Les Notification en Android

Abdelhak-Djamel Seriai

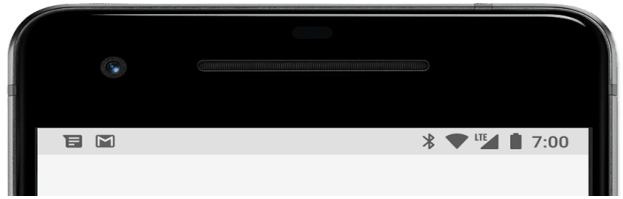
Qu’est Ce Qu’une Notification ?

- Une notification est un message qu'Android affiche en dehors de l'UI de l’ application pour transmettre à l'utilisateur :
  - Des rappels,
  - Des messages transmis par d'autres utilisateurs,
  - Des informations en temps réel provenant de l’application elle-même.

- L'utilisateur peut appuyer sur la notification pour ouvrir l’application associée à la notification ou effectuer une action directement depuis la notification.

- Les notifications apparaissent à différents endroits et sous différents formats.
  - Par exemple, elles peuvent s'afficher automatiquement sous la forme :
    - D'une icône dans la barre d'état,
    - D'un élément plus détaillé dans le panneau des notifications,
    - D'un badge sur l'icône de l'application, ainsi que sur les accessoires connectés associés.

- Exemples de code de notifications : https://github.com/android/user-interface-samples/tree/master/Notifications

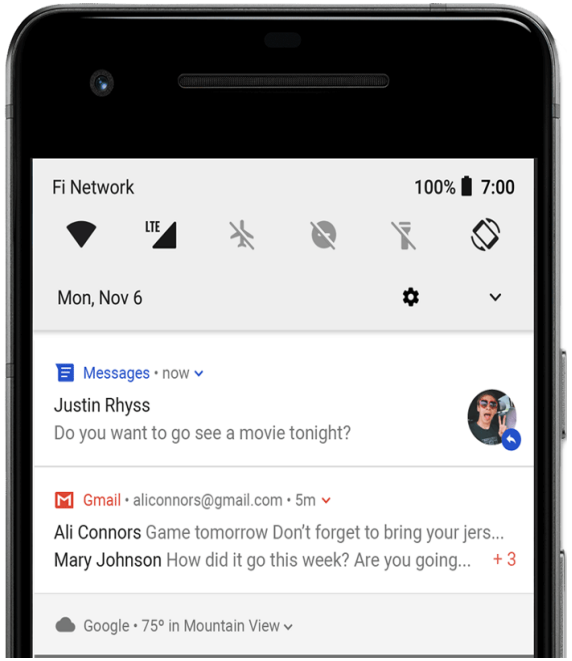


Icônes de notification s'affichent à gauche dans la barre d'état

Affichage des Notifications

- Balayer la barre d'état vers le bas pour ouvrir le panneau des notifications pour :
  - Obtenir plus de détails.
  - Effectuer des actions depuis la notification.

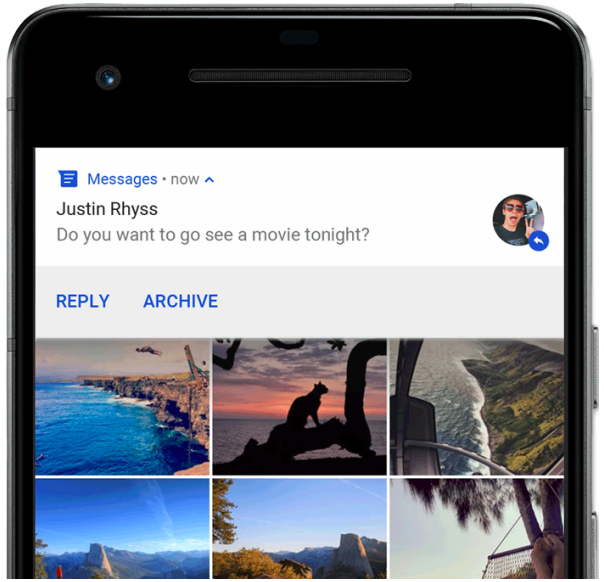
- Faire glisser vers le bas une notification du panneau pour afficher la vue développée donne accès à plus de contenu ainsi qu'à des boutons d'action, le cas échéant.
- Une notification reste visible dans le panneau des notifications jusqu'à ce qu'elle soit ignorée par l'application ou par l'utilisateur.



Notifications dans le panneau des notifications

Affichage des Notifications : Avertissement

- Sous Android 5.0 et versions ultérieures, les notifications peuvent apparaître brièvement dans une fenêtre flottante appelée *avertissement*.
  - Ce type d'affichage est normalement utilisé pour les notifications importantes dont l'utilisateur doit être informé immédiatement, et apparaît uniquement si l'appareil est déverrouillé.
  - L'avertissement apparaît au moment où l'application émet la notification.
  - Il disparaît au bout d'un moment, mais reste visible dans le panneau des notifications.

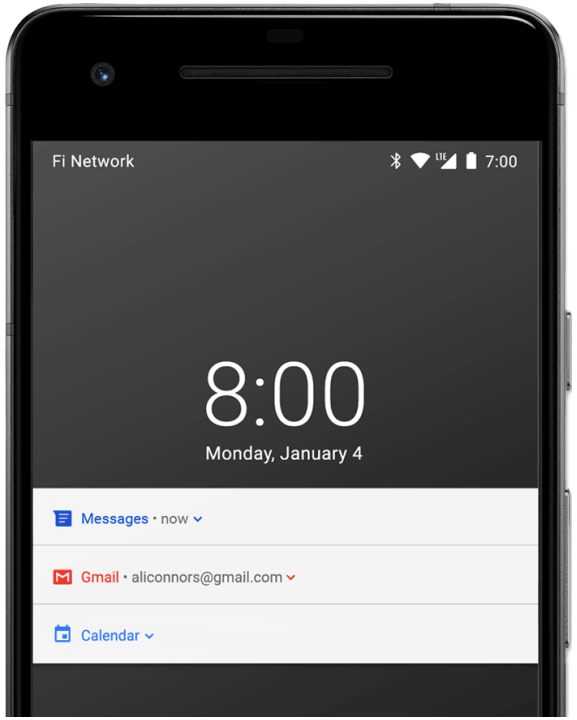


Un avertissement s'affiche devant l'application au premier plan

- Exemples de conditions susceptibles de déclencher des avertissements :
  - L'activité de l'utilisateur est en mode plein écran (l'application utilise *fullScreenIntent*).
  - La notification a un niveau de priorité élevé et utilise des sonneries ou des vibrations sur des appareils fonctionnant sous Android 7.1 (niveau d'API 25) ou une version antérieure.
  - Le canal de la notification est d'importance élevée sur des appareils fonctionnant sous Android 8.0 (niveau d'API 26) ou une version ultérieure.

