

Notifikace u OS Android a iOS

Bc. Ondřej Freisler
FRE0025

Odkazy na prezentace

1. Zadání:
<https://docs.google.com/presentation/d/18J6PTIt01w5DNR0wqWCuPRHqYZBBKWqd2Ujh5XINTG0/edit?usp=sharing>
2. Teoretická část:
https://docs.google.com/presentation/d/1Ddg9p13wlgYLuU_qGJ8-ztI8wf9Vp4cAKlGtwLsms84/edit?usp=sharing
3. Praktická část:
https://docs.google.com/presentation/d/1wCPSBd_8iLsuPUiIRYSONavini61YcljP4cwrHhBIgE/edit?usp=sharing

1. Úvod

Notifikace je zpráva, která je zobrazena mimo API konkrétní aplikace jednak na stavovém řádku mobilního zařízení prostřednictvím ikony a detailněji pak také na tzv. notifikační liště daného operačního systému. Jejím účelem je poskytnout uživateli rychlé a snadno dostupné připomenutí respektive upozornění. Uživatel následně může klepnutím na toto oznámení otevřít přidruženou aplikaci nebo provést odpovídající akci.

Obsah těchto upozornění může být velmi různorodý, neboť každá aplikace je jiná, slouží k něčemu jinému a chce poskytovat uživateli OS různé informace. V základu se notifikace dělí na místní a vzdálené. Místní jsou integrované přímo v OS a k jejich použití není zapotřebí žádná služba třetích stran. Vzdálené zase vyžadují komunikaci se serverem Google nebo Apple.

V případě lokálních notifikací se mohou informace týkat například náhledu na nejnovější zprávu z SMS komunikace, upozornění na příchozí/nepřijatý hovor, nastavený budík, událost v kalendáři, sledování GPS polohy či aktuální přehrávání médií. Naproti tomu vzdálené notifikace slouží jako prostředek pro komunikaci s uživatelem v rámci aplikací třetích stran.

Předmětem této semestrální práce je nejprve zpracovat téma notifikací z teoretického hlediska, kde zpočátku popisují, jak konkrétně se řeší problematika notifikací u OS Android a iOS, dále napojení na notifikační servery z jiných aplikací a nakonec porovnání dříve zmíněného z hlediska bezpečnosti, nákladů a požadavků na provoz.

Praktická část tvoří druhou část práce, ve které již pracuji s mobilním koncovým zařízením OneNote 7T, pro které testuji notifikační funkce. Pro upřesnění jsem vyvinul aplikaci pro OS Android, která demonstruje různé typy notifikačních zpráv, které mají různou strukturu a různou dobu doručení na notifikační lištu. Dále se zaměřuji na řešení zobrazení fronty notifikací při zamčené obrazovce displeje zařízení po určitém čase.

2. Teoretická část

2.1. Princip notifikací

Přestože se mechanismy pro doručování oznámení pro OS Android a iOS liší, základní architektura je pro každé zařízení prakticky stejná:

OS udržuje na pozadí trvalé připojení k serveru. Toto připojení využívá bezvýznamné množství energie a spotřeby dat, neboť neexistuje žádný souvislý datový provoz, dokud si aplikace data sama nestáhne. Komponenty, které vývojářům aplikací umožňují odesílat informace ze svých serverů do zařízení uživatele se nazývají mobilní oznamovací služby (MNS - Mobile Notification Services). V závislosti na OS to může být ve dnešní době jmenovitě Firebase Cloud Messaging (FCM) pro Android (nástupce Google Cloud Messaging) nebo Apple Push Notification Service (APNS) pro iOS.

Klíčové je uvést, že **notifikaci iniciuje vždy server**, nikoliv uživatel zařízení.



1. Server, přidělený aplikaci vývojáře (na obr. jako Provider), odešle notifikační požadavek na server MNS.
2. Pomocí tohoto připojení MNS přenáší tento požadavek na zařízení uživatele.
3. OS tohoto zařízení pak kontroluje přijaté informace, aby je porovnal s aplikacemi, instalovanými v zařízení a zobrazil je tou odpovídající.

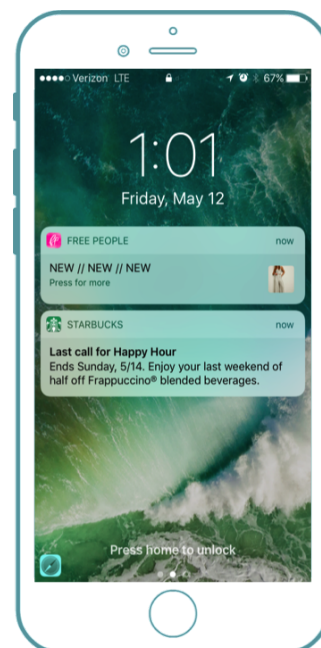
Výhod notifikací je mnoho jak pro uživatele, tak pro vývojáře aplikací. Z technického hlediska podpora nesledovaného připojení k centralizovanému mechanismu mobilních oznámení znamená prodloužení životnosti baterie a efektivní využití prostředků koncových zařízení. To vše kvůli tomu, že MNS potřebují jediné ověřené připojení k síti a cloudu OS narozdíl od spouštění aplikací na pozadí.

2.1.1. Push notifikace

Notifikace se uživateli mohou zobrazovat bez ohledu na to, zda je přidružený program právě spuštěn nebo je neaktivní. V takovém případě se jedná o tzv. push notifikace.

Push notifikace se zobrazují jak na stavovém řádku, tak i na zamykací obrazovce. Po rozbalení tohoto stavového řádku gestem se uživateli zobrazí tato notifikace v rozšířeném (detailnějším) zobrazení a u některých druhů notifikací jsou zde možné i dodatečné rychlé akce - většinou v podobě tlačítek - např. rychlá odpověď, označení zprávy jako přečtené, pozastavení přehrávaných médií, vypnutí právě zvonícího budíku, atd.

Z toho vyplývá, že se push notifikace nejlépe uplatňují pro informování na obsah závislý na aktuálním čase.

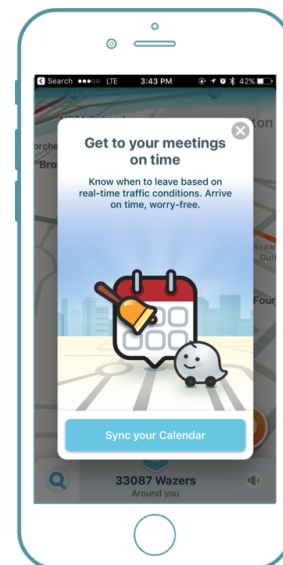


2.1.2. “In-app” notifikace

Druhým typem notifikací jsou tzv. in-app notifikace, které se objeví pouze v případě, pokud je přidružená aplikace právě používána. Tento typ notifikací typicky poskytuje informace, které uživateli asistují s používáním určité funkce v aplikaci.

V porovnání s push notifikacemi navíc nemá tento typ žádná omezení velikosti zobrazení ani vizuálního rozpoložení.

Pokud si tedy uživatel stáhl například aplikaci obchodního řetězce, která však využívá pouze in-app notifikace, nemusí se dozvědět o všech slevových akcích, pozměněné otevírací době, atd., dokud aplikaci sám neotevře.



2.2. Řešení notifikací v rámci OS Android

Počínaje Androidem verze 5.0 se mohou oznámení objevit v plovoucím okně - tzv. *heads-up upozornění*. Zobrazují se však pouze na okamžik a po chvíli zmizí s tím, že se jejich viditelnost přesune do notificační lišty. Takto OS zobrazuje důležitá oznámení, která by měl uživatel okamžitě vědět a věnovat jim svou pozornost. Mezi podmínky, které mohou způsobit zobrazení heads-up notifikací patří zejména:

- Uživatel se právě nachází se spuštěnou aplikací ve fullscreen módu,
- Oznámení má vysokou prioritu a zároveň využívá vyzvánění či vibrace (od systému Android 7.1),
- Notifikační kanál má velkou prioritu (od Android 8.0).

2.2.1. Notifikace na zamykací obrazovce

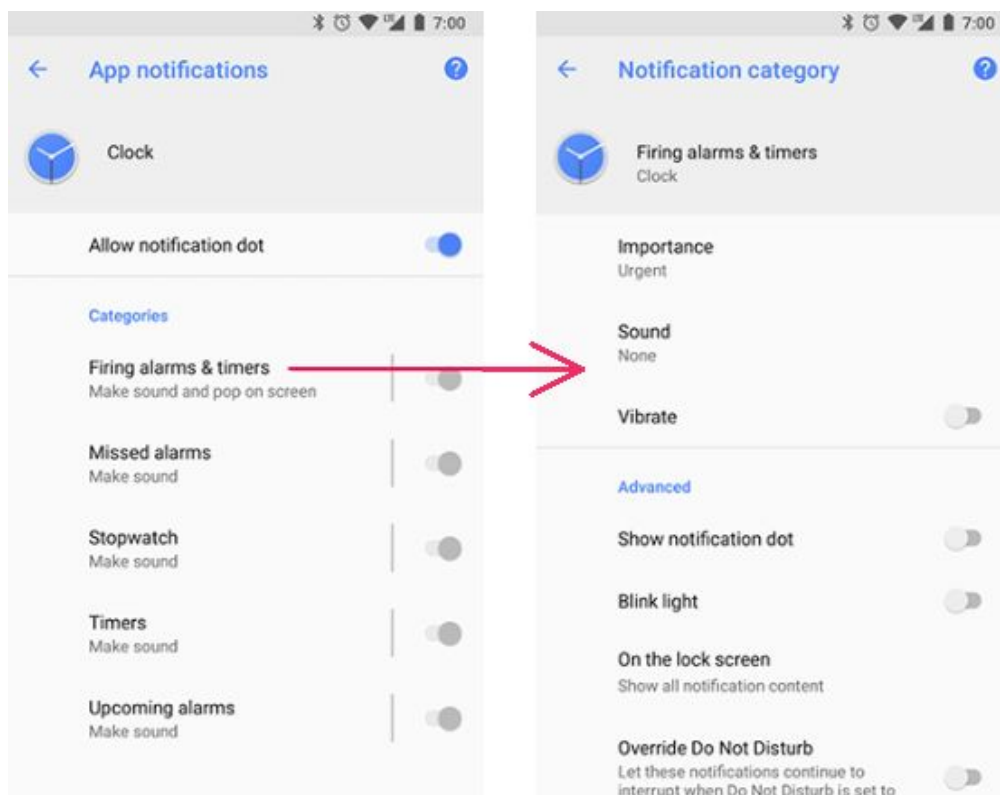
Od verze Android 5.0 se mohou notifikace zobrazovat i na zamykací obrazovce zařízení. Při vývoji aplikace lze nastavit jednak úroveň podrobností, které jsou na zamykací obrazovce viditelné a nebo dokonce zda se vůbec mají zobrazit.

Co se týče systémového nastavení, uživatel může i zde nastavit úroveň viditelných podrobností včetně celkového zakázání. Dále je zde volba filtrovat oznámení pro každý notificační kanál (Android 8.0 a výše).

2.2.2. Notifikační kanály

Od Android 8.0 musí být všechna oznámení přiřazena do tzv. kanálů, jinak se nezobrazí. Dlouhým stisknutím notifikace lze pak měnit chování jejího přidruženého kanálu. Tímto způsobem mohou uživatelé spravovat chování konkrétních kanálů namísto správy veškerých oznámení pro konkrétní aplikaci. Tato správa probíhá opět v nastavení samotného OS. Kanálům s odpovídajícími oznámeními lze také nastavovat důležitost.

Oproti tomu uživatelé starších Android systémů (API 25 a nižší) spravují veškerá oznámení v jednom společném “seznamu” bez rozdělení do kanálů.



2.2.3. Důležitost oznámení

Důležitost určuje, do jaké míry má oznámení uživatele vyrušit (vizuálně či zvukově) v právě prováděné činnosti. Opět na základě verze OS Android - od 8.0 se mění důležitost pro kanál, kdežto pro starší pro každé oznámení zvlášť - vše opět v systémovém nastavení.

Úrovně důležitostí jsou následující:

- Naléhavé: Oznámení vydá zvuk a zobrazí se jako heads-up upozornění,
- Vysoké: Vydá zvuk,
- Střední: Nevydá zvuk,
- Nízký: Žádné vyrušení uživatele.

Všechna oznámení, bez ohledu na důležitost, se zobrazují v nerušivých místech UI, tzn. v notificační liště a jako ikona na stavovém řádku.

2.2.4. Situace v režimu “Nerušit”

Od verze 5.0 existuje pro Android režim “Nerušit”, který umlčí zvuky a vibrace celého zařízení. To platí i pro veškerá oznámení bez ohledu na jejich důležitost. Jejich vizuální zobrazování nicméně zůstává defaultně nezměněno.

2.3. Architektura notifikací pro OS Android

Podle oficiálních dokumentací byla služba Google Cloud Messaging roku 2018 deaktivována a odstraněna z API rozhraní společnosti Google. Ekvivalentní funkce nyní zajišťuje multiplatformní služba Firebase Cloud Messaging (FCM).

Ta umožňuje klientské aplikaci odesílat zprávy s notifikacemi. Každá zpráva může obsahovat až 4 kB užitečného zatížení.

Implementace FCM v sobě zahrnuje 2 hlavní komponenty pro odesílání a příjem:

1. Důvěryhodné prostředí, jako je Firebase Cloud nebo aplikační server, na kterém lze sestavovat a odesílat notifikace.
2. Klientská aplikace přijímající notifikace.

Notifikace jsou odesílány prostřednictvím Firebase Admin SDK nebo serverových FCM protokolů. FCM umožňuje posílat push notifikace přímo z Firebase konzole, z aplikačního serveru nebo jiného důvěryhodného prostředí podporující logiku serveru. Aplikace musí nicméně být nejprve registrována u FCM.



1. Klientská aplikace odešle ID odesílatele, API klíč a ID aplikace FCM,
2. FCM vrácí klientské aplikaci registrační token,
3. Klientská aplikace poté odešle registrační token na server aplikace.

Pro použití FCM je tedy nutné jej kromě jeho zakomponování do mobilní aplikace implementovat také na straně serveru. Také je nutná implementace odeslání registračního ID odesílatele na server.

2.4. Řešení notifikací v rámci OS iOS

U OS iOS se situace velmi podobá Androidu. Notifikace jsou viditelné na zamykací obrazovce, v horní části obrazovky při aktivním používání zařízení a v Notifikačním centru.

Pro správu chování jednotlivých notifikací může uživatel navštívit systémová nastavení pod položkou Oznámení (Notifications). Uživatel zde může oznámení pro každou konkrétní aplikaci zvlášť zapnout/vypnout, povolit viditelnost oznámení v Notifikačním centru nebo na zamykací obrazovce, povolit tzv. odznaky u ikony aplikace a možnost zvolit si z následujících stylů oznámení:

- Banner: Notifikace se objeví v horní části obrazovky po několik sekund během aktivního používání zařízení a pak zmizí.
- Alert: Notifikace se objeví v horní části obrazovky během aktivního používání zařízení a zůstane tam, dokud jí uživatel nevěnuje pozornost.

Při prvním použití aplikace je uživatel u každé aplikace dotázán systémem, zda chce povolit této aplikaci přijímat notifikace. Pokud odpoví, že nechce, své nastavení může kdykoliv změnit později v systémovém nastavení.

2.4.1. iOS notifikace a gesta

Poklepáním na notifikaci při odemčeném zařízení (resp. swipnutím na stranu při zamčeném zařízení) iOS notifikaci odmítne, odebere jej z centra notifikací, otevře příslušnou aplikaci a zobrazí odpovídající informaci, kterých se notifikace týkala. Notifikace se v OS iOS mohou obdobně jako u OS Android “ukládat” do skupin - pokud dané aplikaci přijde notifikací více. Klepnutí na tuto skupinu pak zobrazí jednotlivé notifikace. Každá notifikace může obsahovat i v OS iOS detailnější pohled, který poskytuje více informací a nabízí až 4 tlačítka pro reakci.

Přejetím prstu doleva přes oznámení lze zase spravovat upozornění z dané aplikace a nakonec stisknutím a podržením lze s oznámením provádět rychlé akce, pokud to daná aplikace dovoluje).

2.4.2. Oznamovací centrum

Zde je zobrazena historie oznámení - lze se v něm vrátit v čase a podívat se, co uživateli uniklo. Jedná se o obdobu notifikační lišty u OS Android.

Notifikace lze i zde opět spravovat, kde každé je možné nastavit:

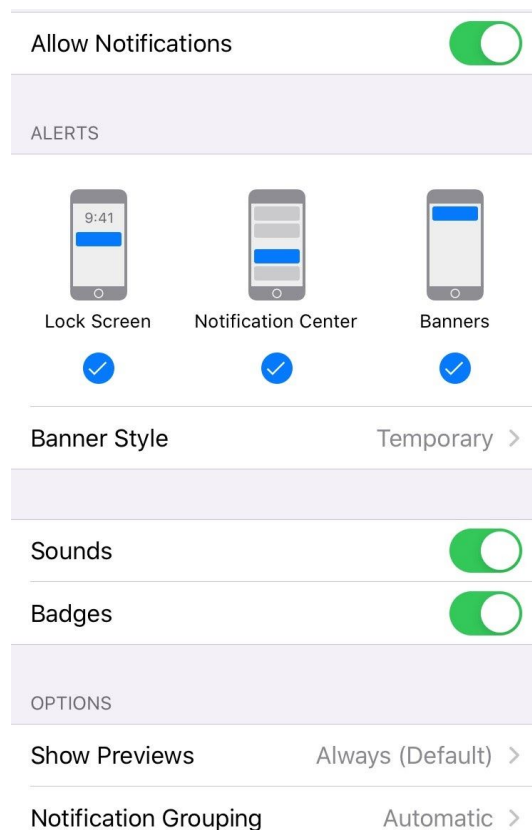
- Doručovat potichu - Oznámení se zobrazí v Oznamovacím centru, ale nezobrazí se na zamčené obrazovce, nepřehrají se zvuky, nezobrazují se bannery ani odznaky na ikoně aplikace.
- Vypnout... - Tato volba vypne veškerá oznámení z konkrétní aplikace.

2.4.3. Změna stylů upozornění

Každé aplikaci lze také nastavit styl zobrazování jejích notifikací. Toho je možné docílit pomocí změny stylů upozornění, které jsou dostupné v Nastavení -> Oznámení.

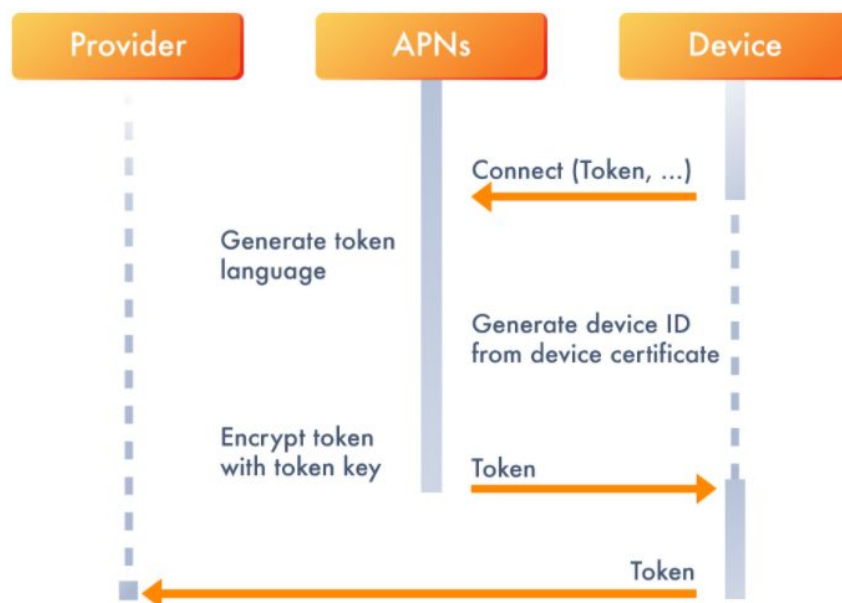
V části Styl oznámení se následně vybere požadovaná aplikace, které chceme upravit styl a zobrazí se obrazovka, uvedená na obrázku vpravo.

Kromě stylu zobrazení notifikace lze dodatečně upravovat i nastavení zvuku a odznaku společně se změnou nastavení oznámení pro skupinu.



2.5. Architektura notifikací pro OS iOS

Pro OS iOS se mechanismy doručování notifikací příliš neliší od Androidu. Funkcí, typickou především pro iOS, jsou počty badgů, které se zobrazují u konkrétní aplikace. