

Caso práctico



[cottonbro](#) (Pexels)

- ¡Tenemos un nuevo reto! -informa Ada a su equipo-. Uno de nuestros clientes solicita que una de las aplicaciones Android que le desarrollemos sea capaz de tomar fotografías desde la cámara del dispositivo de los usuarios. ¿Qué problema veis?

- Es una acción es muy intrusiva desde el punto de vista de la privacidad de los usuarios de las aplicaciones -responde Pedro-, ya que podría capturar imágenes e incluso vídeos en cualquier momento. Los usuarios deben tener todo el control sobre estas acciones.

- Buena puntualización -señala Ada-. Existen otras muchas acciones consideradas peligrosas desde el punto de vista de la privacidad, ya que el teléfono móvil es un acompañante de todas las personas con muchos sensores capaces de captar información visual, audios,

posicionamiento mediante GPS, etc., conectado a Internet.

- Si bien es cierto -prosigue Ada en su explicación-, el sistema Android mejora en cada paso de API para ofrecer mayores herramientas de control a los usuarios de sus aplicaciones, ofreciendo distintos permisos agrupados en varias categorías dependiendo de la criticidad de los datos o acciones que protejan.

- Propongo indagar los tipos de permisos que ofrece la API y lo que protegen -lanza María como propuesta, la cual es aceptada por todo el equipo de muy buen grado.

Es imprescindible que los desarrolladores de aplicaciones Android conozcan los distintos tipos de permisos, lo que protegen y, sobre todo, que sigan una serie de pautas y buenas prácticas para que los usuarios de sus aplicaciones estén tranquilos usándolas en conciencia de lo que estas pueden hacer sobre sus dispositivos.



[Ministerio de Educación y Formación Profesional](#). (Dominio público)

Materiales formativos de FP Online propiedad del Ministerio de Educación y Formación Profesional.

[Aviso Legal](#) 

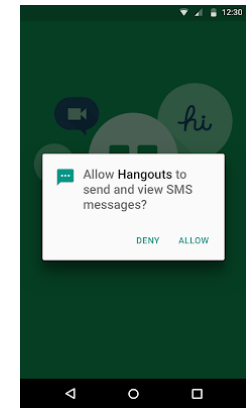
1.- Análisis sobre los permisos

Caso práctico

- Hay tres cosas muy importantes que debemos tener en cuenta cuando hablamos de permisos -explica Ada a su equipo, mostrándoles una pizarra con la información que les quiere transmitir. Quiere que los programadores sean conscientes de cómo deben gestionar los permisos en su aplicación.

- ✓ No todos los datos ni todas las acciones accesibles desde un dispositivo Android son iguales desde el punto de vista de la privacidad. No es lo mismo grabar una conversación privada de nuestro usuario, que acceder a un documento que sea ha generado desde la propia aplicación.
- ✓ Solicitar permisos de acceso a determinados datos o acciones puede provocar que los usuarios sean reacios a usar nuestra aplicación e incluso lleguen a desinstalarla.
- ✓ Es importante minimizar, accediendo a la menor cantidad posible de datos y de acciones en el dispositivo e intentando averiguar la información que nuestra aplicación necesita por otros medios que no impliquen estos permisos. Por ejemplo podemos solicitar el CP del domicilio o lugar de trabajo de nuestro usuario en lugar de acceder a su ubicación para ofrecerle sugerencias cercanas a él.

- El primer paso para construir aplicaciones eficientes y confiables para los usuarios -indica Ada al equipo, que sigue su explicación sin perder detalle-, es conocer los distintos tipos de permisos y sus grupos, para saber cómo Android va a solicitar al usuario el acceso a los mismos.



[Google Developers](#) (Usa educativo nc)

Los permisos de las aplicaciones Android ayudan a respetar la privacidad del usuario, ya que protegen el acceso:

- ✓ Datos restringidos, como el estado del sistema y la información de contacto del usuario.
- ✓ Acciones restringidas, como conectarte a un dispositivo vinculado y grabar audios.

En el caso de que tu aplicación necesite acceder a datos o acciones restringidas a través de un permiso, debes mencionarlo en el archivo de manifiesto AndroidManifest.xml.

Según el grado de confidencialidad del permiso, el sistema podría otorgar el permiso automáticamente o el usuario del dispositivo podría tener que conceder la solicitud. Por ejemplo, si nuestra aplicación solicita permiso para activar la linterna del dispositivo, el sistema otorga ese permiso automáticamente. Sin embargo, si nuestra aplicación necesita leer los contactos del usuario, el sistema le solicita aprobar dicho permiso. Según la versión de la plataforma, el usuario otorga el permiso cuando se instala la aplicación (en Android 5.1 y versiones anteriores) o mientras esta se ejecuta (en Android 6.0 y versiones posteriores).

