Permessi

Lezione 11

Permessi

- Le autorizzazioni delle app aiutano a supportare la privacy degli utenti proteggendo l'accesso a:
  - **Restricted data**, come lo stato del sistema e le informazioni di contatto di un utente
  - **Restricted actions**, come la connessione a un dispositivo e la registrazione dell'audio

Tipi di permessi

- Vari tipi di permessi
  - 1 – Install-time permissions
    - Normal permissions
    - Signature permissions
      - Caso particolare!
  - 2 – Runtime permissions
  - Special permissions
    - Caso particolare!
  - 3 – Permission groups

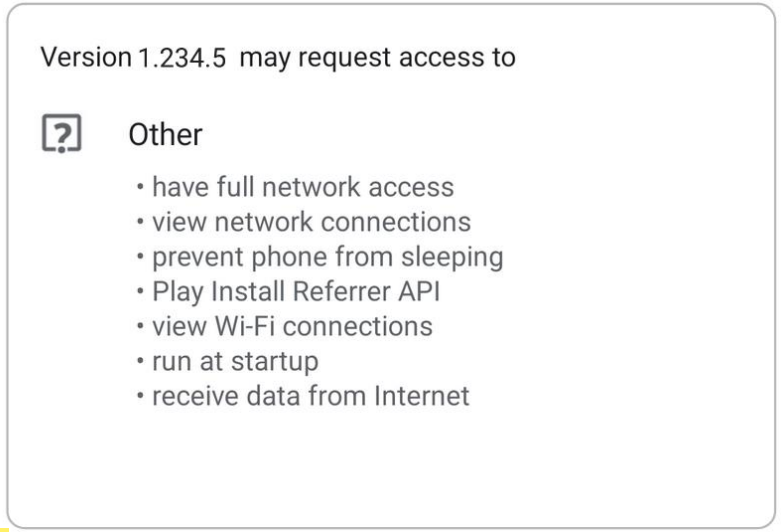
Le install-time permissions vengono concesse automaticamente al momento dell'installazione e restano attive per tutto il ciclo di vita dell'applicazione, senza alcuna possibilità di revoca da parte dell'utente tramite le impostazioni di sistema:
- Quando dichiari un permesso di livello normal o signature nel manifest, il Play Store lo mostra all'utente e poi lo assegna definitivamente all'app non appena questa viene installata.
- Non compaiono nell'elenco dei permessi gestibili da Impostazioni, quindi l'utente non può negarli o revocarli caso per caso.
- L'unico modo "standard" per rimuovere questi permessi è disinstallare l'app (oppure, in scenari avanzati, usare ADB o meccanismi di sicurezza come Play Protect che possono agire su app considerate dannose).

1. Install-Time Permissions
Prima di Android 6.0, tutte le autorizzazioni dichiarate nell'AndroidManifest.xml con protezione "normal" o "signature" venivano concesse automaticamente al momento dell'installazione dell'app, senza alcuna interazione aggiuntiva da parte dell'utente. Questi permessi comportano un rischio minimo e non richiedono runtime prompt.

2. Runtime Permissions
A partire da Android 6.0, le autorizzazioni di tipo "dangerous" (ossia che consentono accesso a dati sensibili o funzioni critiche) devono essere richieste esplicitamente durante l'esecuzione dell'app, non basta la dichiarazione nel manifest. L'utente vede un dialogo di sistema e può revocare o modificare ogni permesso in qualsiasi momento dalle impostazioni.

Install-time permission

- Le autorizzazioni in fase di installazione danno alla tua app un **accesso limitato ai dati restricted** e consentono alla tua app di eseguire azioni restricted che influiscono minimamente sul sistema o su altre app
- Quando dichiari le autorizzazioni al momento dell'installazione nella tua app, il sistema concede automaticamente alla tua app le autorizzazioni quando l'utente installa l'app
  - l'app store presenta un avviso di autorizzazione al momento dell'installazione all'utente quando visualizza la pagina dei dettagli di un'app



Install-time permission

- Fanno parte di questa categoria:
  - **Normal permission**
  - Signature permissions
    - Caso particolare!

https://developer.android.com/reference/android/Manifest.permission

Install-time permission: normal permission

- Queste autorizzazioni consentono l'accesso a dati e azioni che si estendono oltre la sandbox della tua app. Tuttavia, i dati e le azioni presentano un rischio minimo per la privacy dell'utente e per il funzionamento di altre app
- Il sistema assegna il livello di protezione "normale" ai normal permission

NFC

```
public static final String NFC
```

Allows applications to perform I/O operations over NFC.

Protection level: normal

Constant Value: "android.permission.NFC"

INTERNET

```
public static final String INTERNET
```

Allows applications to open network sockets.

Protection level: normal

Constant Value: "android.permission.INTERNET"

