensaje, Snackbar.LENGTH_INDEFINITE)
               .setAction("OK", new View.OnClickListener() {
                  @Override
                  public void onClick(View view) {
                     ActivityCompat.requestPermissions(activity, new String[]{Manifest.permission.CALL_PHONE}, SOLICITUD_PERMISO_LLAMADA);
                  }
               })
               .show();
      } else {
         ActivityCompat.requestPermissions(this,new String[]{Manifest.permission.CALL_PHONE},SOLICITUD_PERMISO_LLAMADA);
      }
   }
```

"Nunca volver a preguntar" o denegación por ajustes

- Si el usuario rechazó la solicitud de permiso en el pasado y selecciona la opción "Never ask again" en el diálogo de solicitud de permiso el método souldShowRequestPermissionRationale() devuelve false. También si desde ajustes se prohíbe que la app tenga ese permiso.
- En ese caso, cuando la app use requestPermissions() para solicitar ese permiso nuevamente, el sistema rechazará la solicitud de inmediato SIN LLEGAR A MOSTRAR EL DIÁLOGO DE SOLICITUD.
- El sistema llama a onRequestPermissionsResult() pasando el valor PERMISSION_DENIED, de la misma manera en que lo haría si el usuario hubiera rechazado explícitamente la solicitud.
- Esto significa que cuando llamas a requestPermissions(), no puedes suponer que haya existido interacción directa con el usuario.



Permisos personalizados

- Además de los permisos definidos por el sistema, los desarrolladores vamos a poder crear nuevos permisos.
- La plataforma Android advierte: "debe evaluar cuidadosamente si es necesario que su aplicación lo haga"
- Ej.:

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

    <permission
      android:name="com.example.myapp.permission.COSTOSA_ACTIVITY"
      android:label="@string/permlab_costosaActivity"
      android:description="@string/permdesc_costosaActivity"
      android:permissionGroup="android.permission-group.CUESTA_DINERO"
      android:protectionLevel="dangerous" />
    ...
</manifest>
```

- Enlace

Permisos para características de hardware opcionales

- El acceso a algunas funciones de hardware (como Bluetooth o la cámara) requiere además otro tipo de permiso de aplicación (**uses-feature**) ya que no todos los dispositivos Android tienen esas características de hardware.
- Ej.: <uses-feature android:name="android.hardware.bluetooth" />
- Sintaxis:

```
<uses-feature
  android:name="string"
  android:required=["true" | "false"] />
```

- El atributo required permite especificar si la aplicación requiere la función declarada y no puede funcionar sin ella, o si prefiere contar con la aplicación y puede funcionar sin ella.

Permisos que implican requisitos de hardware

Categoría	Este permiso...	... implica el requisito de esta función
Bluetooth	BLUETOOTH	android.hardware.bluetooth
	BLUETOOTH_ADMIN	android.hardware.bluetooth
Cámara	CAMERA	android.hardware.camera y android.hardware.camera.autofocus
Ubicación	ACCESS_MOCK_LOCATION	android.hardware.location
	ACCESS_LOCATION_EXTRA_COMMANDS	android.hardware.location
	INSTALL_LOCATION_PROVIDER	android.hardware.location
	ACCESS_COARSE_LOCATION	android.hardware.location.network y android.hardware.location
	ACCESS_FINE_LOCATION	android.hardware.location.gps y android.hardware.location
Micrófono	RECORD_AUDIO	android.hardware.microphone
Telefonía	CALL_PHONE	android.hardware.telephony
	CALL_PRIVILEGED	android.hardware.telephony
	MODIFY_PHONE_STATE	android.hardware.telephony
	PROCESS_OUTGOING_CALLS	android.hardware.telephony
	READ_SMS	android.hardware.telephony
	RECEIVE_SMS	android.hardware.telephony
	RECEIVE_MMS	android.hardware.telephony
	RECEIVE_WAP_PUSH	android.hardware.telephony
	SEND_SMS	android.hardware.telephony
	WRITE_APN_SETTINGS	android.hardware.telephony
	WRITE_SMS	android.hardware.telephony
Wi-Fi	ACCESS_WIFI_STATE	android.hardware.wifi
	CHANGE_WIFI_STATE	android.hardware.wifi
	CHANGE_WIFI_MULTICAST_STATE	android.hardware.wifi

- No siempre es obligatorio fijar <uses-feature>.
- Si una aplicación solicita permisos relacionados con hardware, Google Play considera que la aplicación usa las funciones de hardware subyacentes y, por lo tanto, requiere dichas funciones aunque no se correspondan con las declaraciones de <uses-feature>. Para estos permisos, Google Play agrega las funciones de hardware subyacentes a los metadatos que almacena para la aplicación y establece los filtros correspondientes.
- Por ejemplo, si una aplicación solicita el permiso CAMERA, pero no declara un elemento <uses-feature> para android.hardware.camera, Google Play considera que la aplicación requiere una cámara y no se la mostrará a los usuarios cuyos dispositivos no tengan cámara.
- Si no quieres que Google Play aplique filtrado conforme a una determinada función implícita, puedes inhabilitar el comportamiento. Para hacerlo, declara la función explícitamente en un elemento <uses-feature> e incluye un atributo android:required="false".

Prácticas propuestas

- Realiza la hoja de ejercicios Ejercicios_06_Permisos_e_intents