Sin embargo, si accedes a la documentación oficial, descubrirás que **la necesidad de los permisos para el acceso a los distintos datos y acciones varía con las distintas versiones**, circunstancia que debes tener muy en cuenta a la hora de planificar los permisos de tu aplicación.

1.1.- Permisos necesarios para nuestra aplicación.

Cuando desarrollamos nuestra aplicación, debemos tener en cuenta los casos en que esta use funcionalidades que requieran un permiso. Normalmente, una aplicación necesitará permisos siempre que use información o recursos que no haya creado, o cuando realice acciones que afecten el comportamiento del dispositivo o de demás aplicaciones. Por ejemplo, si una aplicación necesita acceder a Internet, usar la cámara del dispositivo o activar o desactivar la conexión wifi, necesitará los permisos correspondientes.

Nuestra aplicación solo necesita permisos para acciones que realiza directamente. No necesita permisos si solicita a otra aplicación que realice la tarea o proporcione la información (esta acción se denomina **delegación** de permisos). Por ejemplo, si debe leer la lista de direcciones del usuario, necesitará el permiso `READ_CONTACTS`. Pero si usa un **intent** para solicitar información de la aplicación de Contactos del usuario (preinstalada ya en los dispositivos), no necesitará permisos, aunque sí los necesitará la aplicación de Contactos.

Cada aplicación de Android se ejecuta en una zona de pruebas (también conocida como sandbox) con acceso limitado. Si una aplicación necesita usar recursos o información externos a su propia zona de pruebas, debe solicitar el permiso correspondiente haciendo una **declaración del permiso**.

La declaración de estos permisos hace que cada vez que tu aplicación requiera acceder a los recursos protegidos, esta debe pararse y solicitar al usuario el acceso. Esto puede provocar que tus posibles usuarios denieguen el acceso e incluso desinstalen la aplicación porque no comprendan la necesidad de su uso o porque no quieran otorgarlos. Debes saber que existen algunas alternativas que no requieren declarar estos permisos y debes evaluar si tu aplicación debe, por tanto, declararlos.

Además, es posible que existan otras aplicaciones en el terminal que ya cuenten con permisos para realizar la tarea necesaria; Android permite delegar ciertas acciones en otras aplicaciones que ya tengan concedido el acceso a los datos o las acciones requeridas.

Por ejemplo no requieren declarar permisos las siguientes acciones realizadas de esta forma:

- ✓ Para mostrar lugares cercanos solicita un código postal en lugar de acceder a la ubicación.
- ✓ Para hacer fotos o grabar vídeos usa las aplicaciones preinstaladas en el sistema (delega).
- ✓ Desde Android 11 (API) no es necesario solicitar permiso para acceder a contenido multimedia creado desde la propia aplicación previamente.
- ✓ Desde Android 10 no es necesario solicitar permisos para acceder a documentos.

Puedes ver más ejemplos en este enlace [Evaluación de permisos en developer.android.com](https://developer.android.com/permissions-evaluation)  .

1.2.- Agregar permisos al manifiesto.

Para declarar que nuestra aplicación necesita un permiso, dispondremos un elemento **<uses-permission>** en el fichero **AndroidManifest.xml** como elemento secundario del elemento de nivel superior **<manifest>**.

Ejemplo

En el manifiesto de una aplicación que necesite enviar mensajes SMS debería incluirse esta línea:

```
1 <manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.example.snazzyapp">
2     <uses-permission android:name="android.permission.SEND_SMS"/>
3     <application ...>
4         ...
5     </application>
6 </manifest>
```



El comportamiento del sistema después de declarar un permiso depende del grado de confidencialidad de este último. Si el permiso no afecta la privacidad del usuario, el sistema otorga el permiso automáticamente. Si el permiso puede otorgar acceso a información confidencial del usuario, el sistema solicita al usuario que apruebe la solicitud. En el apartado siguiente veremos cuáles son los diferentes tipos de permisos.

1.3.- Tipos de permisos del sistema.

Android categoriza los permisos en diferentes tipos, incluidos los del momento de la instalación, los permisos de tiempo de ejecución y los permisos especiales. Cada tipo de permiso indica el alcance de los datos restringidos a los que tu aplicación puede acceder, así como el alcance de las acciones restringidas que puede realizar, cuando el sistema le otorga ese permiso.

Permisos en el momento de la instalación

Los permisos en el momento de la instalación otorgan acceso limitado a los datos restringidos y permiten que realice acciones restringidas **que casi no afectan al sistema o a otras aplicaciones**. Cuando declaras permisos en el momento de la instalación en la aplicación, el sistema le otorga automáticamente los permisos cuando el usuario la instala. La tienda PlayStore muestra a los usuarios un aviso de permiso en el momento de la instalación al cuando ve la página de detalles de la aplicación. Hay dos tipos:


- ✓ **Permisos normales:** Permiten el acceso a los datos y las acciones que se extienden más allá de la zona de pruebas de tu aplicación. Sin embargo, los datos y las acciones presentan muy poco riesgo para la privacidad del usuario y el funcionamiento de otras aplicaciones. Puedes buscar aquellos permisos normales en [esta página de referencia de Android sobre Permisos en el Manifest](#)  localizando aquellos que tengan indicado **Protection level: normal**.
- ✓ **Permisos de firma:** Permiten el acceso a certificados digitales instalados para firmar digitalmente o identificar a una persona o entidad a través de su cuenta para descargar aplicaciones o acceder a servicios. Puedes buscar aquellos permisos de firma en [esta página de referencia de Android sobre Permisos en el Manifest](#)  localizando aquellos que tengan indicado **Protection level: signature**.

Permisos de tiempo de ejecución (o peligrosos)

Estos permisos abarcan áreas en las cuales la aplicación requiere datos o recursos que incluyen información privada del usuario, o bien que podrían afectar los datos almacenados del usuario o el funcionamiento de otras aplicaciones. Cuando la aplicación solicita un permiso en tiempo de ejecución, el sistema presenta un mensaje de este permiso que debe ser aceptado por el usuario.


Muchos permisos en tiempo de ejecución acceden a los **datos privados del usuario**, un tipo especial de datos restringidos que incluye información que puede ser sensible. Algunos ejemplos de datos privados del usuario incluyen la ubicación y la información de contacto del usuario, la lista de los contactos del usuario, etc.

El micrófono, la cámara y la ubicación brindan acceso a información **particularmente sensible**. Por lo tanto, debes evitar acceder desde tu aplicación a estos dispositivos sin haber obtenido el permiso explícito para ello (permisos CAMERA para la cámara, RECORD_AUDIO para el micro y ACCESS_COARSE_LOCATION o ACCESS_FINE_LOCATION para la localización).

Puedes buscar aquellos permisos peligrosos en [esta página de referencia de Android sobre Permisos en el Manifest](#)  localizando aquellos que tengan indicado **Protection level: dangerous**

Permisos especiales

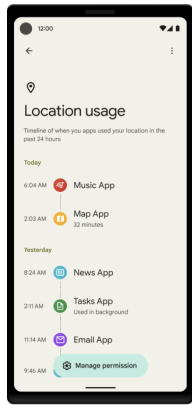
Estos permisos son particularmente confidenciales, por ello, la mayoría de las aplicaciones no deben usarlos. Solo la plataforma (el propio Android) y los OEM (fabricantes de los dispositivos como Samsung, Xiaomi, etc.) pueden definir permisos especiales. Además, la plataforma y los OEM suelen definir permisos especiales cuando quieren proteger el acceso a acciones particularmente importantes, como escribir sobre otras aplicaciones.

Si una aplicación 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. Puedes buscar aquellos permisos especiales en [esta página de referencia de Android sobre Permisos en el Manifest](#)  localizando aquellos que tengan indicado **Protection level: appop**

Debes conocer el Panel de Privacidad

En dispositivos compatibles que ejecutan Android 12 o versiones posteriores, aparece la pantalla del panel de privacidad en la configuración del sistema. En esta pantalla, los usuarios pueden acceder a distintas pantallas que se muestran cuando las aplicaciones acceden a la información de la ubicación, la cámara y el micrófono. En cada una, se muestra un cronograma del momento en que diferentes aplicaciones accedieron a un tipo específico de datos.

Además desde este panel puedes configurar los permisos a todas las aplicaciones, otorgando, quitando, obligando a preguntar, usando en segundo plano, etc.



[Google Developers](#) (Us
educativo nc)

1.4.- Grupos de permisos.