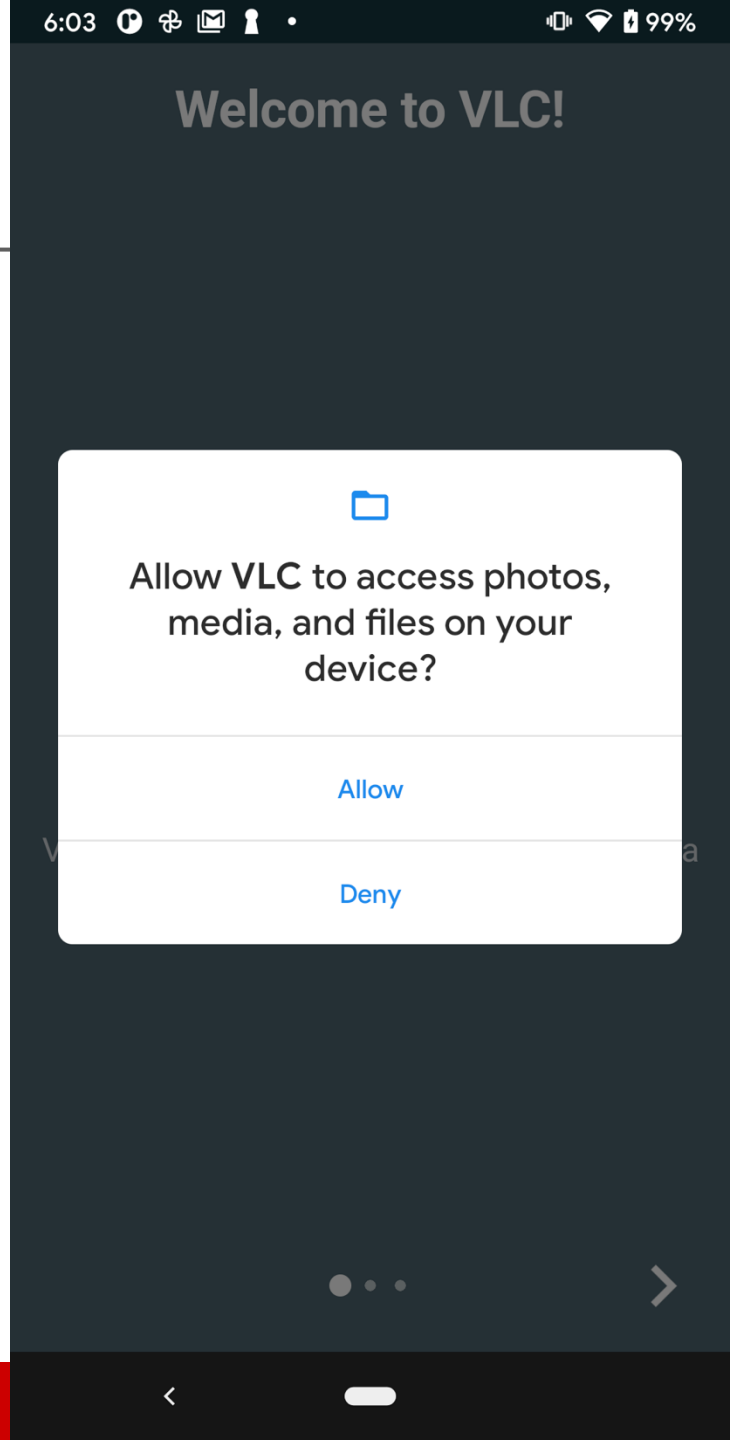
2 Runtime permissions

- Le autorizzazioni di runtime, note anche come autorizzazioni pericolose (dangerous permissions), forniscono alla tua app un accesso aggiuntivo ai dati limitati (restricted) e consentono alla tua app di eseguire azioni limitate (restricted) che influiscono in modo più sostanziale sul sistema e su altre app
  - Pertanto, è necessario richiedere le autorizzazioni runtime nella tua app prima di poter accedere ai dati limitati o eseguire azioni limitate
- Molte autorizzazioni di runtime accedono ai dati degli utenti privati **(private user data),** un tipo speciale di dati limitati che include informazioni potenzialmente sensibili
  - Esempi di dati utente privati includono location e informazioni di contatto.
- Il sistema assegna il livello di protezione "pericoloso" alle autorizzazioni di runtime

2 Runtime permissions

- Quando l'app richiede un'autorizzazione di runtime, il sistema presenta una richiesta di autorizzazione di runtime

- Nota: Il microfono e la camera consentono l'accesso a informazioni particolarmente sensibili!
  - (maggiori dettagli)



3 Permission groups

- Le autorizzazioni possono appartenere a gruppi di autorizzazioni
- I gruppi di autorizzazioni sono costituiti da un insieme di autorizzazioni logicamente correlate
- Ad esempio, le autorizzazioni per l'invio e la ricezione di messaggi SMS potrebbero appartenere allo stesso gruppo, poiché entrambe si riferiscono all'interazione dell'applicazione con gli SMS
- I gruppi di autorizzazioni aiutano il sistema a ridurre al minimo il numero di finestre di dialogo di sistema presentate all'utente quando un'app richiede autorizzazioni strettamente correlate
  - Quando a un utente viene chiesto di concedere le autorizzazioni per un'applicazione, le autorizzazioni appartenenti allo stesso gruppo vengono presentate nella stessa interfaccia
  - Tuttavia, le autorizzazioni possono cambiare gruppo senza preavviso, quindi non dare per scontato che una particolare autorizzazione sia raggruppata con qualsiasi altra autorizzazione

Best practices

- Le autorizzazioni delle app si basano sulle funzionalità di sicurezza del sistema e aiutano Android a supportare i seguenti **obiettivi** relativi alla privacy degli utenti:
  - **Controllo**: l'utente ha il controllo sui dati che condivide con le app
  - **Trasparenza**: l'utente comprende quali dati utilizza un'app e perché l'app accede a questi dati
  - **Data minimization**: un'app accede e utilizza solo i dati necessari per un'attività o un'azione specifica richiamata dall'utente

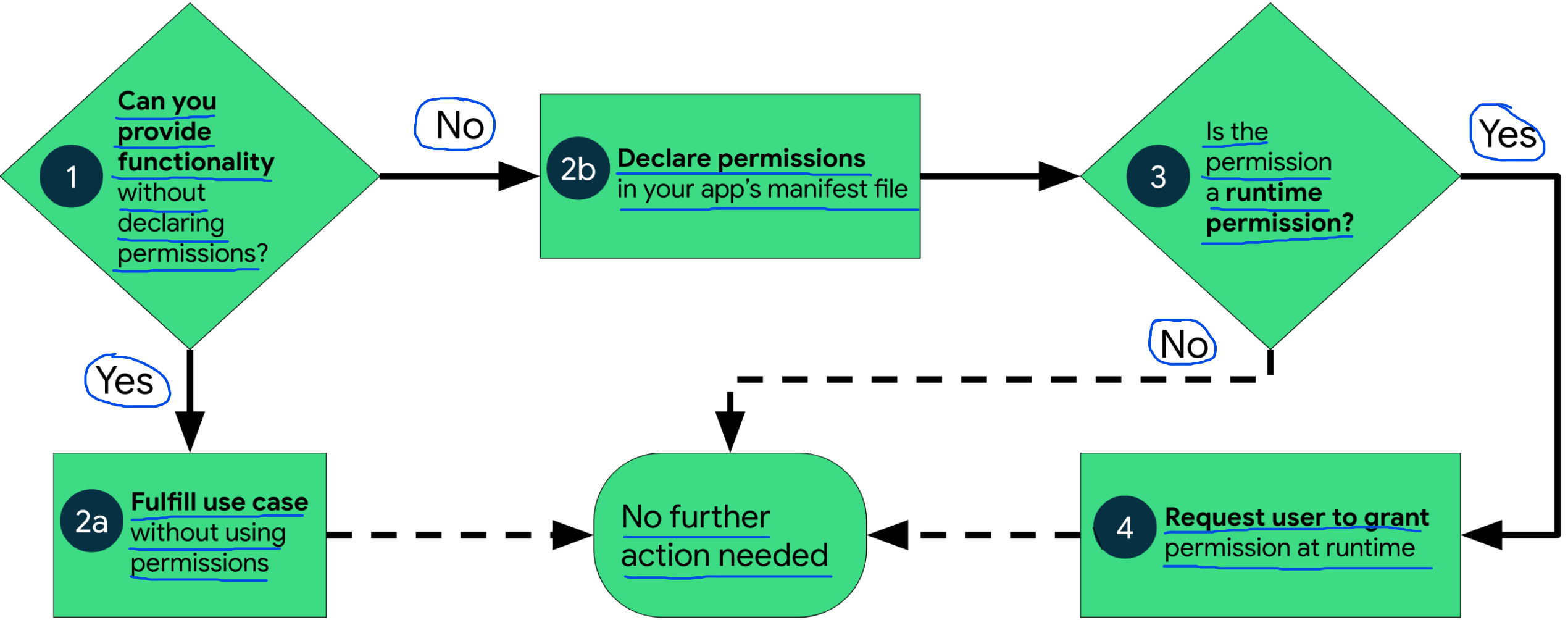
Best practices (1/2)

