HOCHSCHULE
RAVENSBURG-WEINGARTEN
UNIVERSITY
OF APPLIED SCIENCES

BEST PRACTISES FÜR PUSH-NOTIFICATIONS FÜR ANDROID APPS

Simon Fentzl 30663
Stand: 23.02.2022
Abgabedatum: 28.02.2022

Inhaltsverzeichnis

Abbildungsverzeichnis

Einleitung

Als Neuling im Thema Android-Apps war es am Anfang schwierig zu finden, wie man einen Benutzer über Vorgänge in seiner App informiert. Eine Google Suche spuckte da schon des Öfteren aus, wie der Benutzer mit seinen Nachrichten umgehen sollte aus, aber nicht wie ein Programmierer diese umsetzen kann. Nach einiger Zeit stieß ich dann auf den Namen Push-Notification. Ab dann, war es nicht mehr schwer schnell Tutorials und andere Anleitungen zu finden. Doch wie sie nun Einsetzen? Man kennt nur allzu gut die Nervigen Pieptöne, wenn mal wieder eine App was Neues hat, der Freund schreibt oder das Spiel gerade dir sagen will, dass deine Basis attackiert wird. Somit war das nächste was ich suchen musste, WIE setze ich diese Nachrichten gezielt ein.

Um anderen diesen langen Weg zu erleichtern, wird innerhalb dieser Arbeit beleuchtet,

- Was Notifications sind
- Wo und wie sie einsetzbar sind
- Welche Richtlinien man folgen sollte
- Welche Zeilen Code benötigt werden

Im Zuge dieser Arbeit ist auch eine Komplette Demo-App entstanden. Diese ist auf GitHub frei verfügbar.

- https://github.com/Cayleb-Ordo/Notification-Demo

In dieser Dokument sind nicht alle Notification Arten als Code ausführlich aufgeführt. Daher verweise ich an dieser Stelle an dieses Repository und die dort enthaltene README.

Was sind Notifications?

Eine Notification ist ein Weg, dass das System/die App dem Benutzer wichtige Ereignisse mitteilen kann. Der Benutzer hat die Option darauf zu reagieren.

Android hat verschiedene Wege diese Anzuzeigen.

1. Als kleines Icon auf der linken Seiter der Status Bar
2. Aufgeklappt in der Notification-Schublade
3. Bei wichtigen Ereignissen als Heads-Up Notification
4. Bei gesperrtem Bildschirm

Aufbau

Das Grunddesign einer Notification ist immer an das des Systems gekoppelt. Eine App kann lediglich Bereiche der Notification verändern. Sollte dies absolut nicht mit den gewollten Effekten vereinbar sein kann eine komplett Eigenerstellte Notification gebaut werden.

Wichtig ist, mache Details erst in der Aufgeklappten Ansicht einer Notification angezeigt werden. In Abbildung 1 wären dies die Punkte 4, 5 und 6.

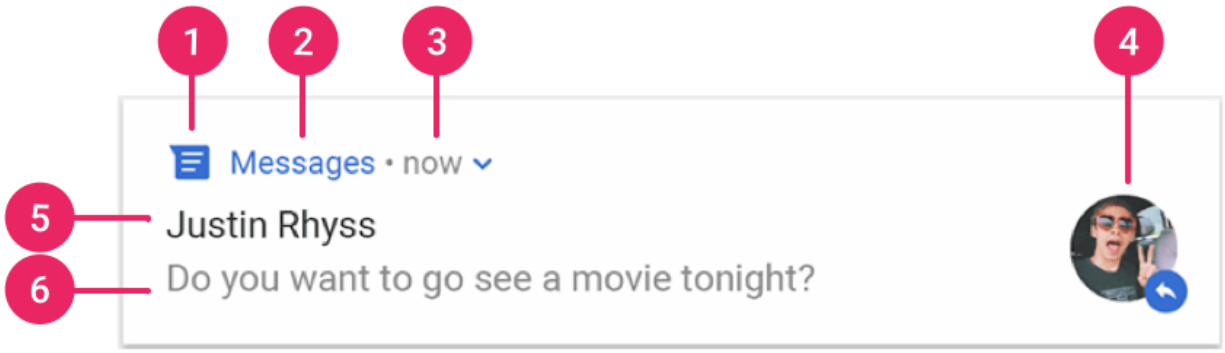

Abbildung 1: Notification Grundgerüst

Wie in Abbildung 1 zu sehen ist, hat eine normale Notification 6 Standardteile.

1. Kleines Icon: Wird benötigt und kann mit setSmallIcon() programmatisch gesetzt werden.
2. Name der App: Wird vom System bereitgestellt.
3. Zeitstempel: Wird normal von System gesetzt, kann aber überschrieben oder verborgen werden.
4. Großes Icon: Optional, kann mit setLargeIcon() gesetzt werden.
5. Titel: Optional, kann mit setContentTitle() gesetzt werden.
6. Text: Optional, kann mit setContentText() gesetzt werden.

Funktion

Aktionen einer Notification

Die wichtigste Aktion sollte immer die sog. Tab-Action sein. Das bedeutet, wenn auf die erweiterte Notification getippt wird, soll eine entsprechende Activity aufgerufen werden. Als Beispiel nehmen wir hier die YouTube App. Diese zeigt an, wenn ein Abonnierter Content Creator ein Video verfügbar gemacht hat. Sobald man auf diese Nachricht klickt, wird YouTube geöffnet und das Video abgespielt. Es könne aber noch weitere Aktionen ausgeführt werden. Dies wird über die Action Buttons am Ende der Notification gesteuert. Siehe Abbildung 2. Action Buttons sollte nicht die gleiche Funktion übernehmen wie die Erstgenannte Tab-Action.

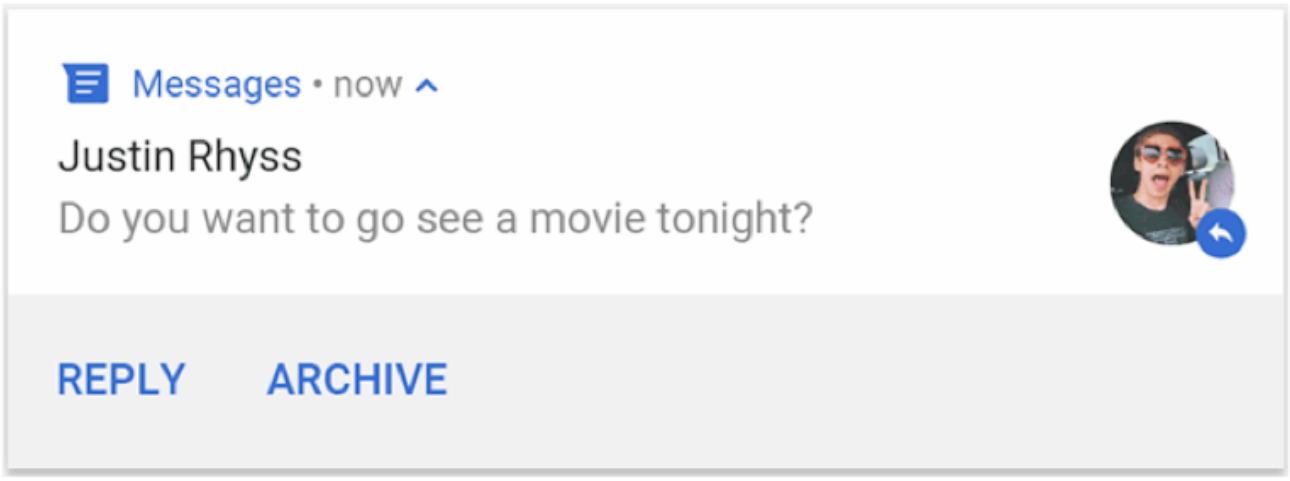

Abbildung 2: Erweiterte Notification mit Action Buttons

Erweiterung der Notification

Um Platz zu sparen sind die Notification grundsätzlich erst verkleinert. Erst wenn der Benutzer die Notification erweitert, wird der Rest angezeigt. Das kann dann mit verschiedenen Styles kombiniert werden. So z. B. wenn man den Inhalt einer E-Mail anzeigen will.

Notification Gruppen

Sollte eine App mehrere Notifications zu unterschiedlichen oder dem gleichen Zeitpunkt bereitstellen, kann es schnell zu einer Überfüllung kommen. Daher kann man diese in einer Gruppe zusammenfassen. Diese werden als Liste angezeigt. Sobald dies erweitert wird, werden alle Notifications individuell angezeigt.

Notification Channels

Seit Android Oreo (8) werden Notification Kanäle benötigt. Es muss mindestens ein Kanal pro App existieren. Andernfalls wird ab Android Oreo die Notification nicht angezeigt. Diese sind ein Weg, Notifications zu kategorisieren. Davor konnte man entweder alle Nachrichten einer App abschalten oder nicht. Mit den Kanälen lässt sich das nun individuell für den Kanal einstellen. Der Programmierer hat auch die Möglichkeit den einzelnen Kanälen unterschiedliche Priorität zuzuweisen.