Todos los permisos peligrosos del sistema Android pertenecen a grupos de permisos. Si el dispositivo tiene Android 6.0 (nivel de API 23) instalado o superior y el atributo `targetSdkVersion` de la aplicación es 23 o un valor superior, se producirá el siguiente comportamiento del sistema cuando la aplicación solicita un permiso peligroso:

- ✓ Si una aplicación solicita un permiso peligroso incluido en su manifiesto y no tiene permisos actualmente en el grupo de permisos, el sistema muestra un cuadro de diálogo al usuario en el que se describe el grupo de permisos al cual la aplicación desea acceder. En el cuadro de diálogo no se describe el permiso específico dentro de ese grupo. Por ejemplo, si una aplicación solicita el permiso `READ_CONTACTS`, en el cuadro de diálogo del sistema se indica únicamente que la aplicación necesita acceso a los contactos del dispositivo. Si el usuario brinda la aprobación, el sistema otorga a la aplicación solamente el permiso que solicitó.
- ✓ Si una aplicación solicita un permiso peligroso incluido en su manifiesto y ya tiene otro permiso peligroso en el mismo grupo de permisos, el sistema lo otorga de inmediato sin interacción con el usuario. Por ejemplo, si a una aplicación ya se le otorgó el permiso `READ_CONTACTS` y luego esta solicita el permiso `WRITE_CONTACTS`, el sistema lo otorga de inmediato.

Cualquier permiso puede pertenecer a un grupo de permisos; entre ellos, los normales y los que define tu aplicación. Sin embargo, el grupo de un permiso solo afecta la experiencia del usuario si es peligroso. Puedes ignorar el grupo de permisos para los permisos normales.

Si el dispositivo tiene instalado Android 5.1 (nivel de API 22) o versiones anteriores, o si el atributo `targetSdkVersion` de la aplicación es 22 o inferior, el sistema solicita al usuario que otorgue el permiso en el momento de la instalación. Nuevamente, el sistema solo dice al usuario qué grupos de permisos necesita la aplicación y no los permisos individuales.

Algunos de estos grupos son: `CAMERA`, `CONTACTS`, `LOCATIONS`, `MICROPHONE`, `PHONE`, `NOTIFICATIONS`, `SMS`, `STORAGE`.

Puedes encontrar en este enlace los grupos de permisos de Android: [Página de referencia de Android sobre Grupos de Permisos en el Manifest](#) 

1-5.- Prácticas recomendadas en la gestión de los permisos.

Objetivos de privacidad

Los permisos de la aplicación se basan en las funciones de seguridad del sistema y ayudan a Android a cumplir los siguientes objetivos de la privacidad del usuario:

- ✓ **Control:** El usuario controla los datos que comparte con las aplicaciones.
- ✓ **Transparencia:** El usuario comprende los datos que usa una aplicación y el motivo por el que accede a estos datos.
- ✓ **Minimización de datos:** Una aplicación accede y usa solo los datos necesarios para una tarea o acción específica que el usuario invoca.

Prácticas recomendadas

Developers.android.com recomienda las siguientes prácticas:

- ✓ **Solicita una cantidad mínima de permisos:** Cuando el usuario solicita una acción específica en tu aplicación, esta solo debe solicitar los permisos que necesita para completarla. Según cómo uses los permisos, puede haber una forma alternativa para cumplir con el caso de uso de tu aplicación sin depender del acceso a información sensible.
- ✓ **Asocia permisos de tiempo de ejecución con acciones específicas:** Solicita permisos lo más tarde posible en los flujos de casos de uso de tu aplicación. Por ejemplo, si tu aplicación permite que los usuarios envíen mensajes de audio a otras personas, espera hasta que el usuario navegue a la pantalla de mensajes y presione el botón para enviar un mensaje de audio. Después de que el usuario presione el botón, la aplicación podrá solicitar acceso al micrófono.
- ✓ **Considera las dependencias de tu aplicación:** Cuando incluyes una biblioteca, también heredas sus requisitos de permisos. Ten en cuenta los permisos que requiere cada dependencia y para qué se usan.
- ✓ **Sé transparente:** Cuando solicitas permisos, sé claro con respecto a la información a la que accedes y a la razón por la que lo haces, de manera que los usuarios puedan tomar decisiones fundamentadas.
- ✓ **Crea accesos explícitos al sistema:** Cuando accedes a datos sensibles o al hardware, como la cámara o el micrófono, proporciona una indicación continua en tu aplicación si el sistema todavía no brinda estos indicadores. Este recordatorio ayuda a los usuarios a comprender exactamente cuándo tu aplicación accede a datos restringidos o realiza acciones restringidas.