1. Richiedi un numero minimo di autorizzazioni
   - Quando l'utente richiede una particolare azione nella tua app, la tua app deve richiedere solo le autorizzazioni di cui ha bisogno per completare quell'azione. A seconda di come stai utilizzando le autorizzazioni, potrebbe esserci un modo alternativo per soddisfare il caso d'uso della tua app senza fare affidamento sull'accesso a informazioni sensibili
2. Associa le autorizzazioni di runtime ad azioni specifiche
   - **Richiedi le autorizzazioni il più tardi possibile** nel flusso dei casi d'uso della tua app. Ad esempio, se la tua app consente agli utenti di inviare messaggi audio ad altri, attendi che l'utente sia passato alla schermata dei messaggi e abbia premuto il pulsante Invia messaggio audio. Dopo che l'utente ha premuto il pulsante, la tua app può richiedere l'accesso al microfono.

Best practices (2/2)

- Considera le dipendenze della tua app
  - Quando includi una libreria, ne erediti anche i requisiti di autorizzazione. E' importante essere consapevoli delle autorizzazioni richieste da ciascuna dipendenza e per cosa vengono utilizzate
- Sii trasparente
  - Quando effettui una richiesta di autorizzazione, sii chiaro a cosa stai accedendo e perché, in modo che gli utenti possano prendere decisioni informate
- Rendi espliciti gli accessi al sistema
  - Quando accedi a dati sensibili o hardware, come la fotocamera o il microfono, fornisci un'indicazione continua nella tua app. Questo promemoria aiuta gli utenti a capire esattamente quando la tua app accede a dati limitati o esegue azioni limitate.

Workflow per usare i permessi



Valuta se la tua app deve dichiarare le autorizzazioni

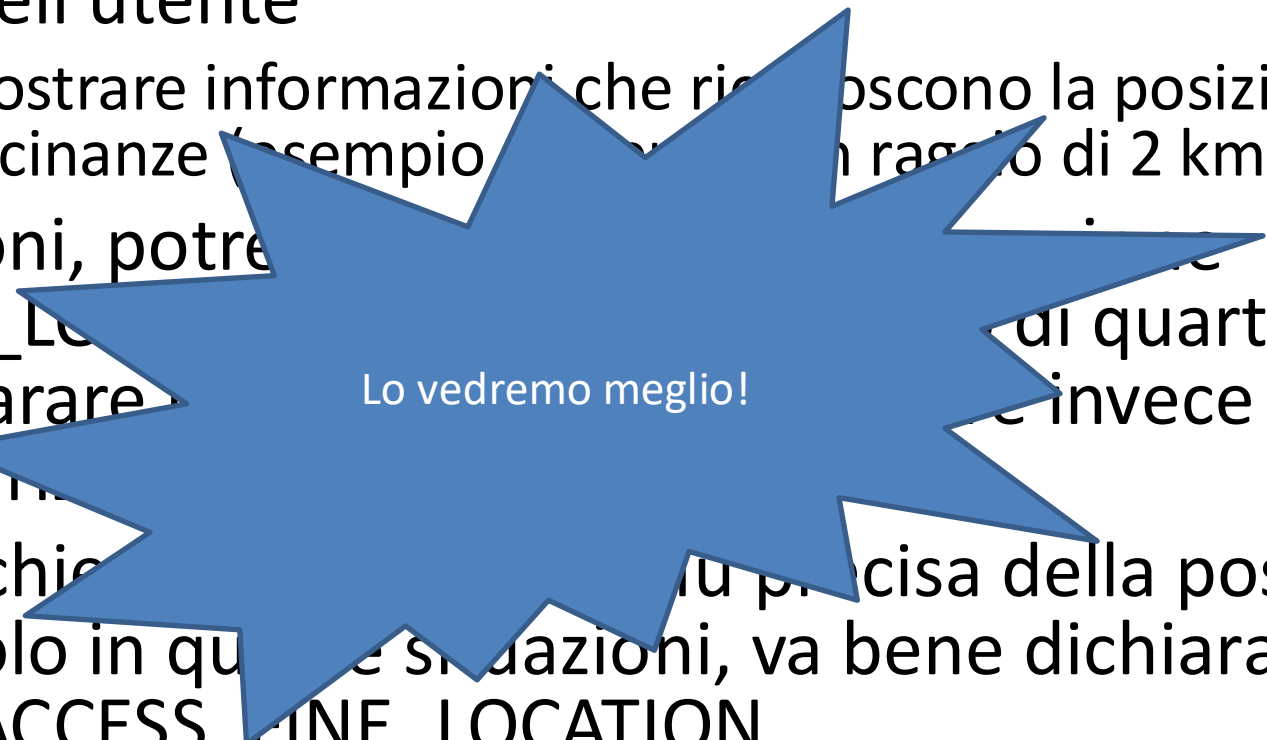
- Prima di dichiarare le autorizzazioni nella tua app, valuta se è necessario farlo
- Ogni volta che l'utente vuole utilizzare una funzionalità dell'app che richiede un'autorizzazione di runtime, l'app deve interrompere il lavoro dell'utente con una richiesta di autorizzazione
  - L'utente deve quindi prendere una decisione. Se l'utente non capisce perché la tua app richiede un'autorizzazione particolare, potrebbe negare l'autorizzazione o addirittura disinstallarla
- Considera anche se un'altra app installata potrebbe essere in grado di eseguire la funzionalità per conto della tua app
  - In questi casi, dovresti delegare l'attività a un'altra app usando un Intent. In tal modo, non è necessario dichiarare le autorizzazioni necessarie perché l'altra app dichiara invece l'autorizzazione