Affichage des Notifications : Écran de Verrouillage

- Sous Android 5.0 et versions ultérieures, les notifications peuvent s'afficher sur l'écran de verrouillage. Possibilité de :
  - Configurer de manière automatisée le niveau de détail visible dans les notifications émises par l'application sur un écran de verrouillage sécurisé.
  - Définir si la notification s'affiche ou non sur l'écran de verrouillage.
  - Modifier les paramètres système pour choisir le niveau de détail visible dans les notifications sur l'écran de verrouillage.
  - Désactiver toutes les notifications sur l'écran de verrouillage.
- Sous Android 8.0 et versions ultérieures, il est possible de choisir de désactiver ou d'activer les notifications sur l'écran de verrouillage pour chaque canal de notification.

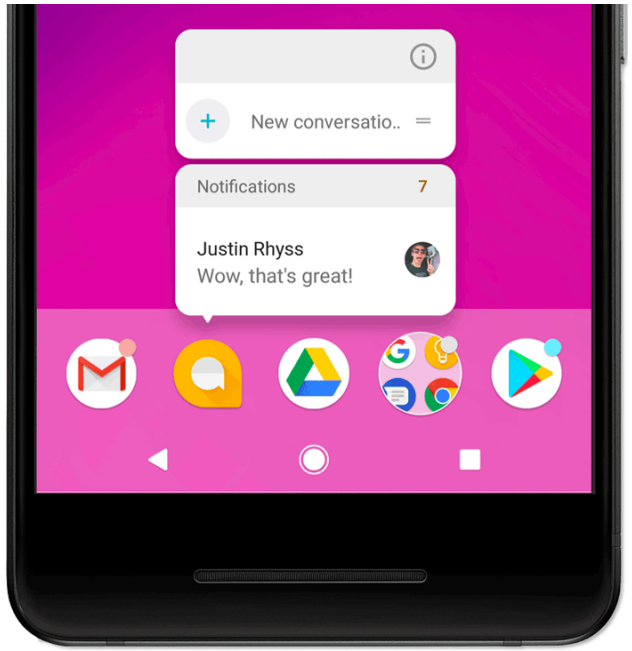


Notifications sur l'écran de verrouillage avec du contenu sensible masqué

Affichage des Notifications : Badge sur l’Icône de l’Application

- Sous Android 8.0 (niveau d'API 26) ou version ultérieure, les icônes d'application signalent la présence de nouvelles notifications par un "badge" coloré (également appelé "pastille de notification").

- Appuyer de manière prolongée sur l'icône d'une application pour afficher les notifications relatives à cette application.

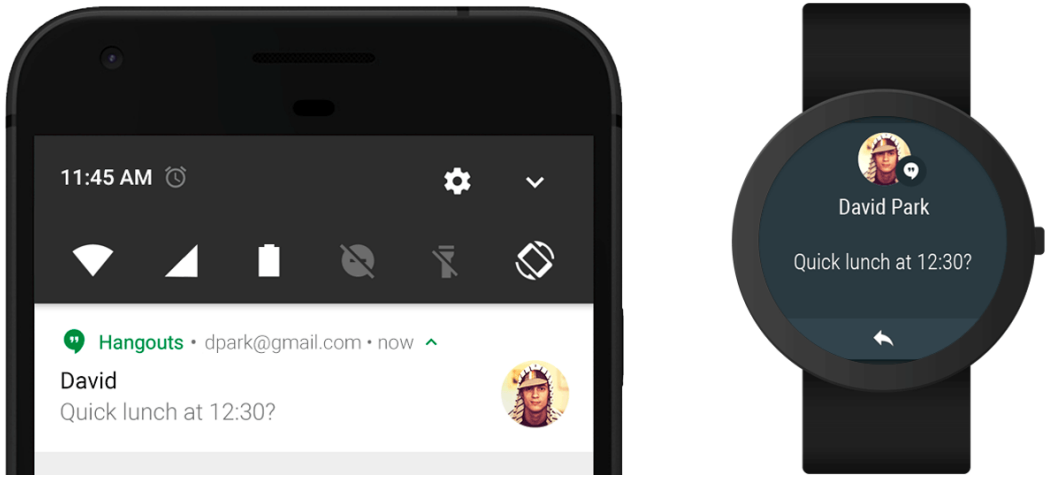


Badges de notification et menu accessible en appuyant de manière prolongée

Affichage des Notifications : Appareils Wear OS

- Sur un appareil Wear OS associé, les notifications s'affichent sur celui-ci automatiquement, y compris les détails à développer et les boutons d'action.

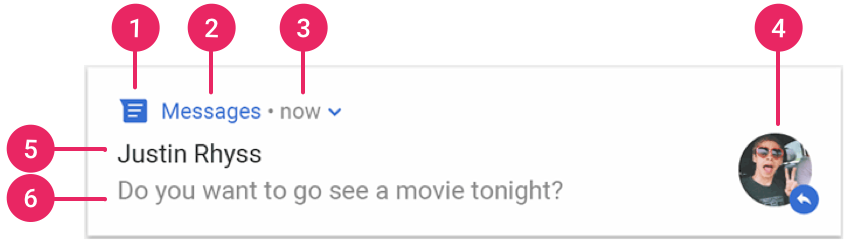
- Possibilité de personnaliser certaines apparences de la notification sur les accessoires connectés



Les notifications s'affichent automatiquement sur un appareil Wear OS associé

Anatomie d'une Notification

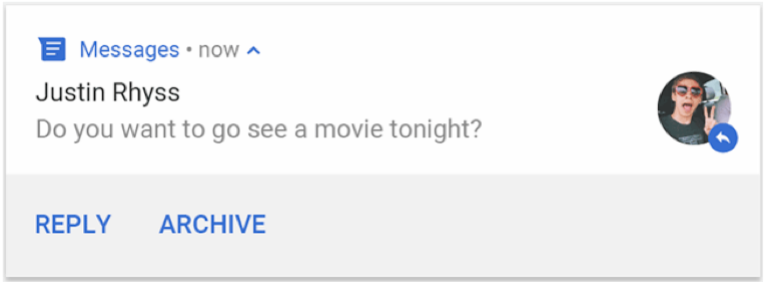
- L'aspect des notifications dépend des modèles du système.
- L'application définit que le contenu de chaque partie du modèle.
- Certains détails de la notification ne sont visibles que dans la vue développée.
- Les éléments les plus courants d'une notification sont :



1. *Petite icône :* obligatoire et définie à l'aide de setSmallIcon()
2. *Nom de l'application :* fourni par le système
3. *Horodatage :* fourni par le système, peut être remplacé par setWhen() ou le masquer avec setShowWhen(false)
4. *Grande icône* : facultative (généralement utilisée uniquement pour les photos de contact ; définie à l'aide de setLargeIcon()
5. *Titre :* facultatif et défini à l'aide de setContentTitle()
6. *Texte :* facultatif et défini à l'aide de setContentText()

Actions de Notification

- Chaque notification peut ouvrir une activité correspondante (facultative) dans l'application lorsque l'utilisateur appuie dessus.
- Il est possible d’ajouter des boutons d'action permettant d'effectuer une tâche liée à l'application depuis la notification.

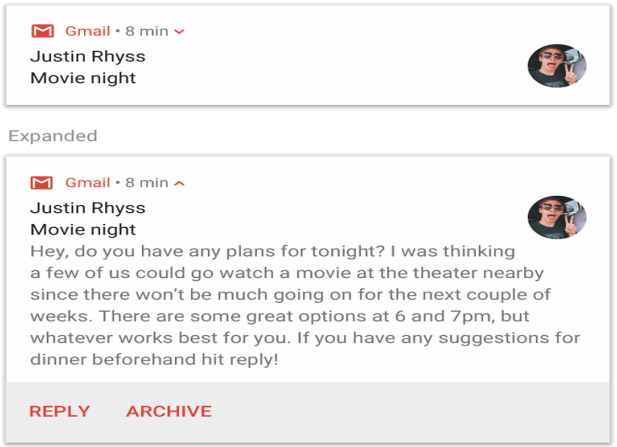


Une notification comportant des boutons d'action

- Sous Android 7.0 (niveau d'API 24) et versions ultérieures, il est possible d’ajouter une action permettant de répondre aux messages ou de saisir un autre texte directement depuis la notification.

Notification à Développer

- Par défaut, le contenu textuel de la notification est tronqué pour tenir sur une seule ligne.
- Pour que la notification soit plus longue, il faut activer une zone de texte plus importante par application d'un modèle supplémentaire.

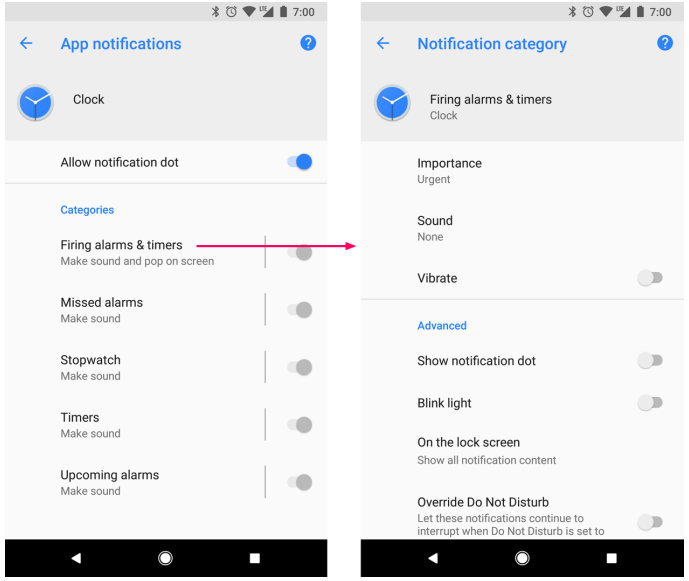


Une notification à développer pour les textes longs

- Il est possible de créer une notification à développer contenant une image, avec un affichage de type boîte de réception, une discussion par chat ou des commandes de lecture multimédia.

Canaux de Notification

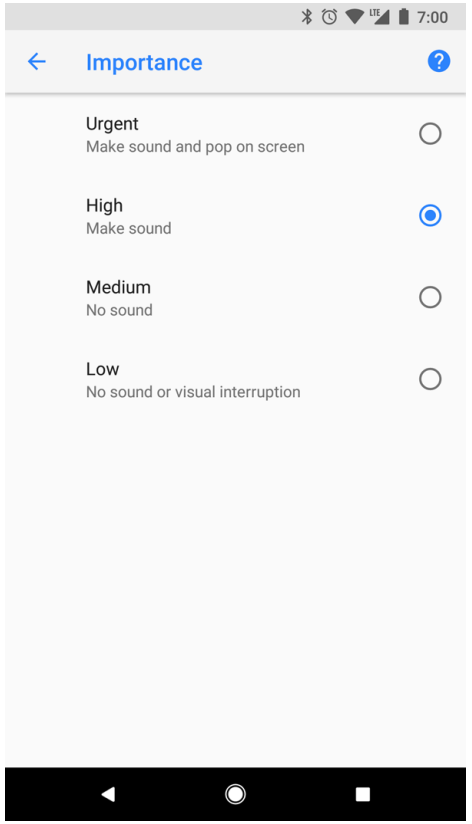
- Sous Android 8.0 (niveau d'API 26) et versions ultérieures, toutes les notifications doivent être attribuées à un canal, sans quoi elles ne s'affichent pas.
- En catégorisant les notifications en canaux, il serait possible d'agir sur les paramètres du système Android pour:
  - Désactiver certains canaux de notification de l'application (au lieu de *toutes* vos notifications).
  - Contrôler les options visuelles et sonores de chaque canal.
  - Appuyer de manière prolongée sur une notification pour modifier les comportements du canal associé.



Paramètres de notification pour l'application Clock (Horloge) et l'un de ses canaux

Importance d'une Notification

- C'est en fonction de l'*importance* d'une notification qu'Android détermine le degré d'interruption (visuelle et sonore) d'une notification.
  - Plus l'importance d'une notification est élevée, plus la notification peut être interruptive.
- Sous Android 8.0 (niveau d'API 26) et versions ultérieures, l'importance d'une notification dépend de l'importance du canal sur lequel la notification a été publiée.
  - Il est possible de modifier l'importance d'un canal de notification dans les paramètres système.
- Sous Android 7.1 (niveau d'API 25) et versions antérieures, l'importance de chaque notification est déterminée par la priorité (priority) de la notification.



Modification de l'importance de chaque canal sous Android 8.0 et versions ultérieures

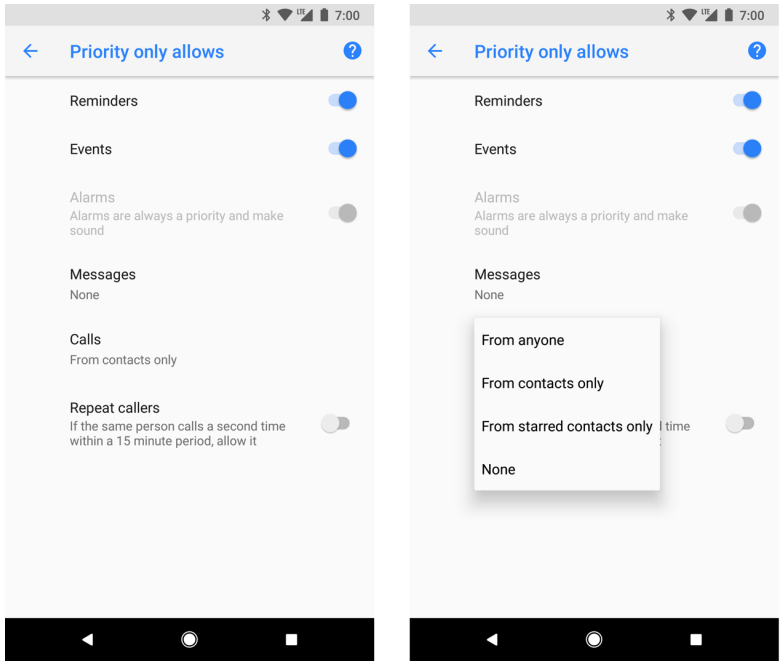
Importance d'une Notification

- Les niveaux d'importance possibles sont les suivants :
  - Urgence : émet un signal sonore et s'affiche sous la forme d'un avertissement.
  - Importance élevée : émet un signal sonore.
  - Importance moyenne : pas de signal sonore.
  - Importance faible : pas de signal sonore et ne s'affiche pas dans la barre d'état.

- Toutes les notifications, quelle que soit leur importance, s'affichent aux emplacements de l'UI du système qui n'impliquent pas d'interruption : panneau des notifications, badge sur l'icône.

Mode "Ne pas Déranger"

- Sous Android 5.0 (niveau d'API 21)
  - Le *mode "Ne pas déranger »* permet de désactiver les sons et les vibrations pour toutes les notifications.
  - Les notifications apparaissent toujours dans l'UI du système, sauf indication contraire de l'utilisateur.
- Le mode "Ne pas déranger" comporte trois niveaux différents :
  - *Silence total :* bloque tous les sons et les vibrations, y compris ceux des alarmes, de la musique, des vidéos et des jeux.
  - *Alarmes uniquement* : bloque tous les sons et vibrations, sauf les alarmes.
  - *Prioritaires uniquement* :
    - Les utilisateurs peuvent définir quelles catégories du système peuvent les interrompre (telles que les alarmes, les rappels, les événements, les appels ou les messages).
    - Pour les messages et les appels, il est possible de les choisir de les filtrer en fonction de l'expéditeur ou de l'appelant (figure 13).



Autoriser les notifications en fonction des catégories du système (à gauche) et de l'expéditeur ou de l'appelant (à droite).

Compatibilité des Notifications

- L'UI du système de notification et les API associées aux notifications ont évolué de façon constante avec les différentes versions d'Android.

- Pour utiliser les dernières fonctionnalités des API de notification tout en maintenant la compatibilité avec les anciens appareils, utilisez l'API de notification de la bibliothèque Support NotificationCompat et ses sous-classes, ainsi que NotificationManagerCompat.
  - Cela vous évitera d'écrire du code conditionnel pour vérifier les niveaux d'API, car ces API le font pour vous.
  - NotificationCompat est mis à jour à mesure de l'évolution de la plate-forme pour inclure les dernières méthodes.
  - La disponibilité d'une méthode dans NotificationCompat ne garantit pas que la fonctionnalité correspondante sera disponible sur les appareils anciens.
    - Dans certains cas, l'appel d'une nouvelle API interrompt toute opération sur les appareils anciens.
      - Par exemple, NotificationCompat.addAction() affiche uniquement le bouton d'action sur un appareil fonctionnant sous Android 4.1 (niveau d'API 16) et versions ultérieures.

Création d’une Notification

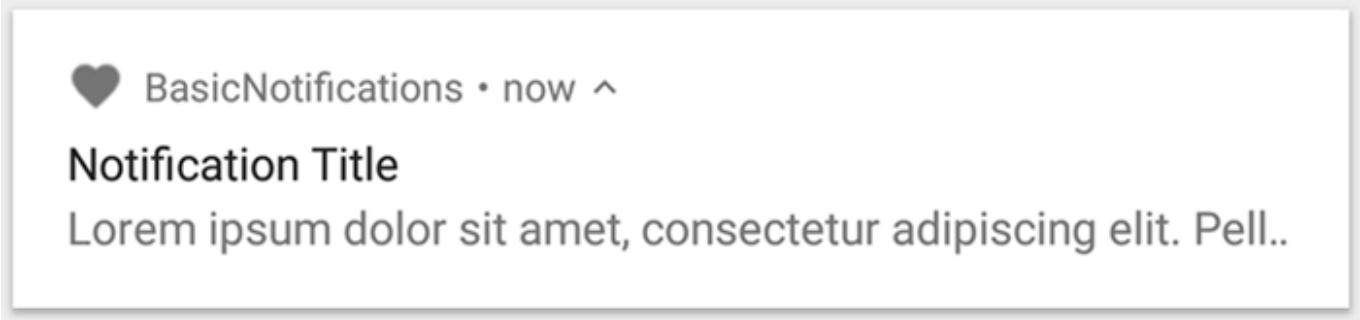
Ajout de la Bibliothèque Support

- Bien que la plupart des projets créés avec Android Studio incluent les dépendances nécessaires pour utiliser *NotificationCompat*, il est souhaitable de vérifier que le fichier *build.gradle* au niveau du module inclut la dépendance suivante:
  - *dépendances {implémentation "com.android.support:support-compat:28.0.0"}*

- D'autres bibliothèques du groupe *com.android.support* incluent également la compatibilité de support en tant que dépendance transitive.
  - Donc, d'autres API de bibliothèque de support, sont utilisées, il serait possible d'accéder à *NotificationCompat* sans nécessiter la dépendance exacte.

Créer une Notification Simple

- Une notification dans sa forme la plus simple et la plus compacte également connue sous le nom de formulaire réduit affiche :
  - Une icône.
  - Un titre.
  - Une petite quantité de texte de contenu.



Une notification avec un titre et un texte

Définir le Contenu de la Notification

- Définir le contenu et le canal de la notification à l'aide d'un objet *NotificationCompat.Builder*.
- Exemple : Créer une notification avec les éléments suivants:
  - Une petite icône, définie par *setSmallIcon ()*.
    - Il s'agit du seul contenu visible requis.
  - Un titre, défini par *setContentTitle ().*
  - Le corps du texte, défini par *setContentText ().*
  - La priorité de notification, définie par *setPriority ().*
    - La priorité détermine le degré d'intrusion de la notification sur Android 7.1 et inférieur.
    - Pour Android 8.0 et supérieur, définir l'importance du canal.

```
NotificationCompat.Builder builder = new NotificationCompat.Builder( this, CHANNEL_ID )
        .setSmallIcon( R.drawable.notification_icon )
        .setContentTitle( textTitle )
        .setContentText( textContent )
        .setPriority( NotificationCompat.PRIORITY_DEFAULT );
```

Définir le Contenu de la Notification

- Le constructeur *NotificationCompat.Builder* requiert un ID de canal.
  - Ceci est requis pour la compatibilité avec Android 8.0 (API niveau 26) et supérieur, mais est ignoré par les anciennes versions.
  - Par défaut, le contenu textuel de la notification est tronqué pour tenir sur une seule ligne.
    - Pour que la notification soit plus longue, il faut activer une notification extensible en ajoutant un modèle de style avec *setStyle ().*
    - Exemple :

```
NotificationCompat.Builder builder = new NotificationCompat.Builder( this, CHANNEL_ID )
        .setSmallIcon( R.drawable.notification_icon )
        .setContentTitle( "My notification" )
        .setContentText( "Much longer text that cannot fit one line..." )
        .setStyle( new NotificationCompat.BigTextStyle( )
            .bigText( "Much longer text that cannot fit one line..." ) )
        .setPriority( NotificationCompat.PRIORITY_DEFAULT );
```

Créez un Canal et définissez l'importance

- Avant de pouvoir envoyer la notification sur Android 8.0 et supérieur, il est nécessaire d'enregistrer le canal de notification de l'application auprès du système en transmettant une instance de *NotificationChannel* à *createNotificationChannel ()*.
  - Le code suivant est bloqué par une condition sur la version SDK_INT:

```
private void createNotificationChannel( ) {
    // Create the NotificationChannel, but only on API 26+ because
    // the NotificationChannel class is new and not in the support library
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O ) {
        CharSequence name = getString(R.string.channel_name );
        String description = getString(R.string.channel_description );
        int importance = NotificationManager.IMPORTANCE_DEFAULT;
        NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name, importance );
        channel.setDescription(description );
        // Register the channel with the system; you cant change the importance
        // or other notification behaviors after this
        NotificationManager notificationManager = getSystemService(NotificationManager.class );
        notificationManager.createNotificationChannel(channel );
    }
}
```

Créez un canal et définissez l'importance

- Comme il est nécessaire de créer le canal de notification avant de publier des notifications sur Android 8.0 et supérieur, il faut exécuter le code précédent dès le démarrage de l'application.
  - Il est prudent d'appeler ce code à plusieurs reprises, car la création d'un canal de notification existant n'effectue aucune opération.
  - Notez que le constructeur *NotificationChannel* requiert une importance, en utilisant l'une des constantes de la classe *NotificationManager.*
    - Ce paramètre détermine comment interrompre l'utilisateur pour toute notification appartenant à ce canal.
  - Il est également nécessaire de définir la priorité avec *setPriority ()* pour prendre en charge Android 7.1 et inférieur (comme illustré ci-dessus).

Définir l'Action de Toucher de la Notification

- Chaque notification doit répondre à un tapotement, généralement pour ouvrir une activité dans l'application qui correspond à la notification.
- Pour ce faire, il est nécessaire de spécifier une intention de contenu définie avec un objet *PendingIntent* et la transmettre à *setContentIntent ().*
- Exemple : L'extrait de code suivant montre comment créer une intention de base pour ouvrir une activité lorsque l'utilisateur appuie sur la notification:

```
// Create an explicit intent for an Activity in your app
Intent intent = new Intent(this, AlertDetails.class);
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, intent, 0);

NotificationCompat.Builder builder = new NotificationCompat.Builder(this, CHANNEL_ID)
        .setSmallIcon(R.drawable.notification_icon)
        .setContentTitle("My notification")
        .setContentText("Hello World!")
        .setPriority(NotificationCompat.PRIORITY_DEFAULT)
        // Set the intent that will fire when the user taps the notification
        .setContentIntent(pendingIntent)
        .setAutoCancel(true);
```

Définir l’Action de Toucher de la Notification

- Ce code appelle *setAutoCancel ()*, qui supprime automatiquement la notification lorsque l'utilisateur la touche.

- La méthode *setFlags ()* permet de préserver l'expérience de navigation attendue de l'utilisateur après l'ouverture d’une application via la notification. Cela dépend du type d'activité démarrée, qui peut être l'un des suivants:
  - Une activité qui existe exclusivement pour les réponses à la notification.
    - Il n'y a aucune raison pour que l'utilisateur accède à cette activité lors de l'utilisation normale de l'application, donc l'activité démarre une nouvelle tâche au lieu d'être ajoutée à la tâche existante et à la pile arrière de votre application. Il s'agit du type d'intention créé dans l'exemple.

  - Une activité qui existe dans le flux d'applications régulier de l’application.
    - Dans ce cas, le démarrage de l'activité doit créer une pile arrière afin que les attentes de l'utilisateur pour les boutons *Précédent* soient préservées.

Afficher la Notification

- Pour faire apparaître la notification, appel de *NotificationManagerCompat.notify ()*, en passant comme paramètre un ID unique pour la notification et le résultat de *NotificationCompat.Builder.build ()*.

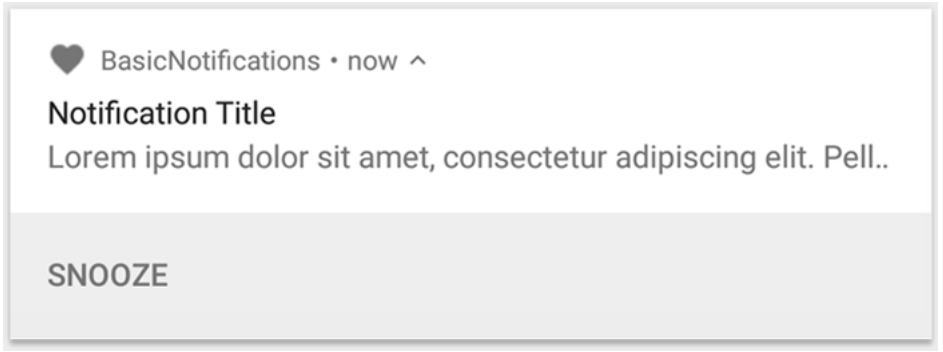
```
NotificationManagerCompat notificationManager = NotificationManagerCompat.from( this );

// notificationId is a unique int for each notification that you must define
notificationManager.notify( notificationId, builder.build( ) );
```

- Il faut enregistrer l'ID de notification transmis à NotificationManagerCompat.notify ()
  - Nécessaire plus tard pour mettre à jour ou supprimer la notification.
- A partir d'Android 8.1 (niveau 27 de l'API), les applications ne peuvent pas émettre une notification plus d'une fois par seconde.
  - Si l'application publie plusieurs notifications en une seconde, elles apparaissent toutes comme prévu, mais seule la première notification par seconde émet un son.