Autoevaluación

A partir de Android 6.0, API 23 (MarshMallow)...

- ☐ El sistema, tras declarar un permiso en el archivo de manifiesto, siempre solicita al usuario que apruebe la solicitud de dicho permiso.
- ☐ El sistema solicita al usuario que apruebe la solicitud de un permiso, sin declarar dicho permiso en el archivo de manifiesto.
- ☐ El sistema, tras declarar un permiso en el archivo de manifiesto, ya no solicita al usuario que apruebe la solicitud de dicho permiso.
- ☐ El sistema, tras declarar un permiso en el archivo de manifiesto, puede solicitar o no, al usuario, que apruebe la solicitud de dicho permiso, dependiendo de su grado de confidencialidad.

Incorrecto

Incorrecto

Incorrecto

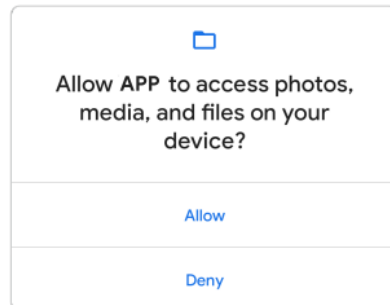
Opción correcta

Solución

1. Incorrecto
2. Incorrecto
3. Incorrecto
4. Opción correcta

2.- Solicitud de permisos en tiempo de ejecución.

Caso práctico



[Google Developers](#) (Uso educativo no)

- Nos ha quedado claro cómo opera Android con los permisos -le indica Juan a Ada-, ahora debemos ser capaces de construir una aplicación que acceda a la cámara del dispositivo para conseguir capturar una imagen.
- Me parece perfecto -responde Ada-. Para ello el primer paso es, como vimos previamente, declarar la necesidad de este permiso en el archivo de manifiesto de la aplicación Android, y el segundo paso es solicitar el permiso necesario al usuario. ¿Qué pueden ocurrir?
- En caso de que no conceda el permiso de forma explícita, la aplicación no deberá acceder a la cámara -responde Juan-. Si se lo concede, la aplicación tendrá acceso a la cámara.

A partir de Android 6.0 (nivel de API 23), los usuarios conceden permisos a las aplicaciones mientras se ejecutan, no cuando instalan la aplicación. Este enfoque simplifica el proceso de instalación de la aplicación, ya que el usuario no necesita conceder permisos cuando instala o actualiza la aplicación. También brinda al usuario mayor control sobre la funcionalidad de la aplicación; por ejemplo, un usuario podría optar por proporcionar a una aplicación de cámara acceso a esta, pero no a la ubicación del dispositivo. El usuario puede revocar los permisos en cualquier momento desde la pantalla de configuración de la aplicación o desde el nuevo Panel de Permisos (disponible desde Android 12).

Cómo hemos visto anteriormente existen distintos tipos de permisos en Android: permisos normales y de firma, peligrosos y especiales.

Los permisos normales y de firma no ponen en riesgo la privacidad del usuario de forma directa. Si tu aplicación tiene un permiso normal en su manifiesto, el sistema concede el permiso automáticamente.

Los permisos peligrosos pueden permitir que la aplicación acceda a información confidencial del usuario. Si tienes un permiso peligroso en el manifiesto, el usuario debe autorizar explícitamente a tu aplicación.

En todas las versiones de Android, nuestra aplicación debe declarar los permisos normales y peligrosos que necesita en su manifiesto, el efecto de esa declaración es diferente según la versión de sistema y el nivel de SDK de destino de nuestra aplicación:

- ✓ Si el dispositivo tiene Android 5.1 o una versión anterior, o el nivel de SDK de destino de tu aplicación es el 22 o uno inferior: Si tienes un permiso peligroso en tu manifiesto, el usuario debe conceder el permiso cuando instale la aplicación; si no otorga el permiso, el sistema no instalará la aplicación.
- ✓ Si el dispositivo tiene Android 6.0 o una versión posterior, y el nivel de SDK de destino de tu aplicación es el 23 o uno posterior: Los permisos deben estar indicados en el manifiesto de la aplicación, y esta debe solicitar cada permiso peligroso que necesite mientras la aplicación esté en ejecución. El usuario puede conceder o negar cada permiso y la aplicación puede continuar ejecutándose con capacidades limitadas aun cuando el usuario rechace una solicitud de permiso.

Desde esta API 23 los usuarios pueden revocar permisos desde cualquier aplicación en cualquier momento, aunque la aplicación esté orientada a un nivel de API inferior. **Debemos probar nuestra aplicación para verificar que se comporte correctamente** cuando no cuente con un permiso necesario, independientemente del nivel de API al que esté orientada nuestra aplicación.

2.1.- Comprobación de existencia de permisos.

Si nuestra aplicación necesita un permiso peligroso, debemos verificar si tenemos ese permiso cada vez que realicemos una operación que lo requiera. El usuario siempre puede revocar el permiso. Por lo tanto, aunque la aplicación haya usado la cámara el día anterior, no podemos dar por hecho que seguirá teniendo ese permiso para el día en curso.

Para comprobar si tenemos un permiso, llamaremos al método **ContextCompat.checkSelfPermission()**.

Ejemplo

En este fragmento se muestra la manera de comprobar si la actividad tiene permiso para realizar operaciones de escritura en el calendario:

```
1 // Assume thisActivity is the current activity
2 int permissionCheck = ContextCompat.checkSelfPermission(thisActivity, Manifest.permission.WRITE_CALENDAR);
```

Si la aplicación tiene el permiso, el método muestra **PackageManager.PERMISSION_GRANTED** y esta puede continuar con la operación. Si la aplicación no tiene el permiso, el método muestra **PERMISSION_DENIED** y la aplicación debe solicitar permiso al usuario de manera explícita.

2.2.- Solicitar permisos.

Si nuestra aplicación necesita un permiso peligroso indicado en el manifiesto, debe solicitar al usuario que lo otorgue. Android ofrece varios métodos que podemos usar para solicitar un permiso. Cuando llamamos a estos métodos, aparece un diálogo de Android estándar que no se puede personalizar.

Si nuestra aplicación todavía no tiene el permiso que necesita, debemos llamar a uno de los métodos **requestPermissions()** para solicitar los permisos correspondientes. Tu aplicación pasa los permisos que necesita y también un *código de solicitud* de entero que tú especificas para identificar esta solicitud de permiso. Este método funciona de manera asíncronica: realiza la devolución inmediatamente y, cuando el usuario responde al cuadro de diálogo, el sistema llama al método *callback* de la aplicación con los resultados y pasa el mismo código de solicitud que la aplicación le pasó a **requestPermissions()**.

Ejemplo

El siguiente código verifica si la aplicación tiene permiso para leer los contactos del usuario y solicita el permiso si es necesario:

```
1 // Here, thisActivity is the current activity
2 if (ContextCompat.checkSelfPermission(thisActivity, Manifest.permission.READ_CONTACTS)
3     != PackageManager.PERMISSION_GRANTED) {
4     // Should we show an explanation?
5     if (ActivityCompat.shouldShowRequestPermissionRationale(thisActivity,
6         Manifest.permission.READ_CONTACTS)) {
7         // Show an explanation to the user *asynchronously* -- don't block
8         // this thread waiting for the user's response! After the user
9         // sees the explanation, try again to request the permission.
10    } else {
11        // No explanation needed, we can request the permission.
12        ActivityCompat.requestPermissions(thisActivity,
13            new String[]{Manifest.permission.READ_CONTACTS},
14            MY_PERMISSIONS_REQUEST_READ_CONTACTS);
15        // MY_PERMISSIONS_REQUEST_READ_CONTACTS is an
16        // app-defined int constant. The callback method gets the
17        // result of the request.
```

```
17     }
18 }
19
```

Cuando nuestra aplicación solicita permisos, el sistema muestra al usuario un cuadro de diálogo. Cuando el usuario responde, el sistema invoca el método **onRequestPermissionsResult()** de nuestra aplicación y le pasa la respuesta del usuario. Nuestra aplicación debe anular ese método para averiguar si se otorgó el permiso. El callback recibe el mismo código de solicitud que le pasaste a **requestPermissions()**.

Ejemplo

Si una aplicación solicita acceso a **READ_CONTACTS**, es posible que tenga el siguiente método callback:

```
1  @Override
2  public void onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults) {
3      switch (requestCode) {
4          case MY_PERMISSIONS_REQUEST_READ_CONTACTS: {
5              // If request is cancelled, the result arrays are empty.
6              if (grantResults.length > 0
7                  && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
8                  // permission was granted, yay! Do the
9                  // contacts-related task you need to do.
10             } else {
11                 // permission denied, boo! Disable the
12                 // functionality that depends on this permission.
13             }
14             return;
15         }
16         // other 'case' lines to check for other
17         // permissions this app might request
18     }
19 }
```

El cuadro de diálogo que muestra el sistema describe el grupo de permisos al que nuestra aplicación necesita acceder, pero no indica el permiso específico. Por ejemplo, si solicitamos el permiso **READ_CONTACTS**, el cuadro de diálogo del sistema simplemente indica que nuestra aplicación necesita acceder a los contactos del dispositivo. El usuario solamente debe otorgar el permiso una vez para cada grupo de permisos. Si nuestra aplicación requiere otros permisos de ese grupo (que se indican en el manifiesto de tu aplicación), el sistema los concede automáticamente. Cuando solicitamos el permiso, el sistema llama a tu método callback **onRequestPermissionsResult()** y pasa **PERMISSION_GRANTED**, de la misma manera en que lo haría si el usuario hubiera aceptado explícitamente nuestra solicitud a través del cuadro de diálogo del sistema.

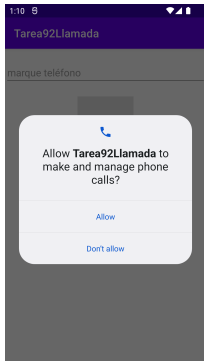
Por ejemplo, supón que indicamos **READ_CONTACTS** y **WRITE_CONTACTS** en el manifiesto de nuestra aplicación. Si solicitamos **READ_CONTACTS**, el usuario concede el permiso y luego solicitamos **WRITE_CONTACTS**, el sistema nos otorga de inmediato ese permiso sin interactuar con el usuario.

Si un usuario rechaza una solicitud de permiso, nuestra aplicación debe tomar una medida adecuada. Por ejemplo, nuestra aplicación podría mostrar un diálogo en el que se explique por qué no podría realizar la acción solicitada por el usuario para la cual se requiere ese permiso.

Cuando el sistema solicita al usuario que otorgue un permiso, el usuario tiene la opción de indicar al sistema que no solicite ese permiso de nuevo. En ese caso, cuando la aplicación use **requestPermissions()** para solicitar ese permiso nuevamente, el sistema rechazará la solicitud de inmediato. El sistema llama a nuestro método callback **PERMISSION_DENIED** y pasa **onRequestPermissionsResult()**, de la misma manera en que lo haría si el usuario hubiera rechazado explícitamente nuestra solicitud una vez más. Esto significa que cuando llamamos a **requestPermissions()**, no podemos suponer que haya existido interacción directa con el usuario.

2.3.- Ejercicio resuelto.

Ejercicio Resuelto



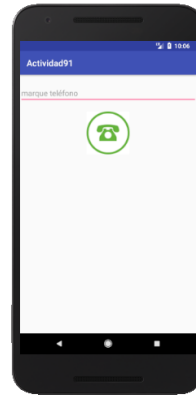
[Google Developers](#) (Uso educativo
nc)

Crear una aplicación que permita realizar una llamada telefónica a un número de teléfono que podrá indicarse a través de un campo de texto.

La aplicación debe implementar un correcto uso del sistema de permisos de Android, de forma que la primera vez que se ejecute la aplicación, debe solicitar al usuario permiso para poder llamar por teléfono ("**android.permission.CALL_PHONE**"), tal y como se muestra en la captura de pantalla inicial.

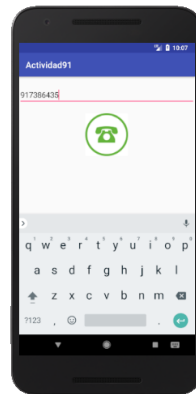
Una vez autorizado, se cargará una interfaz similar a la mostrada en la figura 1 (busca en el sistema o en Internet una imagen apropiada para representar el botón de llamada). Si ahora escribimos un número de teléfono se mostrará una interfaz similar a la mostrada en la figura 2. Simplemente pulsaremos el botón de llamada y ya procederá a realizarse la llamada, tal como se muestra en la figura 3.

Figura 1. Tras conceder los permisos.



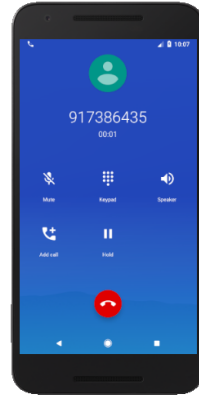
[Google Developers](#) (Uso educativo
nc)

Figura 2. Escribir un nº de tfno.



[Google Developers](#) (Uso educativo
nc)

Figura 3. Llamada realizada.



[Google Developers](#) (Uso educativo
nc)

Mostrar retroalimentación

AndroidManifest.xml

Debemos añadir esta línea al fichero:

```
1 | <uses-permission android:name="android.permission.CALL_PHONE"/>
```

activity_main.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     tools:context="com.cidead.pmdm.tarea921lamada.MainActivity">
8
9     <EditText
10         android:id="@+id/etNumeroTelefono"
11         android:layout_width="match_parent"
12         android:layout_height="wrap_content"
13         android:hint="marque teléfono"
14         android:layout_marginTop="20dp"/>
15
16     <Button
17         android:id="@+id/btnLlamar"
18         android:layout_width="100dp"
19         android:layout_height="100dp"
20         android:layout_below="@id/etNumeroTelefono"
21         android:layout_marginTop="20dp"
22         android:background="@drawable/telefono"
23
24         android:layout_centerHorizontal="true"/>
25
26 </RelativeLayout>
```

MainActivity.java

```
1 package com.cidead.pmdm.tarea921lamada;
2
```



```
3 import androidx.annotation.NonNull;
4 import androidx.appcompat.app.AppCompatActivity;
5 import androidx.core.app.ActivityCompat;
6 import androidx.core.content.ContextCompat;
7
8 import android.Manifest;
9 import android.content.Intent;
10 import android.content.pm.PackageManager;
11 import android.net.Uri;
12 import android.os.Bundle;
13 import android.text.TextUtils;
14 import android.view.View;
15 import android.widget.Button;
16 import android.widget.EditText;
17 import android.widget.Toast;
18
19 public class MainActivity extends AppCompatActivity {
20
21     private static final int MAKE_CALL_PERMISSION_REQUEST_CODE = 1;
22
23     private EditText etNumeroTelefono;
24     private Button btnLlamar;
25
26     @Override
27     protected void onCreate(Bundle savedInstanceState) {
28         super.onCreate(savedInstanceState);
29         setContentView(R.layout.activity_main);
30
31         btnLlamar = (Button) findViewById(R.id.btnLlamar);
32         etNumeroTelefono = (EditText) findViewById(R.id.etNumeroTelefono);
33
34         btnLlamar.setOnClickListener(new View.OnClickListener() {
35             @Override
36             public void onClick(View view) {
37                 String numeroTelefono = etNumeroTelefono.getText().toString();
38
39                 if (!TextUtils.isEmpty(numeroTelefono)) { //Si se ha escrito un número, trataremos de llamarl
40                     if (comprobarPermiso(Manifest.permission.CALL_PHONE)) {
41                         String dial = "tel:" + numeroTelefono; //Se tiene que poner literalmente esto
```

```

42         if (ActivityCompat.checkSelfPermission(getApplicationContext(), Manifest.permission.C
43             // TODO: Consider calling
44             //     ActivityCompat#requestPermissions
45             // here to request the missing permissions, and then overriding
46             //     public void onRequestPermissionsResult(int requestCode, String[] permissions
47             //                                     int[] grantResults)
48             // to handle the case where the user grants the permission. See the documentation
49             // for ActivityCompat#requestPermissions for more details.
50             return;
51         }
52         startActivity(new Intent(Intent.ACTION_CALL, Uri.parse(dial)));
53     } else {
54         Toast.makeText(MainActivity.this, "Permiso de llamada denegado", Toast.LENGTH_SHORT).
55     }
56     } else {
57         Toast.makeText(MainActivity.this, "Escriba número de teléfono", Toast.LENGTH_SHORT).show(
58     }
59     }
60     });
61
62     if (comprobarPermiso(Manifest.permission.CALL_PHONE)) {
63         btnLlamar.setEnabled(true);
64     } else {
65         btnLlamar.setEnabled(false);
66         ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.CALL_PHONE}, MAKE_CALL_P
67     }
68 }
69
70
71 private boolean comprobarPermiso(String permission) {
72     return ContextCompat.checkSelfPermission(this, permission) == PackageManager.PERMISSION_GRANTED;
73 }
74
75 @Override
76 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] gra
77     super.onRequestPermissionsResult(requestCode, permissions, grantResults);
78     switch (requestCode) {
79         case MAKE_CALL_PERMISSION_REQUEST_CODE:
80             if (grantResults.length > 0 && (grantResults[0] == PackageManager.PERMISSION_GRANTED)) {

```

```
81         btnLlamar.setEnabled(true);
82         Toast.makeText(this, "Para llamar al número indicado, pulse el botón de llamada", Toast.L
83     }
84     return;
85 }
86 }
87 }
```

