Les notifications

LAFRANCE Antoine (FI)
23/10/2018

Résumé

Les notifications sont une des fonctionnalités fondamentales des applications. La plupart des applications existantes les utilisent car elles sont très utiles et permettent de mettre en avant une information à l'utilisateur à un moment précis, sans qu'il ait à lancer l'application.

Dans ce TP nous verrons qu'il est très simple de mettre en place une notification, mais aussi qu'il est possible de lui donner plusieurs aspects en fonction de ce que l'on souhaite mettre en avant.

Code source

Code source **initial** : https://github.com/ig1na/TP-Notifications-Kotlin/tree/master/init

Code source **final** : https://github.com/ig1na/TP-Notifications-Kotlin/tree/master/final

Ressources : https://github.com/ig1na/TP-Notifications-Kotlin/tree/master/resources

Pré-requis

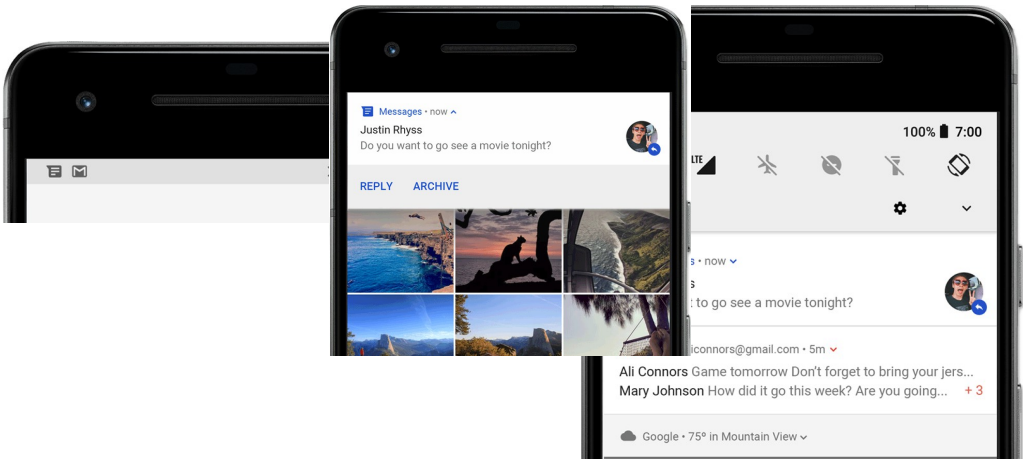
- Savoir créer un projet sous Android Studio
- Savoir créer un bouton et lui assigner une action
- Savoir créer une activité

Explications du TP

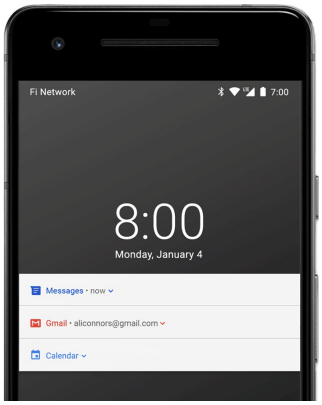
Les notifications Android peuvent apparaître sous plusieurs formes, de la plus basique où l'on obtient une information textuelle, à la plus sophistiquée où il est possible d'interagir directement avec l'application. Les notifications peuvent apparaître à différents endroits du système comme par exemple sur l'écran de verrouillage, dans la barre d'état, ou directement sur le haut de l'écran.

Les notifications les plus simples affichent dans un premier temps une icône dans la barre d'état, puis si l'utilisateur étend la barre d'état, ces dernières afficheront une vue plus détaillées avec éventuellement du texte et d'autres informations comme l'heure où la notification a été émise par exemple.

Si une notification est très importante, il est possible depuis Android 5.0 de l'afficher très brièvement en avant-plan, au dessus de l'application que l'utilisateur est en train d'utiliser.



L'arrivée d'Android 5.0 a aussi permis d'afficher des notifications sur l'écran de verrouillage, et l'utilisateur peut même décider de ce qu'il souhaite que l'application affiche comme détails, par exemple il peut décider de n'afficher que le nom de l'application et pas le contenu de la notification.



Les notifications peuvent également déclencher des actions, telles que lancer une activité si on appuie dessus, ou encore contrôler certaines fonctionnalités de l'application comme mettre en pause une musique.

Depuis Android 8.0, les notifications doivent être assignées à une catégorie, ce qui permet à l'utilisateur de désactiver certaines catégories de notifications, pour ne plus recevoir certaines d'entre elles mais toujours recevoir les autres. La notion d'importance est également présente et permet d'avoir différents niveau ce qui va déterminer de quelle façon l'utilisateur est interrompu par une notification.

Nous ne le verrons pas dans le cadre de ce TP, mais il est aussi possible d'afficher une petite pastille de couleur au dessus de l'icône de l'application ayant des notifications tant que l'utilisateur n'a pas ouvert celle-ci. Il est aussi possible de gérer les notifications sur les objets connectés comme les montres.

Pour plus d'informations, vous pouvez consulter le guide officiel :
https://developer.android.com/guide/topics/ui/notifiers/notifications

Etape 1 : créer un bouton pour lancer une notification

Pour déclencher et afficher une notification sur notre activité principale, nous allons utiliser un simple bouton. Bien-sûr dans un contexte réel, lancer une notification à partir d'un bouton n'aurait aucun intérêt.

Dans un premier temps, vous pouvez importer le projet qui se trouve dans le dossier « **init** » du git de ce TP.

Vous pouvez ensuite créer une **nouvelle activité** que vous appellerez « **NotificationDetails** », que nous lancerons lors d'un appui sur notre notification, et un bouton que vous appellerez « **notification_1** », pour lancer et afficher la notification.

Une fois ceci fait, nous allons créer notre notification à l'aide du **NotificationCompat.Builder.** Dans un premier temps, créez une fonction privée **createNotification()** puis ajoutez-y ce code :

```
var mBuilder: NotificationCompat.Builder = NotificationCompat.Builder(this, CHANNEL_ID)
        .setSmallIcon(R.drawable.heart)
        .setContentTitle(textTitle)
        .setContentText(textContent)
        .setPriority(NotificationCompat.PRIORITY_DEFAULT)
```

En observant ce code, nous pouvons constater plusieurs fonctions qui permettent de construire notre notification. Premièrement nous définissons une icône avec la fonction **setSmallIcon**. Cependant cette icône n'existe pas, il va falloir l'ajouter comme ressource à notre projet. Pour ce faire, récupérez l'image « **heart.png** » qui se trouve dans le dossier « **resources** » du **git**. Sur **android studio**, dans le **panneau projet à gauche**, allez dans le dossier « **res** » puis faites **clic droit** sur le dossier « **drawable** » et cliquez sur « **Show in explorer** ». Le dossier « **drawable** » de votre projet s'ouvre alors dans l'explorateur de fichiers. **Copiez-collez le fichier** « **heart.png** » dans ce dossier.

Vous pouvez aussi remarquer la présence de deux fonctions, **setContentTitle** qui va définir le titre de la notification, et **setContentText** qui définit le texte. Ces variables sont déjà présentes dans le code initial mais vous pouvez les modifier si vous le souhaitez.

Enfin, nous pouvons constater la présence de la fonction **setPriority** qui elle détermine à quel point la notification va interrompre l'utilisateur. Pour celle-ci, nous lassons la valeur par défaut.

Comme indiqué dans l'introduction, depuis Android 8.0 il est nécessaire d'enregistrer chaque notification dans une catégorie ou **channel**. Pour ce faire, vous pouvez copier cette fonction, puis **appelez là dans votre fonction onCreate**. N'oubliez pas de corriger les erreurs en ajoutant les imports.

```
private fun createNotificationChannel() {
       if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
              val name = channel_name
              val descriptionText = channel_description
              val importance = NotificationManager.IMPORTANCE_DEFAULT
              val channel = NotificationChannel(CHANNEL_ID, name, importance).apply {
                     description = descriptionText
                     }
              val notificationManager: NotificationManager =
                     getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager
              notificationManager.createNotificationChannel(channel)
       }
}
```

Afin de répondre à un appui sur la notification, nous allons lui assigner une activité. Pour ce faire, recopiez ce code dans la fonction **createNotification,** avant la création du **mBuilder.**

```
val intent = Intent(this, NotificationDetails::class.java).apply {
        flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TASK
}
val pendingIntent: PendingIntent = PendingIntent.getActivity(this, 0, intent, 0)
```

Ce code a pour but de créer une **Intent** de notre activité **NotificationDetails** en attente, qui sera appelée par un appui sur la notification. Il ne faut pas oublier cependant, d'ajouter ces ligne à la suite de notre **mBuilder**.

```
.setContentIntent(pendingIntent)
.setAutoCancel(true)
```

La première ligne permet d'assigner notre **Intent** à notre notification, et la deuxième de supprimer la notification une fois que l'on a appuyé dessus.

Il faut maintenant pouvoir afficher la notification, pour ce faire, recopier ce code à la suite de notre **mBuilder** dans votre fonction **createNotification** :

```
with(NotificationManagerCompat.from(this)) {
        notify(notif_1_ID, mBuilder.build())
}
```

Nous avons tout ce qu'il faut pour créer et afficher notre notification, cependant, il faut qu'un élément déclencheur puisse appeler notre fonction **createNotification.** Nous allons donc ajouter ce code à la fin de notre fonction **onCreate** ce qui aura pour but d'ajouter un **listener** à notre bouton et ainsi d'appeler notre fonction lors d'un appui **:**

```
val button1 = findViewById<Button>(R.id.notification_1)
button1.setOnClickListener(object : View.OnClickListener {
        override fun onClick(v: View) {
                createNotification()
        }
})
```

Ceci va assigner notre bouton à une variable, et lui donner la fonction createNotification comme listener.

Vous pouvez compiler et tester sur votre smartphone. Une notification devrait apparaître dans la barre d'état lorsque vous appuierez sur le bouton. Si vous déroulez la barre d'état et appuyez sur la notification, la seconde activité devrait se lancer.

Etape 2 : ajouter des fonctionnalités à notre notification

Il est possible de rendre votre notification étendable, c'est à dire que si elle contient une image ou du texte trop grand pour tenir sur l'affichage normal dans la barre d'état, il est possible de l'étendre pour tout afficher. Nous allons nous intéresser ici au texte.

Pour ce faire, c'est très simple, il suffit d'ajouter une fonction à notre builder :

```
.setStyle(NotificationCompat.BigTextStyle().bigText("Un très long text qu'il est impossible
d'afficher sur une notification normale, nous devons donc l'étendre pour l'afficher. Bla bla
bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla"))
```

Maintenant, si vous appuyez sur le haut de votre notification, elle devrait s'étendre pour afficher plus de texte.

Il est possible d'ajouter un bouton directement sur la notification pour pouvoir effectuer une action sans forcément se rendre sur l'application, car, en général, un clic sur la notification nous fait nous rendre sur une activité de notre application.

Pour ce faire il suffit d'ajouter la fonction **addAction** à notre **mBuilder :**

```
        .addAction(R.drawable.heart, "ACTION", pendingIntent)
```

Ici nous réutilisons les mêmes composants que tout à l'heure, mais il est possible évidemment d'indiquer une autre activité, une autre icône et un autre titre. Compilez et vérifiez que votre notification possède bien un bouton lorsque vous l'étendez.

Beaucoup d'autres possibilités s'offrent à vous dans la création de notification, et nous n'avons vu ici qu'une infime partie de ce qu'il est possible de faire. N'hésitez pas à consulter le guide officiel pour en savoir plus.

Informations complémentaires

- *https://developer.android.com/training/notify-user/build-notification#kotlin Le guide pour créer des notifications*