Ajouter des Boutons d’Action

- Une notification peut proposer jusqu'à trois boutons d'action qui permettent à l'utilisateur de répondre rapidement, comme répondre à un SMS.
-
- Ces boutons d'action ne doivent pas dupliquer l'action effectuée lorsque l'utilisateur appuie sur la notification.



Une notification avec un bouton d'action

Ajouter des Boutons d'Action

- Pour ajouter un bouton d'action, passer un *PendingIntent* à la méthode *addAction ()*.
  - Cela revient à configurer l'action « toucher par défaut » de la notification, sauf qu'au lieu de lancer une activité, faire une variété d'autres choses, comme démarrer un *BroadcastReceiver* qui effectue un travail en arrière-plan afin que l'action n'interrompe pas l'application qui est déjà ouverte.
- Exemple, le code suivant montre comment envoyer une diffusion à un récepteur spécifique:

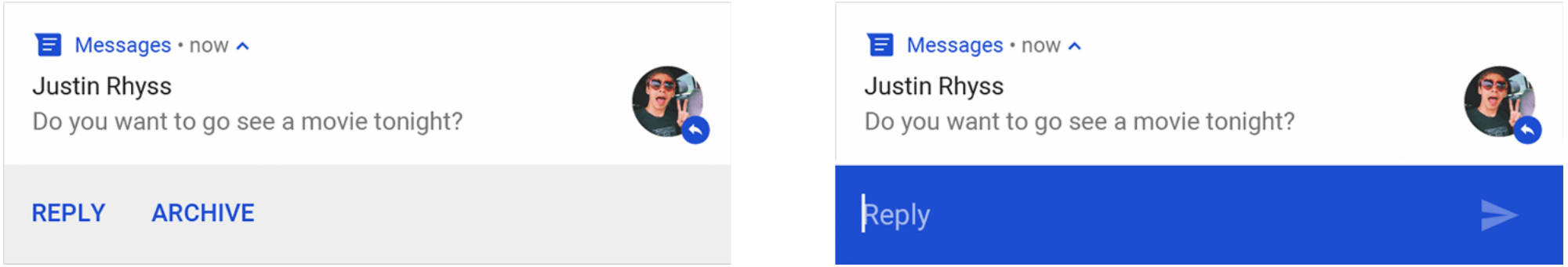
```
Intent snoozeIntent = new Intent(this, MyBroadcastReceiver.class);
snoozeIntent.setAction(ACTION_SNOOZE);
snoozeIntent.putExtra(EXTRA_NOTIFICATION_ID, 0);
PendingIntent snoozePendingIntent =
    PendingIntent.getBroadcast(this, 0, snoozeIntent, 0);

NotificationCompat.Builder builder = new NotificationCompat.Builder(this, CHANNEL_ID)
    .setSmallIcon(R.drawable.notification_icon)
    .setContentTitle("My notification")
    .setContentText("Hello World!")
    .setPriority(NotificationCompat.PRIORITY_DEFAULT)
    .setContentIntent(pendingIntent)
    .addAction(R.drawable.ic_snooze, getString(R.string.snooze),
        snoozePendingIntent);
```

Ajouter une Action de Réponse Directe

- L'action de réponse directe, introduite dans Android 7.0 (API niveau 24), permet aux utilisateurs de saisir du texte directement dans la notification, qui est envoyée à l'application sans ouvrir d'activité.
- L'action de réponse directe apparaît comme un bouton supplémentaire dans la notification qui ouvre une entrée de texte.
  - Lorsque l'utilisateur a fini de taper, le système joint la réponse textuelle à l'intention spécifiée pour l'action de notification et envoie l'intention à l'application.

- Exemple : Utiliser une action de réponse directe pour permettre aux utilisateurs de répondre aux messages texte ou de mettre à jour les listes de tâches à partir de la notification.



Appuyez sur le bouton "Répondre" pour ouvrir l'entrée de texte

Ajouter une Action de Réponse Directe

- Ajouter un bouton de réponse
  - Pour créer une action de notification prenant en charge la réponse directe:
    1. Créez une instance de *RemoteInput.Builder* à ajouter à l'action de notification.
       - Le constructeur de cette classe accepte une chaîne de caractères que le système utilise comme clé pour la saisie de texte.
       - L'application utilise cette clé pour récupérer le texte de l'entrée.

```
// Key for the string thats delivered in the actions intent.
private static final String KEY_TEXT_REPLY = "key_text_reply";

String replyLabel = getResources().getString(R.string.reply_label);
RemoteInput remoteInput = new RemoteInput.Builder(KEY_TEXT_REPLY)
        .setLabel(replyLabel)
        .build();
```

Ajouter une Action de Réponse Directe

- Ajouter un bouton de réponse
  - Pour créer une action de notification prenant en charge la réponse directe:
    1. Créez une instance de *RemoteInput.Builder* à ajouter à l'action de notification.
    2. Créez un *PendingIntent* pour l'action de réponse.

```
// Build a PendingIntent for the reply action to trigger.
PendingIntent replyPendingIntent =
        PendingIntent.getBroadcast(getApplicationContext(),
            conversation.getConversationId(),
            getMessageReplyIntent(conversation.getConversationId()),
            PendingIntent.FLAG_UPDATE_CURRENT);
```

Ajouter une Action de Réponse Directe

- Ajouter un bouton de réponse
  - Pour créer une action de notification prenant en charge la réponse directe:
    1. Créez une instance de *RemoteInput.Builder* à ajouter à l'action de notification.
    2. Créez un *PendingIntent* pour l'action de réponse.
    3. Attachez l'objet *RemoteInput* à une action à l'aide de *addRemoteInput ().*

```
// Create the reply action and add the remote input.
NotificationCompat.Action action =
        new NotificationCompat.Action.Builder(R.drawable.ic_reply_icon,
                getString(R.string.label), replyPendingIntent)
                .addRemoteInput(remoteInput)
                .build();
```

Ajouter une action de réponse directe

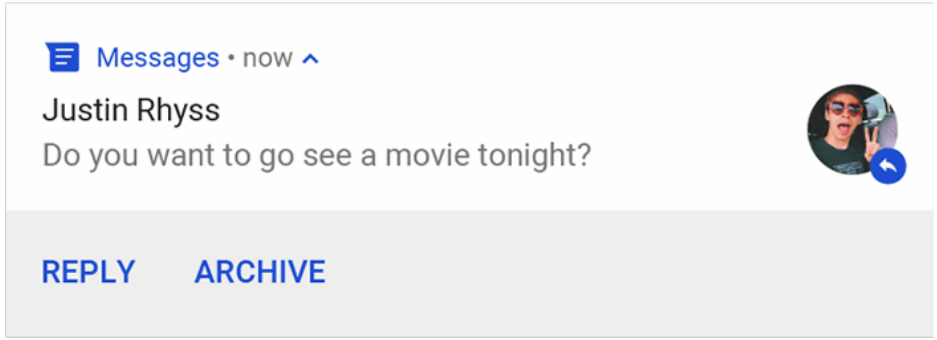
- Ajouter un bouton de réponse
  - Pour créer une action de notification prenant en charge la réponse directe:
    1. Créez une instance de *RemoteInput.Builder* à ajouter à l'action de notification.
    2. Créez un *PendingIntent* pour l'action de réponse.
    3. Attachez l'objet *RemoteInput* à une action à l'aide de *addRemoteInput ().*
    4. Appliquez l'action à une notification et émettez la notification.

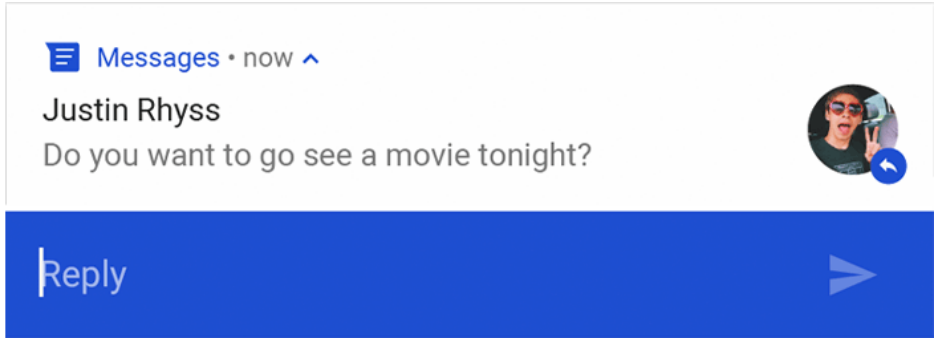
```
// Build the notification and add the action.
Notification newMessageNotification = new Notification.Builder( context, CHANNEL_ID )
        .setSmallIcon( R.drawable.ic_message )
        .setContentTitle( getString( R.string.title ) )
        .setContentText( getString( R.string.content ) )
        .addAction( action )
        .build( );

// Issue the notification.
NotificationManagerCompat notificationManager = NotificationManagerCompat.from( this );
notificationManager.notify( notificationId, newMessageNotification );
```

Ajouter une action de réponse directe

- Ajouter un bouton de réponse
  - Pour créer une action de notification prenant en charge la réponse directe:
    1. Créez une instance de *RemoteInput.Builder* à ajouter à l'action de notification.
    2. Créez un *PendingIntent* pour l'action de réponse.
    3. Attachez l'objet *RemoteInput* à une action à l'aide de *addRemoteInput ().*
    4. Appliquez l'action à une notification et émettez la notification.
    5. Le système invite l'utilisateur à saisir une réponse lorsqu'il déclenche l'action de notification





Ajouter une Action de Réponse Directe

- Récupérer l'entrée utilisateur à partir de la réponse
  - Pour recevoir l'entrée utilisateur à partir de l'interface de réponse de la notification, appeler *RemoteInput.getResultsFromIntent ()*, en lui transmettant l'intention reçue par le *BroadcastReceiver*:

```
private CharSequence getMessageText(Intent intent) {
   Bundle remoteInput = RemoteInput.getResultsFromIntent(intent);
   if (remoteInput != null) {
      return remoteInput.getCharSequence(KEY_TEXT_REPLY);
   }
   return null;
}
```

Ajouter une Action de Réponse Directe

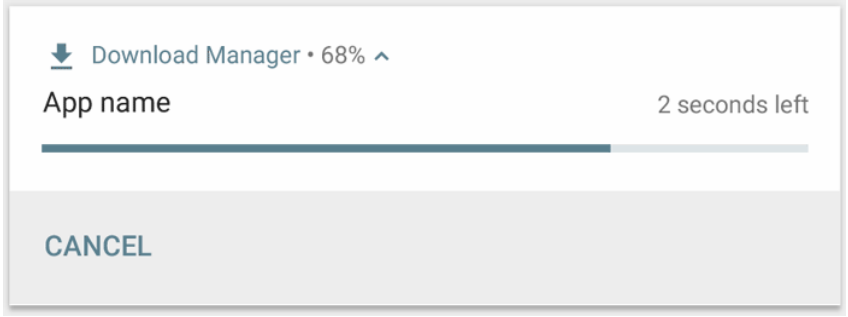
- Récupérer l'entrée utilisateur à partir de la réponse:
  - Après avoir traité le texte, mettre à jour la notification en appelant *NotificationManagerCompat.notify ()* avec le même ID et la même balise/tag (le cas échéant).
    - Cela est nécessaire pour masquer l'interface utilisateur de réponse directe et confirmer à l'utilisateur que sa réponse a été reçue et traitée correctement.

```
// Build a new notification, which informs the user that the system
// handled their interaction with the previous notification.
Notification repliedNotification = new Notification.Builder(context, CHANNEL_ID)
        .setSmallIcon(R.drawable.ic_message)
        .setContentText(getString(R.string.replied))
        .build();

// Issue the new notification.
NotificationManagerCompat notificationManager = NotificationManagerCompat.from(this);
notificationManager.notify(notificationId, repliedNotification);
```

Ajouter une Barre de Progression

- Les notifications peuvent inclure un indicateur de progression animé qui montre aux utilisateurs l'état d'une opération en cours.



La barre de progression pendant et après l'opération

- S'il est possible d'estimer à quel moment l'opération sera terminée, utiliser la forme "déterminée" de l'indicateur en appelant *setProgress (max, progress, false).*
  - Le premier paramètre est la valeur "complète" (par exemple 100).
  - Le deuxième paramètre est la quantité actuellement terminée.
  - Le troisième paramètre indique qu'il s'agit d'une barre de progression déterminée.
- Au fur et à mesure de l'opération, appelez en continu *setProgress (max, progress, false)* avec une valeur mise à jour pour la progression et relancez la notification.

Ajouter une Barre de Progression

```
...
NotificationManagerCompat notificationManager = NotificationManagerCompat.from( this );
NotificationCompat.Builder builder = new NotificationCompat.Builder( this, CHANNEL_ID );
builder.setContentTitle( "Picture Download" )
    .setContentText( "Download in progress" )
    .setSmallIcon( R.drawable.ic_notification )
    .setPriority( NotificationCompat.PRIORITY_LOW );

// Issue the initial notification with zero progress
int PROGRESS_MAX = 100;
int PROGRESS_CURRENT = 0;
builder.setProgress( PROGRESS_MAX, PROGRESS_CURRENT, false );
notificationManager.notify( notificationId, builder.build( ) );

// Do the job here that tracks the progress.
// Usually, this should be in a
// worker thread
// To show progress, update PROGRESS_CURRENT and update the notification with:
// builder.setProgress( PROGRESS_MAX, PROGRESS_CURRENT, false );
// notificationManager.notify( notificationId, builder.build( ) );

// When done, update the notification one more time to remove the progress bar
builder.setContentText( "Download complete" )
    .setProgress( 0,0,false );
notificationManager.notify( notificationId, builder.build( ) );
```

Définir une Catégorie à l'Echelle du Système

- Android utilise certaines catégories prédéfinies à l'échelle du système pour déterminer s'il faut déranger l'utilisateur avec une notification donnée lorsque l'utilisateur a activé le mode « *Ne pas déranger*»
- Si une notification appartient à l'une des catégories de notification prédéfinies définies dans *NotificationCompat*, il est nécessaire de la déclarer comme telle en passant la catégorie appropriée à *setCategory ().*
  - Exemples de catégories prédéfinies dénies dans *NotificationCompat* : *CATEGORY_ALARM*, C*ATEGORY_REMINDER*, *CATEGORY_EVENT* ou *CATEGORY_CALL*.
- Ces informations sur la catégorie de notification sont utilisées par le système pour prendre des décisions concernant l'affichage de la notification lorsque l'appareil est en mode « *Ne pas déranger* ».
  - A faire uniquement si les notifications correspondent à l'une des catégories définies dans *NotificationCompat*.

Définir une Catégorie à l'Echelle du Système

```
NotificationCompat.Builder builder = new NotificationCompat.Builder( this, CHANNEL_ID )
        .setSmallIcon( R.drawable.notification_icon )
        .setContentTitle( "My notification" )
        .setContentText( "Hello World!" )
        .setPriority( NotificationCompat.PRIORITY_DEFAULT )
        .setCategory( NotificationCompat.CATEGORY_MESSAGE );
```

Afficher un Message Urgent

- Une application peut avoir besoin d'afficher un message urgent et sensible au temps, tel qu'un appel téléphonique entrant ou une alarme qui sonne.
- Dans ces situations, il faut associer une intention « *plein écran/ full-screen* » à la notification.
- Lorsque la notification est invoquée, les utilisateurs voient l'un des éléments suivants, selon l'état de verrouillage de l'appareil:
  - Si l'appareil de l'utilisateur est verrouillé, une activité plein écran apparaît, couvrant l'écran de verrouillage.
  - Si l'appareil de l'utilisateur est déverrouillé, la notification apparaît sous une forme développée qui inclut des options pour gérer ou ignorer la notification.
- Les notifications contenant des intentions plein écran sont essentiellement intrusives, il est donc important d'utiliser ce type de notification uniquement pour les messages les plus urgents et sensibles au facteur temps
- Si l'application cible Android 10 (niveau API 29) ou supérieur, il est nécessaire de demander l'autorisation *USE_FULL_SCREEN_INTENT* dans le fichier manifeste de l'application afin que le système lance l'activité en plein écran associée à la notification urgente.

Afficher un Message Urgent

- Un extrait de code qui montre comment associer une notification à une intention plein écran:

```
Intent fullScreenIntent = new Intent( this, ImportantActivity.class );
PendingIntent fullScreenPendingIntent = PendingIntent.getActivity( this, 0,
      fullScreenIntent, PendingIntent.FLAG_UPDATE_CURRENT );

NotificationCompat.Builder builder = new NotificationCompat.Builder( this, CHANNEL_ID )
      .setSmallIcon( R.drawable.notification_icon )
      .setContentTitle( "My notification" )
      .setContentText( "Hello World!" )
      .setPriority( NotificationCompat.PRIORITY_DEFAULT )
      .setFullScreenIntent( fullScreenPendingIntent, true );
```

Définir la Visibilité de l'Ecran de Verrouillage

- Pour contrôler le niveau de détail visible dans la notification à partir de l'écran de verrouillage:
  - Appelez *setVisibility ()*
  - Spécifiez l'une des valeurs suivantes:
    - *VISIBILITY_PUBLIC* : affiche le contenu complet de la notification.
    - *VISIBILITY_SECRET* : n'affiche aucune partie de cette notification sur l'écran de verrouillage.
    - *VISIBILITY_PRIVATE* : affiche des informations de base, telles que l'icône de la notification et le titre du contenu, mais masque le contenu complet de la notification.
      - Lorsque *VISIBILITY_PRIVATE* est défini, vous pouvez également fournir une autre version du contenu de la notification qui masque certains détails.
      - Exemple
        - Une application SMS peut afficher une notification indiquant que vous avez 3 nouveaux messages texte, mais masque le contenu du message et les expéditeurs.
        - Pour fournir cette notification alternative
          - Créez d'abord la notification alternative avec *NotificationCompat.Builder*.
          - Attachez la notification alternative à la notification normale avec *setPublicVersion ().*

Mettre à Jour une Notification

- Pour mettre à jour une notification après l'avoir émise, appeler à nouveau *NotificationManagerCompat.notify (),* en lui passant une notification avec le même ID utilisé précédemment.
- Si la notification précédente a été rejetée, une nouvelle notification est créée à la place.
- Il est éventuellement possible d'appeler *setOnlyAlertOnce ()* afin que la notification n'interrompe l'utilisateur (avec des indices sonores, vibratoires ou visuels) que la première fois que la notification apparaît et pas pour des mises à jour ultérieures.
- Attention: Android applique une limite de taux lors de la mise à jour d'une notification. Si vous publiez des mises à jour d'une notification trop fréquemment (beaucoup en moins d'une seconde), le système peut supprimer certaines mises à jour.

Supprimer une Notification

- Les notifications restent visibles jusqu'à ce que l'un des événements suivants se produise:
  - L'utilisateur rejette la notification.
  - L'utilisateur clique sur la notification.
  - Vous avez appelé *setAutoCancel ()* lorsque vous avez créé la notification.
  - Vous appelez *cancel ()* pour un ID de notification spécifique. Cette méthode supprime également les notifications en cours.
  - Vous appelez *cancelAll (),* qui supprime toutes les notifications que vous avez précédemment émises.
  - Si vous définissez un délai d'expiration lors de la création d'une notification à l'aide *de setTimeoutAfter (),* le système annule la notification une fois la durée spécifiée écoulée. Si nécessaire, vous pouvez annuler une notification avant l'expiration du délai spécifié.