Esempio: Mostra luoghi vicini

- La tua app potrebbe aver bisogno di conoscere la posizione approssimativa dell'utente
  - Ciò è utile per mostrare informazioni che riconoscono la posizione, come i ristoranti nelle vicinanze (es. entro un raggio di 2 km)
- In queste situazioni, potresti dichiarare l'autorizzazione ACCESS_COARSE_LOCATION (precisione a livello di quartiere) ma è meglio non dichiarare l'autorizzazione e chiedere invece all'utente di inserire un indirizzo o un codice postale
- Altri casi d'uso richiedono una stima più precisa della posizione di un dispositivo. Solo in queste situazioni, va bene dichiarare l'autorizzazione ACCESS_FINE_LOCATION

Esempio: Mostra luoghi vicini

- La tua app potrebbe aver bisogno di conoscere la posizione approssimativa dell'utente
  - Ciò è utile per mostrare informazioni che richiedono la posizione, come i ristoranti nelle vicinanze (esempio entro un raggio di 2 km)
- In queste situazioni, potrebbe ... ACCESS_COARSE_LOCATION (di quartiere) ma è meglio non dichiarare ... invece all'utente di inserire un indirizzo
- Altri casi d'uso richiedono una più precisa della posizione di un dispositivo. Solo in queste situazioni, va bene dichiarare l'autorizzazione ACCESS_FINE_LOCATION



Esempio: Scatta una foto

- Gli utenti potrebbero scattare foto nella tua app, utilizzando l'app della fotocamera di sistema preinstallata
- In questa situazione, non dichiarare l'autorizzazione della CAMERA, Richiama invece l'azione di Intent ACTION_IMAGE_CAPTURE

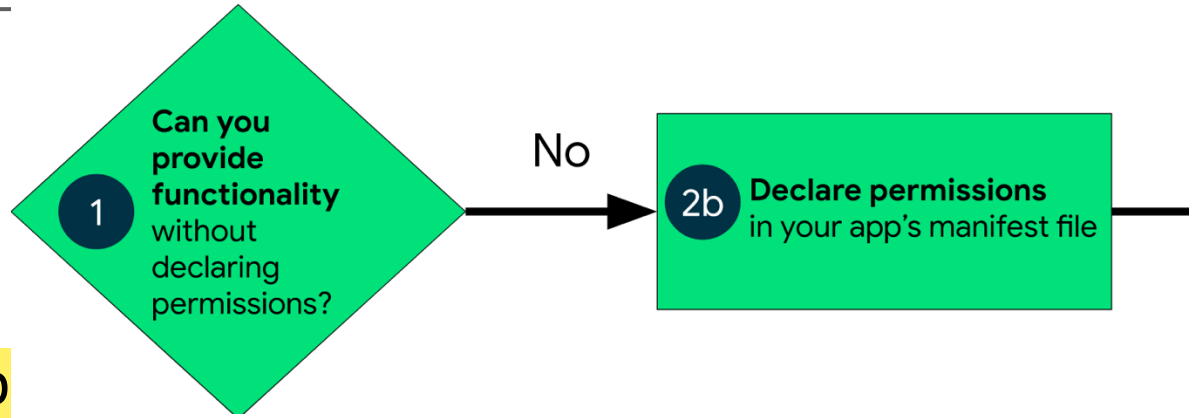
Esempio: Registra un video

- Gli utenti potrebbero registrare video nella tua app, utilizzando l'app della fotocamera di sistema preinstallata
- In questa situazione, non dichiarare l'autorizzazione della CAMERA, MA Richiama invece l'azione di Intent ACTION_VIDEO_CAPTURE

Apri i file multimediali o i documenti di un'altra app

- La tua app potrebbe mostrare contenuti creati da un'altra app, come foto, video o file di testo
- Per supportare questa funzionalità, utilizza il photo picker
  - non richiede alcuna autorizzazione di runtime per essere utilizzato
  - Quando un utente interagisce con il photo picker per selezionare foto o video da condividere con la tua app, il sistema concede l'accesso temporaneo in lettura all'URI associato ai file multimediali selezionati
- Se la tua app deve accedere ai file multimediali senza utilizzare il photo picker, non è necessario dichiarare alcuna autorizzazione di archiviazione
  - Se accedi ai file multimediali creati dalla tua app, la tua app ha già accesso a questi file nel media store
  - Se accedi a file multimediali creati da altre app, utilizza Storage Access Framework

Dichiarare app permission



- Come indicato nel workflow per l'utilizzo delle autorizzazioni, se la tua app richiede autorizzazioni, devi dichiarare queste autorizzazioni nel file manifest della tua app
- Queste dichiarazioni aiutano gli app store e gli utenti a comprendere il set di autorizzazioni che l'app potrebbe richiedere
- Il processo di esecuzione di una richiesta di autorizzazione dipende dal tipo di autorizzazione:
  - Se l'autorizzazione è un'autorizzazione al momento dell'installazione (install-time permissions), come un'autorizzazione normale, l'autorizzazione viene concessa automaticamente al momento dell'installazione
  - Se l'autorizzazione è un'autorizzazione di runtime e se la tua app è installata su un dispositivo che esegue Android 6.0 (livello API 23) o versioni successive, sei tu a dover richiedere l'autorizzazione

Dichiarare app permission: manifest.xml

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

In questo caso, l'app ha bisogno dei permessi per accedere alla camera

Quando dichiari nell'AndroidManifest.xml un <uses-feature>, stai informando sia il sistema Android che il Play Store sulle capacità hardware di cui la tua app può o deve disporre.

Dichiarare hardaware: manifest.xml

<uses-permission> : Dichiarazione di permessi software.
<uses-feature> : Dichiara un requisito hardware o una capacità software.

In questo caso, l'app dichiara che non ha bisogno della camera per funzionare

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

- true : la feature deve essere presente sul dispositivo.
- false : la feature è opzionale: se manca, l'app funziona comunque, semplicemente disabilita le parti che la usano.

Cosa succede se ometti android:required ?
- Default: required="true".

Attenzione: se non imposti android:required su false nella tua dichiarazione <uses-feature>, Android presuppone che l'hardware sia necessario per l'esecuzione dell'app. Il sistema impedisce quindi ad alcuni dispositivi di installare la tua app.