Wichtigkeit

Es können Nachrichten Priorität zugewiesen werden. Das wirkt sich auf die Art aus, wie die Nachricht den Benutzer unterbricht. Je höher die Priorität desto größer der Störfaktor. Zum Beispiel wird bei einer niedrigen Priorität keine Nachricht auf dem Sperrbildschirm angezeigt.

Diese Priorität ist ab Android 8 an die des verantwortlichen Kanals gekoppelt.

Folgende allgemeine Möglichkeiten stehen zur Verfügung:

- Wichtig: Audio und Erweiterte Anzeige
- Hoch: Ton
- Medium: kein Ton
- Niedrig: kein Ton, erscheint nicht in der Status bar

Für die Feinabstimmung empfiehlt sich die Klasse NotificationManager, dort sind ein paar weitere Abstimmungen der Wichtigkeit enthalten.

Arten

Um vielen Einsätzen gerecht zu werden hat Android schon einige vordefinierte Styles und Verhalten implementiert. Hier werden kurz ihre Eigenschaften aufgelistet.

Default

Diese Notification enthält nur die Standardelemente. Diese reicht für kleinere Nachrichten, wie z. B. eine Kalenderbenachrichtigung.

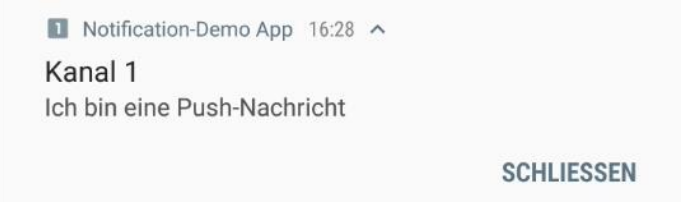

Abbildung 3: Standard Notification

Progress bar

Es wird ein Prozentualer Balken angezeigt. Sollte der Fortschritt erfolgt sein kann diese auch einen kontinuierliche Animation anzeigen.

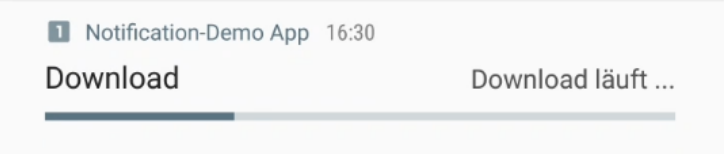

Abbildung 4: Fortschrittsanzeige

Big-PictureStyle

Hier wird in der eingeklappte Ansicht das Bild in Kleinformat angezeigt. Ebenso ein kleiner Text der zu dem entsprechenden Bild passt. Sobald die Notification erweitert wird, wird das Bild in großer Auflösung dargestellt. Der vorherige Text wird nun oberhalb des Bildes angezeigt.

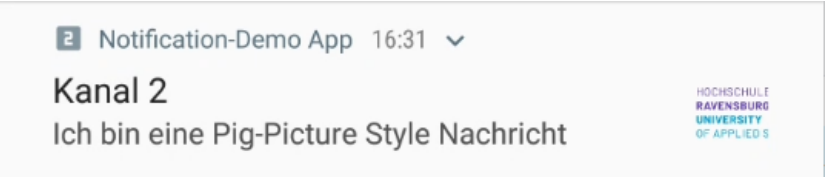

Abbildung 5: Eingeklappte Ansicht des BigPicture Style

Abbildung 6: Erweiterte Ansicht des BigPicture Style

Big-Text Style

Analog zum Big-Picture Style wird hier in der zusammengeklappten Variante nur ein kurzer Text dargestellt. Meist eine zusammenfassung des Langen Orginaltextes. Sobald die Notification erweritert ist, wird der gesammte Text dargestell.

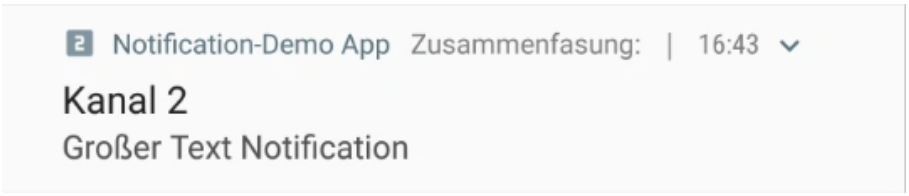

Abbildung 7: Eingeklappte Ansicht des Big-Text Style

Abbildung 8: Erweiterte Ansicht des Big-Text Style

Inbox Style

Eine Abwandlung des Big-Text Styles ist die Inbox Nachricht. Hier werden kleine Zusammenfassungszeilen angezeigt. Gut einsetzbar bei E-Mail-Benachrichtigungen.