Controllare disponibilità hardware: codice

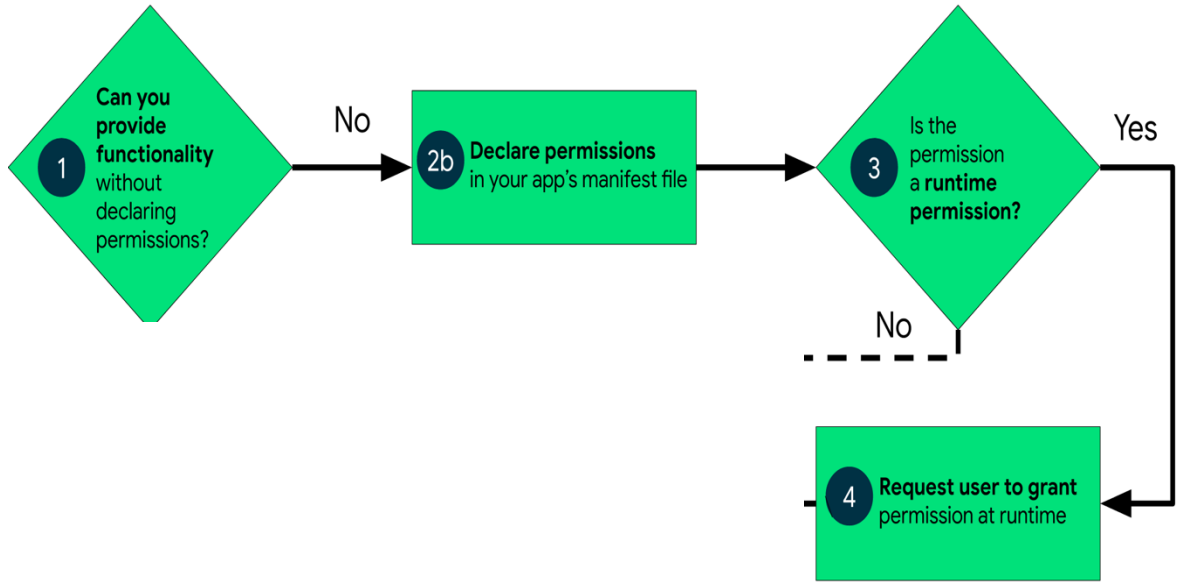
```
// Check whether your app is running on a device that has a front-facing camera.
if (applicationContext.packageManager.hasSystemFeature(
        PackageManager.FEATURE_CAMERA_FRONT)) {
    // Continue with the part of your app's workflow that requires a
    // front-facing camera.
} else {
    // Gracefully degrade your app experience.
}
```

Nota!

- Per dichiarare un'autorizzazione solo sui dispositivi che supportano le autorizzazioni di runtime, ovvero i dispositivi che eseguono Android 6.0 (livello API 23) o versioni successive, includi l'elemento «uses-permission-sdk-23» invece dell'elemento «uses-permission»
- In entrambi i casi è comunque possibile impostare l'attributo maxSdkVersion per indicare che i dispositivi che eseguono una versione di Android superiore al valore specificato non necessitano di un'autorizzazione particolare
- Ciò consente di eliminare le autorizzazioni non necessarie pur garantendo la compatibilità per i dispositivi meno recenti
  - Ad esempio, la tua app potrebbe mostrare contenuti multimediali, come foto o video, che l'utente ha creato mentre si trovava nella tua app. In questa situazione, non è necessario utilizzare l'autorizzazione READ_EXTERNAL_STORAGE sui dispositivi che eseguono Android 10 (livello API 29) o versioni successive, purché la tua app abbia come target Android 10 o versioni successive. Tuttavia, per compatibilità con dispositivi meno recenti, puoi dichiarare l'autorizzazione READ_EXTERNAL_STORAGE e impostare android:maxSdkVersion su 28.

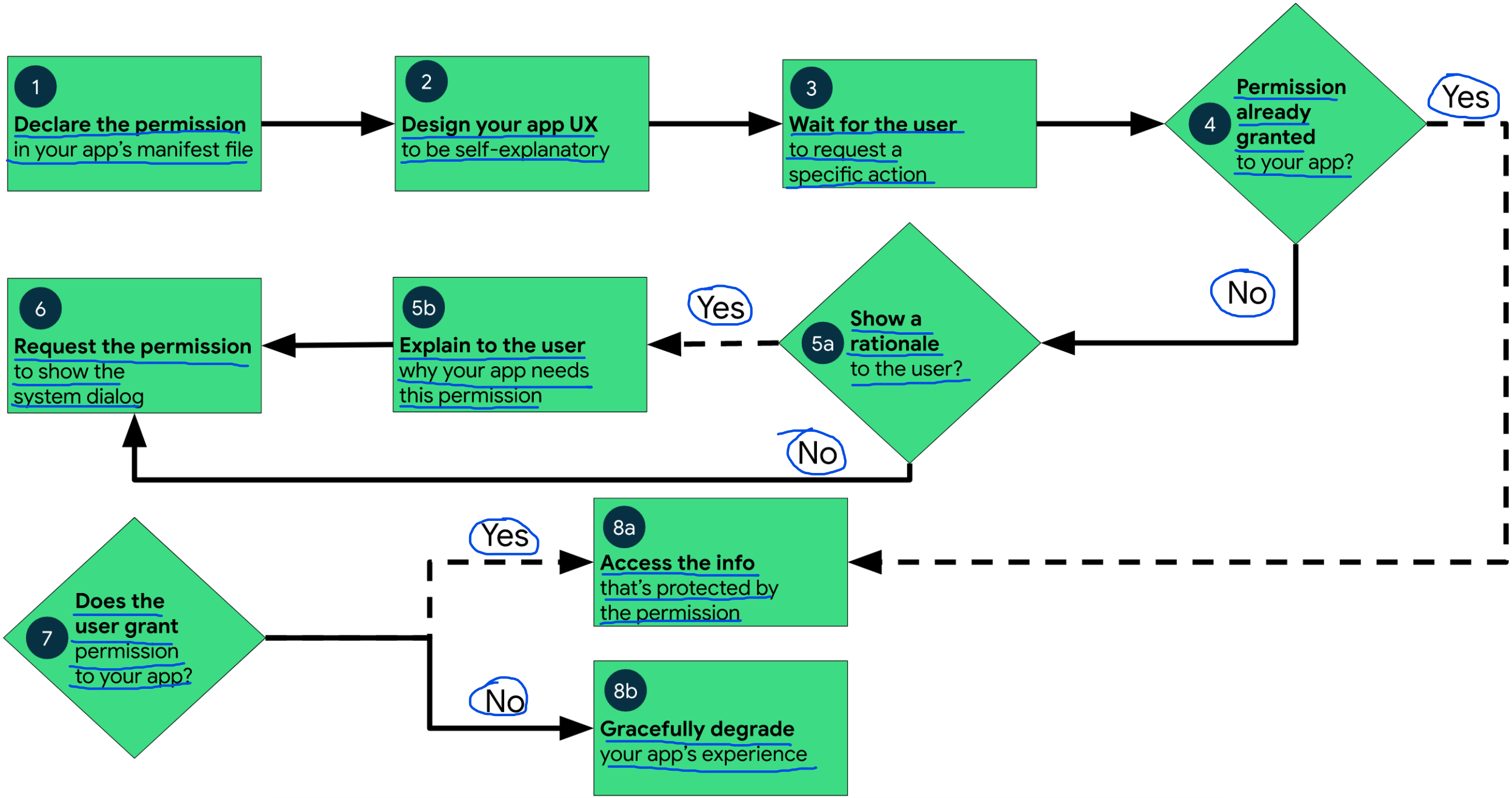
Richiedere App permission

- Ogni app Android viene eseguita in una sandbox ad accesso limitato
- Se la tua app deve usare risorse o informazioni al di fuori della propria sandbox, puoi dichiarare un'autorizzazione e impostare una richiesta di autorizzazione che fornisca questo accesso
- Se dichiari autorizzazioni pericolose (dangerous - runtime) e se la tua app è installata su un dispositivo che esegue Android 6.0 (livello API 23) o versioni successive, devi richiedere le autorizzazioni pericolose in fase di esecuzione (runtime)
    - Se non dichiari autorizzazioni pericolose o se la tua app è installata su un dispositivo che esegue Android 5.1 (livello API 22) o inferiore, le autorizzazioni vengono concesse automaticamente



Principi base

- I principi di base per la richiesta di autorizzazioni in fase di esecuzione (run-time) sono i seguenti:
  1. – Chiedi le autorizzazioni nel contesto corretto, quando l'utente inizia a interagire con la funzionalità che lo richiede
  2. – Non bloccare l'utente. Fornisci sempre la possibilità di annullare un flusso dell'interfaccia utente relativo alle autorizzazioni
  3. – Se l'utente nega o revoca un'autorizzazione di cui una funzionalità ha bisogno, degrada gradualmente la tua app in modo che l'utente possa continuare a utilizzare la tua app, possibilmente disabilitando la funzionalità che richiede l'autorizzazione.
  4. – Non dare per scontato alcun comportamento del sistema. Ad esempio, non dare per scontato che le autorizzazioni vengano visualizzate nello stesso gruppo di autorizzazioni. Un gruppo di autorizzazioni aiuta semplicemente il sistema a ridurre al minimo il numero di finestre di dialogo di sistema presentate all'utente quando un'app richiede autorizzazioni strettamente correlate


1
Declare the permission
in your app's manifest file
2
Design your app UX
to be self-explanatory
3
Wait for the user
to request a
specific action
4
Permission
already
granted
to your app?
Yes
No
5a
Show a
rationale
to the user?
Yes
5b
Explain to the user
why your app needs
this permission
6
Request the permission
to show the
system dialog
No
7
Does the
user grant
permission
to your app?
Yes
8a
Access the info
that's protected by
the permission
No
8b
Gracefully degrade
your app's experience

Determinare se alla tua app è già stata concessa l'autorizzazione

- Per verificare se l'utente ha già concesso alla tua app un'autorizzazione particolare, passa tale autorizzazione al metodo ContextCompat.checkSelfPermission()
- Questo metodo restituisce PERMISSION_GRANTED o PERMISSION_DENIED, a seconda che l'app disponga dell'autorizzazione

Spiega perché la tua app necessita dell'autorizzazione

- La finestra di dialogo dei permessi appare quando si chiama **requestPermissions()** che dice di che permessi avresti bisogno ma non dice perché
- Quindi è una buona idea spiegare all'utente perché servono quei permessi PRIMA di chiamare requestPermissions()
- Ricerche mostrano che gli utenti sono molto più a loro agio con le richieste di autorizzazione se sanno perché l'app ne ha bisogno
  - ad esempio se l'autorizzazione è necessaria per supportare una funzionalità principale dell'app o per la pubblicità
- Di conseguenza, se utilizzi solo una frazione delle chiamate API che rientrano in un gruppo di autorizzazioni (permission group), è utile elencare in modo esplicito quali di tali autorizzazioni stai utilizzando e perché.
  - Ad esempio, se utilizzi solo la geolocalizzazione approssimativa (coarse location), informa l'utente nella descrizione dell'app o negli articoli della guida sulla tua app.
- In determinate condizioni, è anche utile informare gli utenti sull'accesso ai dati sensibili in tempo reale
  - Ad esempio, se stai accedendo alla videocamera o al microfono, è una buona idea informare l'utente utilizzando un'icona di notifica da qualche parte nell'app o nella barra delle notifiche (se l'applicazione è in esecuzione in background), in modo che non sembra che tu stia raccogliendo dati di nascosto

Nota!

- Nota: a partire da Android 12 (livello API 31), gli indicatori di privacy avvisano l'utente ogni volta che le applicazioni accedono al microfono o alla fotocamera

Spiega perché la tua app necessita dell'autorizzazione

- Se il metodo ContextCompat.checkSelfPermission() restituisce PERMISSION_DENIED, chiamare shouldShowRequestPermissionRationale()

  Significa che hai già chiesto quel permesso in precedenza e l'utente l'ha negato almeno una volta (ma non ha scelto "Non chiedere più").

- Se questo metodo restituisce true, mostra all'utente un'interfaccia utente «didattica»
  - In questa interfaccia utente, descrivere il motivo per cui la funzione, che l'utente desidera abilitare, necessita di un'autorizzazione particolare

    mostra un'interfaccia "didattica" che spieghi chiaramente perché quel permesso è necessario per la funzionalità che l'utente sta cercando di usare.
  - NOTA: Restituisce true se l'app ha richiesto questa autorizzazione in precedenza e l'utente ha negato la richiesta.

Richiedere i permessi

- Dopo che l'utente ha visualizzato un'interfaccia utente didattica o il valore restituito di shouldShowRequestPermissionRationale() indica che non è necessario mostrare un'interfaccia utente didattica, è il momento di richiedere l'autorizzazione!
- Gli utenti visualizzano una finestra di dialogo di autorizzazione di sistema, in cui possono scegliere se concedere una particolare autorizzazione alla tua app
- Per fare ciò, utilizza RequestPermission (libreria AndroidX), dove consenti al sistema di gestire il codice di richiesta di autorizzazione per te (Caso 1)
- Poiché l'utilizzo di RequestPermission semplifica la logica, è la soluzione consigliata quando possibile
  - Tuttavia, se necessario, puoi anche gestire tu stesso un codice di richiesta come parte della richiesta di autorizzazione e includere questo codice di richiesta nella logica di richiamata dell'autorizzazione (più dettagli: https://developer.android.com/training/permissions/requesting#manage-request-code-yourself)

Caso 1: Consenti al sistema di gestire il codice di richiesta di autorizzazione per te

- È quindi possibile utilizzare una delle seguenti classi:
  - Per richiedere un'unica autorizzazione, utilizzare **RequestPermission**
  - Per richiedere più autorizzazioni contemporaneamente, utilizzare RequestMultiplePermissions
- Vediamo RequestPermission
  - Nella logica di inizializzazione dell'activity, passare un'implementazione di ActivityResultCallback in una chiamata a *registerForActivityResult*()
  - ActivityResultCallback definisce il modo in cui l'app gestisce la risposta dell'utente alla richiesta di autorizzazione. Tieni un riferimento al valore restituito di *registerForActivityResult*(), che è di tipo ActivityResultLauncher
  - Per visualizzare la finestra di dialogo delle autorizzazioni di sistema quando necessario, chiama il metodo launch() sull'istanza di ActivityResultLauncher che hai salvato nel passaggio precedente. Dopo la chiamata a launch(), viene visualizzata la finestra di dialogo delle autorizzazioni di sistema. Quando l'utente effettua una scelta, il sistema richiama in modo asincrono la tua implementazione di ActivityResultCallback, che hai definito nel passaggio precedente

Nota

- La tua app non può personalizzare la finestra di dialogo che appare quando chiami launch()
- Per fornire più informazioni o il contesto all'utente, modifica l'interfaccia utente della tua app in modo che sia più facile per gli utenti capire perché una funzionalità nella tua app necessita di un'autorizzazione particolare
  - Ad esempio, potresti modificare il testo nel pulsante che abilita la funzione
- Inoltre, il testo nella finestra di dialogo delle autorizzazioni di sistema fa riferimento al gruppo di autorizzazioni associato all'autorizzazione richiesta
  - Questo raggruppamento di autorizzazioni è progettato per la facilità d'uso del sistema e la tua app non deve basarsi su autorizzazioni all'interno o all'esterno di un gruppo di autorizzazioni specifico

Gestire la risposta

```
// Register the permissions callback, which handles the user's response to the
// system permissions dialog. Save the return value, an instance of
// ActivityResultLauncher. You can use either a val, as shown in this snippet,
// or a lateinit var in your onAttach() or onCreate() method.

val requestPermissionLauncher =
    registerForActivityResult(RequestPermission()
    ) { isGranted: Boolean ->
        if (isGranted) {
            // Permission is granted. Continue the action or workflow in your
            // app.
        } else {
            // Explain to the user that the feature is unavailable because the
            // feature requires a permission that the user has denied. At the
            // same time, respect the user's decision. Don't link to system
            // settings in an effort to convince the user to change their
            // decision.
        }
    }
```

Controllare i permessi

```
when {
    ContextCompat.checkSelfPermission(
            CONTEXT,  Manifest.permission.REQUESTED_PERMISSION) == PackageManager.PERMISSION_GRANTED -> {
        // You can use the API that requires the permission.
    }
    shouldShowRequestPermissionRationale(...) -> {
        // In an educational UI, explain to the user why your app requires this
        // permission for a specific feature to behave as expected, and what
        // features are disabled if it's declined. In this UI, include a
        // "cancel" or "no thanks" button that lets the user continue
        // using your app without granting the permission.
        showInContextUI(...)
    }
    else -> {
        // You can directly ask for the permission.
        // The registered ActivityResultCallback gets the result of this request.
        requestPermissionLauncher.launch(
                Manifest.permission.REQUESTED_PERMISSION)
    }
}
```

Ritorna true solo se in precedenza hai già chiamato la richiesta di permesso e l'utente l'ha negata (ma non ha selezionato "Non chiedere più").

Dove CONTEXT e REQUESTED_PERMISSION vanno sostituiti con i corretti valori!

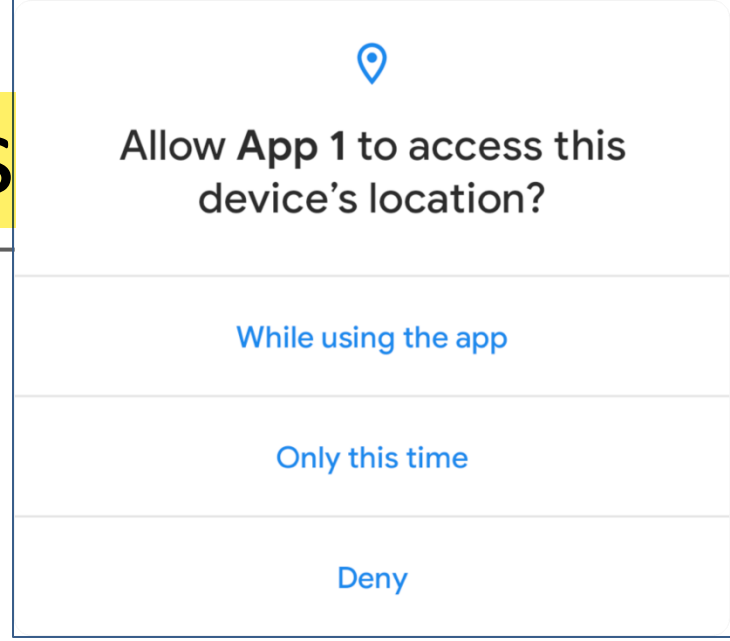
Gestione rifiuto del permesso dell'utente

- Se l'utente nega una richiesta di autorizzazione, l'app dovrebbe aiutare gli utenti a comprendere le implicazioni del negare l'autorizzazione (ma, allo stesso tempo, «rispettare» la decisione dell'utente di negare un'autorizzazione)
- In particolare, la tua app dovrebbe **rendere gli utenti consapevoli delle funzionalità** che non funzionano a causa dell'autorizzazione mancante
- Quando lo fai, tieni a mente le seguenti best practice:
  - Guida l'attenzione dell'utente. Evidenzia una parte specifica dell'interfaccia utente della tua app in cui sono presenti funzionalità limitate perché l'app non dispone dell'autorizzazione necessaria. Diversi esempi di ciò che potresti fare includono quanto segue:
    - Mostra un messaggio in cui sarebbero stati visualizzati i risultati o i dati della funzione
    - Visualizza un pulsante diverso che contiene un'icona e un colore di errore
  - Sii specifico. Non visualizzare un messaggio generico; menziona invece quali funzionalità non sono disponibili perché la tua app non dispone dell'autorizzazione necessaria.
  - Non bloccare l'interfaccia utente. In altre parole, non visualizzare un messaggio di avviso a schermo intero che impedisce agli utenti di continuare a utilizzare la tua app.

Gestione rifiuto del permesso dell’utente

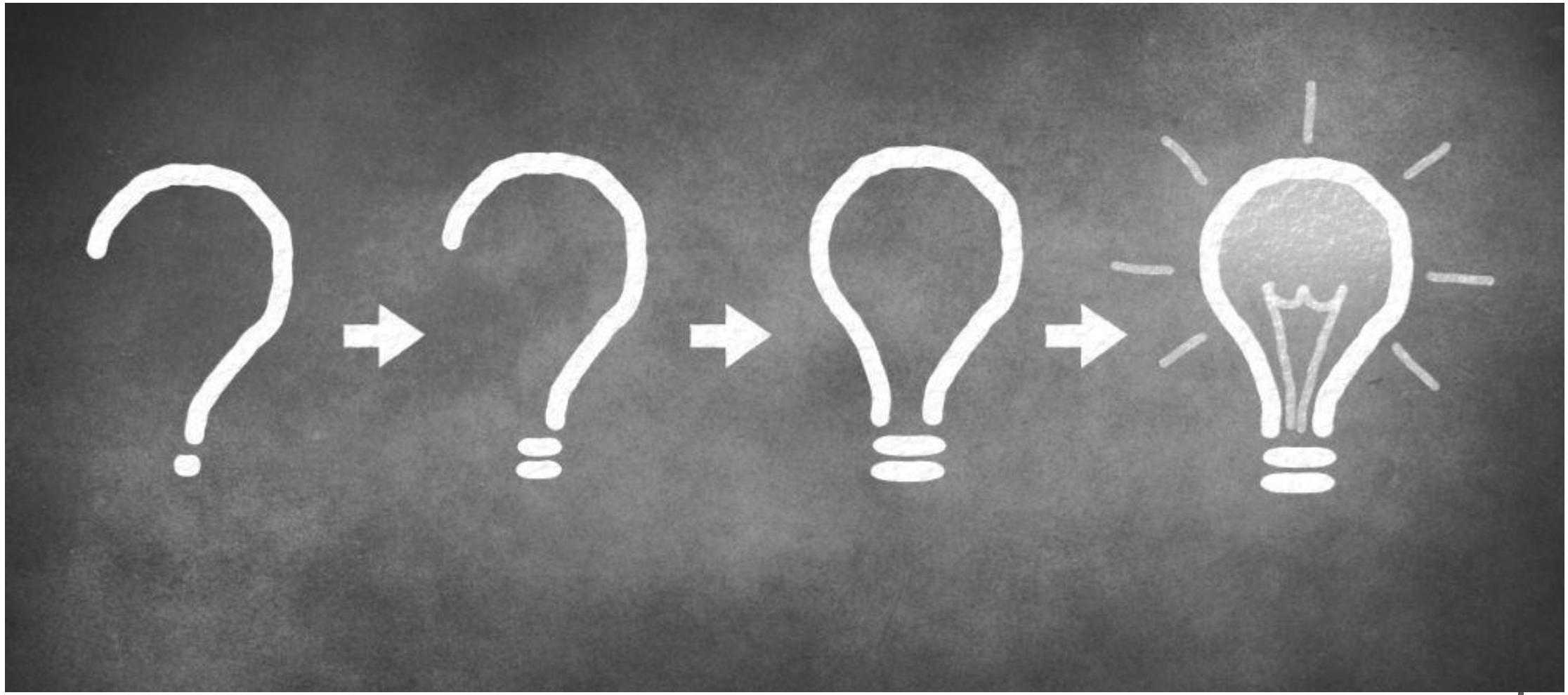
- NB: A partire da **Android 11** (livello API 30), se l'utente seleziona «Nega» per un'autorizzazione specifica più di una volta, l'utente non visualizzerà più la finestra di dialogo delle autorizzazioni di sistema se l'app richiede nuovamente tale autorizzazione
- L'azione dell'utente implica "non chiedere più". Nelle versioni precedenti, gli utenti vedevano la finestra di dialogo delle autorizzazioni di sistema ogni volta che l'app richiedeva un'autorizzazione, a meno che l'utente non avesse precedentemente selezionato una casella di controllo o un'opzione "non chiedere più»
- In alcune situazioni, l'autorizzazione potrebbe essere negata automaticamente, senza che l'utente intraprenda alcuna azione. (Allo stesso modo, anche un'autorizzazione potrebbe essere concessa automaticamente.) È importante non dare per scontato nulla sul comportamento automatico. Ogni volta che la tua app deve accedere a funzionalità che richiedono un'autorizzazione, dovresti verificare che alla tua app sia ancora concessa tale autorizzazione

One-time permissions



- A partire da **Android 11** (livello API 30), ogni volta che l'app richiede un'autorizzazione relativa alla posizione, al microfono o alla fotocamera, la finestra di dialogo delle autorizzazioni rivolte all'utente contiene un'opzione chiamata «Solo questa volta»
- Se l'utente seleziona questa opzione nella finestra di dialogo, alla tua app viene concessa un'autorizzazione temporanea una tantum
- La tua app può quindi accedere ai dati correlati per un periodo di tempo che dipende dal comportamento della tua app e dalle azioni dell'utente:
  - Mentre l'attività della tua app è visibile, la tua app può accedere ai dati.
  - Se l'utente invia la tua app in background, la tua app può continuare ad accedere ai dati per un breve periodo di tempo
  - Se avvii un servizio in primo piano mentre l'attività è visibile e l'utente sposta l'app in background, la tua app può continuare ad accedere ai dati fino a quando il servizio in primo piano non viene interrotto
  - Se l'utente revoca l'autorizzazione una tantum, ad esempio nelle impostazioni di sistema, l'app non può accedere ai dati, indipendentemente dal fatto che tu abbia avviato un servizio in primo piano. Come con qualsiasi autorizzazione, se l'utente revoca l'autorizzazione una tantum della tua app, il processo della tua app termina.
- Quando l'utente apre la tua app e una funzionalità nella tua app richiede l'accesso alla posizione, al microfono o alla fotocamera, all'utente viene nuovamente richiesta l'autorizzazione

Domande???



Riferimenti

- https://developer.android.com/guide/topics/permissions/overview
- https://developer.android.com/training/permissions/evaluating
- https://developer.android.com/training/permissions/declaring
- https://developer.android.com/training/permissions/requesting