Media-Controlls

Für die Kontrolle bei Mediaplayern gibt es ein extra Style. Dieser hat in der eingeklappten Variante drei Action Buttons, die er darstellt. Sobald die Notification ausgeklappt wird, können bis zu 5 dargestellt werden. Im Falle eines Musikplayers wären dies z. B. pausieren, nächstes oder Lautstärkekontrolle. Sobald die Notification mit einer Mediasession gekoppelt wird, färbt sie sich entsprechend dem Albumcover das als Großes Logo dargestellt wird.

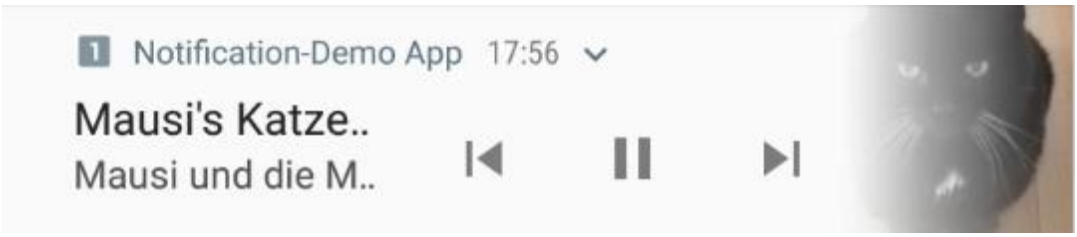

Abbildung 9: Media-Kontrollen in der eingeklappten Ansicht

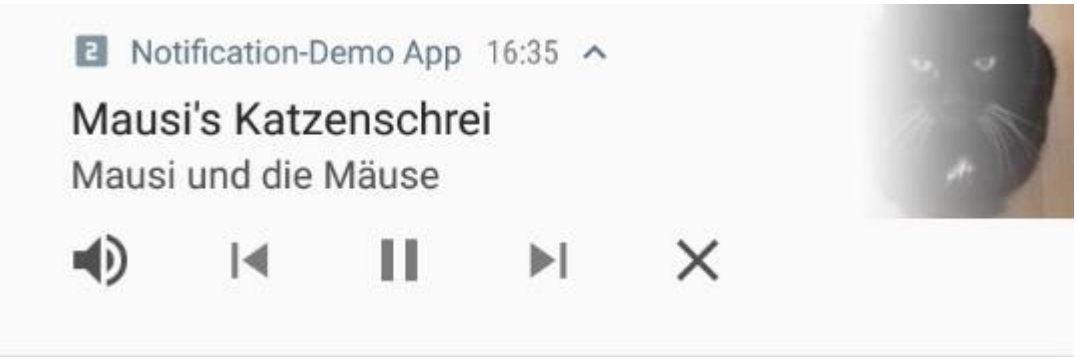

Abbildung 10: Media-Kontrollen in der erweiterten Ansicht

Direct Reply und Messaging Style

Dies ist der Beste Style für Messanger Apps. Hier werden bis zu sieben Zeilen einer Konversation in der erweiterten Ansicht dargestellt. Es wird ein Gruppenname oberhalb dessen angezeigt. Wichtig ist hierbei, dass Android das Vorhandensein eines Titels, diesen als Hinweis nimmt das dies eine Gruppe ist. Des Weiteren kann einem der normalen Action Buttons eine spezielle Funktion zugewiesen werden. Diese nimmt den vom Benutzer eingegebenen Text und leitet ihn intern an die richtige Stelle weiter. Das wäre im Falle von z. B. WhatsApp das Senden dieser Nachricht an den entsprechenden Kontakt.

Abbildung 11: Messaging Style mit Reply und Schließen Button

Custom Style

Wenn keiner der vorher genannten Styles zu der eigenen App passt, kann man mit RemoteViews eigene Styles erstellen. In diesen kann man ganz normal XML Layouts erstellen, alle normalen Elemente sind möglich. Zu beachten sind die Einschränkungen der RemoteViews. Es gibt außerdem die Möglichkeit, um seine Eigene Nachricht einen Style zu legen, der die Nachricht konsistenter mit dem restlichen System macht. Android empfiehlt dies.

Abbildung 12: Custom Notification eingeklappt

Guidelines für Notifications

Gemäß der Blogseite MobiLoud werden 71% aller App Deinstallationen durch Notifications ausgelöst. Das ist ein recht hohe Anzahl, die recht einfach zu verhindern ist. Ein anderer Blog beschreibt auch, wie diese Dinge einfach umzusetzen sind. CleverTrap schreibt in ihrem Blog, es komme auf drei Wichtige Punkte an.

1. Relevant und zeitgerecht
2. Persönlich
3. Aktionsgerecht

Diese drei Punkte sind die Hauptmerkmale einer Notification.

Was zu beachten ist bei der Entwicklung einer App

Allgemeines

Wann empfiehlt es sich nun eine Notification zu senden? Sie sollten auf keinen Fall der Haupt Kommunikationskanal zum Benutzer sein. Wie erwähnt, sind zu häufige Unterbrechungen durch eine irrelevante Nachricht irritierend. Unter diese Kategorie fallen:

- Werbung für andere Apps in Notifications (Play Store verbietet dies)
- Eine noch nie geöffnete App
- Anfrage einer Bewertung der App
- Informationen bei denen der Nutzer nicht interagieren muss, z.B. automatische Synchronisierung mit Cloud
- Aufmerksamkeitsgehabe, wie „Lange nichts von dir gehört…“
- Fehlermeldungen die nicht ein Eingreifen des Benutzers benötigen

Es gibt auch einen Fall der laut Material Design eine Notification zwingend erfordert. Dies wäre bei einem Foreground Service der Fall. Dieser läuft im Hintergrund ohne Benutzerinteraktion. Während dieser läuft, sollte eine Notification als Anzeige genutzt werden. Sie sollte nicht wegwischbar sein, jedoch immer noch eine Möglichkeit den Service abzubrechen geben. Ein Beispiel wäre hier der Download Balken einer App oder Inhalte aus dem Internet.

An dieser Stelle sei auf die Seite CleverTrap.com verwiesen. Diese enthält viele Nützliche Beispiele für gute Notifications basierend auf den drei Hauptpunkten.

Notification Kanäle

Seit Android 8 werden Kanäle benötigt. Diese sind ein guter Weg, um dem Benutzer eine differenzierte Auswahl über seine Benachrichtigungen zu geben. Er kann bei diesen nach Belieben den anzeigepunkt Ändern. So empfiehlt es sich bei einer App, die sehr wichtige und vielleicht weniger wichtige Benachrichtigungen in zwei separate Kanäle mit unterschiedlicher Priorität zu setzen. Dadurch kann der Benutzer selbst entscheiden welchen Nachrichten er benötigt oder nicht.

Ein weiterer Punkt ist bei den Kanälen die Beschreibung. Mit „Kanal1“ als Beschreibung weiß der Benutzer nicht, was dieser Kanal tut. Besser wäre hier eine kurze Beschreibung was dieser Kanal macht. Z. B. „Informiert über kritische Ereignisse“. Ist der Kanal Name Eindeutig, kann man die Beschreibung auch weglassen. Siehe WhatsApp. Es hat keine Beschreibung der Kanäle, allerdings einen Eindeutigen Namen pro Kanal.

Notifications

Bei den Notifications sind die drei wichtigsten Punkte Relevanz/Zeitnah, dem Benutzer entsprechend und Aktionsgerecht. Solang diese eingehalten werden kann man den Benutzer eigentlich nicht vertreiben. Desweitern sind weitere Tipps aufgeführt, um das Interesse des Nutzers beizubehalten.

Relevant und zeitgerecht

Das allerwichtigste bei einer Notification ist, den Nutzer nicht mit unnötigen Mitteilungen zu bombardieren. Dies dürfte einer der Hauptgründe sein, wenn Benutzer die App aufgrund der schlechten Notifications deinstallieren. Es muss geschaut werden, wann es sinnvoll ist, den Benutzer über Vorgänge in der App zu informieren.

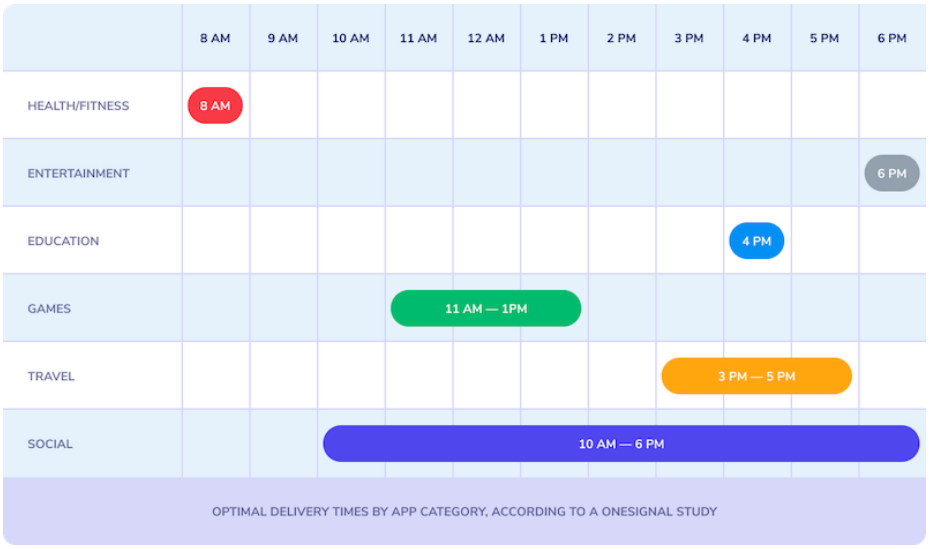

Abbildung 13: Optimale Zeiten für Benutzer eine Notification zukommen zu lassen

So empfehlen sich Alarme zum Beispiel direkt zu triggern. Oder sollte die Nachricht über ein Ereignis außerhalb der eigenen Zeitzone informieren muss erst geschaut werden in welcher der Nutzer sich befindet. Beispiel wäre hier ein Sale eines Shops in Amerika, der Nutzer ist jedoch in Europa. Negativ ist es, wenn der Benutzer zu spät über Änderung informiert wird. Jeder Zocker ärgert sich darüber, nicht mitbekommen zu haben, dass die eigene Basis angegriffen wird. Frustrierend und kann auch dazu führen, dass der Nutzer das spiel nach einer Weile nicht mehr aktiv spielt.

Persönlich

Hierauf sollte gut Acht gegeben werden. Viele Nutzer sehen das Handy heute als Helfer an. Wenn die App mit persönlichen Daten zu tun hat, sollten sich diese auch in ihren Nachrichten widerspiegeln. Das kann der Benutzername sein, einbinden von Nutzerdaten wie Musikgeschmack und eine persönlich klingende Anrede. Bei einer Fitness App könnte man z. B. schreiben „Hey, du hast heute dein Tageslimit erreicht. Super!“. So fühlt sich der Nutzer motivierter Sport zu machen und gleichzeitig die App weiterhin zu nutzen.

Aktionsgerecht

Wichtig ist, den Benutzer in der Notification an der Hand zu nehmen. Das bedeutet es muss immer klar sein, was mit dieser Mitteilung ausgesagt werden will. Action Buttons klare Beschreibungen zuweisen. Den Text aussagekräftig formulieren. Z. B. bei einem Kalender wäre „Alarm“ wäre nicht ausreichend. Es sollte dem Anwender hier ganz deutlich klar gemacht werden um welchen Termin es sich handelt. Desweitern sei hier nochmal erwähnt, nicht die gleiche Tab Action auf einen Action Button zu setzen. Das führt nur zu Irritation.

Geh dem Benutzer nicht auf die Nerven

Dieser Tipp ist sehr selbstverständlich. Respektiere deinen Benutzer! Keiner will bei jedem bisschen Änderung der App eine Mitteilung bekommen. Dies wird am besten erreicht, wenn man das Attribut OnlyAlertOnce verwendet, sollte die Notification ständig aktualisiert werden. Gutes Beispiel ist hier WhatsApp. Sobald in einer Gruppe eine Nachricht gesendet wurde, wird eine Notification getriggert. Diese

hat Sound. Bei der nächsten Nachricht, diese kann auch 2 Sekunden später eintreffe, wird dann kein Sound mehr abgespielt. Der Benutzer wird als nur einmal „gestört“.

Strukturiert und präzise

Notifications sind kurzlebig. Deshalb sollten Texte so präzise wie möglich formuliert werden. Dabei sollten möglichst immer die schon vorhanden Stile genutzt werden. Ein Messanger dienst am besten MessagingStyle, die E-Mail-App den BigText Style. So bleibt die Erfahrung über die Geräte gleich und der Nutzer weiß genau was im mitgeteilt werden soll.

Implementierungsbeispiele

Auf der Android-Developer Seite findet man ausführliche Guides, wie Notifications erstellt werden können. Diese sind leider nicht aktuell, manchmal muss man schauen welche der Android-Packages benötigt werden. Die hier aufgeführten Codebeispiele sind mit der androidx-Library erstellt. Hier werden exemplarisch zwei Arten von Notification gezeigt. Wichtig ist hierbei das es sich nur um die Notification selbst handelt, nicht darum wie und wann sie eingesetzt wird.

Achtung! Diese Codeschnipsel sind für die Sprache Java, nicht Kotlin. Entsprechende Codestellen findet man auf der Android-Developer Seite.

Wenn der Code kopiert werden soll, ist der einfachste Weg, den gesamten Code über das in der Einleitung erwähnte GitHub-Repo zu holen. Die README enthält alle wichtigen Details.

Grundlegendes

Bei der Implementierung sollte darauf geachtet werden, dass diese Notifications sehr schnell extrem lang werden. Zu empfehlen wäre hier ein Builder-klasse, die nur die Notifications zusammenbaut. Beachtet das die Notifications immer eine Eindeutige ID brauchen, sonst überschreibt sich einfach deren Inhalt und der Broadcast Receiver kann diese nicht genau unterscheiden. Man sollte hier daran denken, die Vorgaben im Kapitel Guidelines für Notifications, hier auch umzusetzen. Daher ist zu empfehlen, die Dokumentation von Android genau zu studieren, damit der Code perfekt auf den jeweiligen Benutzerfall zugeschnitten ist.

Notification-Kanal

Die Überprüfung, ob Android 8.0 oder höher kann auch weggelassen werden, wenn darunterliegende Versionen nicht unterstütz werden sollen. Richtlinien zur Namensgebung sollten beachtet werden.

Die ChannelID, der Name und die Beschreibung sind alles Strings. Die Importance ist eine Konstante der NotificationManager Klasse. Diese kann frei nach Bedürfnissen gewählt werden.

Code

```
//der if prüft ob wir android 8.0 oder höher haben, sonnst muss er kein channel erstellen.
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    NotificationChannel channel = new NotificationChannel(CHANNEL1_ID, channel1Name, importance);
    channel.setDescription(channel1Description);
    //Channel dem System mitteilen. Danach nicht mehr veränderbar
    NotificationManager notificationManager = context.getSystemService(NotificationManager.class);
    notificationManager.createNotificationChannel(channel);
}
```

Normale Notification inklusive Action Buttons

Erstellung einer einfachen Push-Notification mit Titel, Tab-Action, Inhalts Text, Icon und einem Action-Button. Es muss bei der Action kein Icon spezifiziert werden. Bei der Unterscheidung der PendingIntent's aufpassen. Diese müssen eindeutig unterscheidbar sein, z. b. eine eindeutige Action.

Code

```
Notification defaultNot = new NotificationCompat.Builder(context, channelid)
        .setSmallIcon(icon)
        .setContentTitle(contentTitle)
        .setContentText(context.getString(R.string.NotDefContent))
        .setPriority(NotificationCompat.PRIORITY_DEFAULT)
        .setContentIntent(buildContentIntent())
        .addAction(R.drawable.ic_launcher_foreground,
context.getString(R.string.NotActionClose), buildDismissIntent(notID))
        .setAutoCancel(true) // Lässt die Nachricht nicht verschwinden bis auf
sie geklickt wird
        .setGroup(NOTIFICATION_GROUP_KEY)
        .build();
return defaultNot;
```

Messaging Style mit Reply Button

Erstellen einer Messaging-Style Notification mit Antwortmöglichkeit (siehe WhatsApp). Diese erfordert weitere Einstellungen um zu Funktionieren. Wichtig ist das der ConversionTitle im Style nicht für Chats unter drei Personen verwendet werden soll. Ebenfalls wichtig ist das der PendingIntent des RemoteInput immer einzigartig ist, sonst könnte es sein das der User einem anderen Chat die Nachricht schreibt als er annimmt.

Bei dieser Notification müssen noch ein paar weitere Einstellungen und Objekte angelegt werden. Bestenfalls in dieser Reihenfolge.

RemoteInput

Code

```
//RemoteInput, anhand dessen wird er eingegebene Text später entnommen
RemoteInput remoteInput = new RemoteInput.Builder(KEY_TEXTRPLY)
        .setLabel(context.getString(R.string.NotRplyLabel))
        .build();
```

Action Button mit RemoteInput

Code

```
//Antwort ActionButton, dem der RemoteInput angehängt wird.
Intent rplyIntent = new Intent(context,
NotificationReceiver.class).setAction(ACTION_REPLY).putExtra(PAYLOAD,notID);
PendingIntent rplyPendingIntent = PendingIntent.getBroadcast(context, notID,
rplyIntent, PendingIntent.FLAG_UPDATE_CURRENT);
NotificationCompat.Action rplyAction = new
NotificationCompat.Action.Builder(R.drawable.ic_reply,
context.getString(R.string.NotActionReply), rplyPendingIntent )
        .addRemoteInput(remoteInput)
        .build();
```

Messaging Style

Die Zeile, an der die Message hinzugefügt wird, Muss für jedes Mitglied des Gruppenchats aufgerufen werden. Das nächste Beispiel veranschaulicht dies.

Code

```
//MessangingStyle, wichtig hier das Person Objekt. Nur eine Charsequence ist in
der alten Funktion.
NotificationCompat.MessagingStyle messagingStyle = new
NotificationCompat.MessagingStyle(me);
messagingStyle.setConversationTitle(context.getString(R.string.MessageTitle));
messagingStyle.addMessage(notMessage); // Fügt die Nachricht dem Style hinzu
```

Die Message erstellt man separat. Diese kann bei einer Chat-App aus vorhandenen Listen genommen werden. Bei diesem Beispiel wird eine for-each Schleife auf eine Liste angewendet.

Die Message benötigt einen Text, einen Zeitstempel und den Sender der Nachricht. Diese Informationen werden dann in der Notification dargestellt.

Code

```
for (Message chatMessage : NotificationDemoApplication.MESSAGES){
    NotificationCompat.MessagingStyle.Message notMessage = new
NotificationCompat.MessagingStyle.Message(
            chatMessage.getText(),
            chatMessage.getTimestamp(),
            chatMessage.getSender()
    );
    messagingStyle.addMessage(notMessage); // Fügt die Nachricht dem Style hinzu
}
```

Notification

Code

```
//Eigentlich Notification bauen
Notification rplyNot = new NotificationCompat.Builder(context, channelid)
        .setSmallIcon(icon)
        .setContentIntent(buildContentIntent())
        .setStyle(messagingStyle)
        .addAction(rplyAction)
        .addAction(R.drawable.ic_launcher_foreground,
context.getString(R.string.NotActionClose), buildDismissIntent(notID))
        .setColor(context.getColor(R.color.Primary))
        .setPriority(NotificationCompat.PRIORITY_DEFAULT)
        .setAutoCancel(true)
        .setOnlyAlertOnce(true)
        .setGroup(NOTIFICATION_GROUP_KEY)
        .build();
```

Notification Anzeigen

Sämtliche Notifications werden, sobald sie gebaut werden, dann über den NotificationManager dem System mitgeteilt. Dazu muss eine Variable vom Typ NotificationManager angelegt werden. Dieser verwaltet alle Notifications. Mit dem Aufruf cancel() wird die Notification gelöscht.

Code

```
notificationManager.notify(notID, notification);
```

Code

```
notificationManager.cancel(notID);
```

Fazit

Wenn man eine App schreibt, sollte man sich zwingend mit den Notifications beschäftigen. Sie sind ein mächtiges Tool, um den Nutzer auch außerhalb der UI informiert zu halten. Ihr Nutzen hängt sehr stark damit zusammen, wie gut sie umgesetzt werden. Wie in Material Design beschrieben werden sie bei bestimmten Anwendungstypen vorausgesetzt. Meist sind sie aber einfach ein guter Kanal, um mit dem Benutzer zu kommunizieren.

Quellen

Code

- https://developer.android.com/guide/topics/ui/notifiers/notifications
- https://www.raywenderlich.com/1214490-android-notifications-tutorial-getting-started
- https://www.vogella.com/tutorials/AndroidNotifications/article.html
- https://stackoverflow.com/questions/56457402/error-cannot-find-symbol-class-mediastyle-after-migrating-to-androidx/56596662
- YouTube-Kanal: Coding in Flow, Playlist Notifications & Notification-Channels

Best Practices

- https://www.mobiloud.com/blog/push-notifications-best-practices
- https://clevertap.com/blog/push-notification-best-practices/
- https://material.io/design/platform-guidance/android-notifications.html#usage
- https://medium.com/kayvan-kaseb/some-best-practices-in-using-android-notification-a957f9245278
- https://onesignal.com/blog/6-best-practices-for-push-notifications/
- https://tracker.my.com/blog/103/10-tips-tricks-to-master-push-notifications-in-mobile-games?lang=en
- https://www.internetworld.de/sonstiges/push-nachrichten/5-tipps-push-nachrichten-1752368.html

Abbildungen

Abbildungen 1 & 2: https://developer.android.com/guide/topics/ui/notifiers/notifications

Abbildungen 3-12: Screenshot der Demo App, https://github.com/Cayleb-Ordo/Notification-Demo

Abbildung 13: https://onesignal.com/blog/6-best-practices-for-push-notifications/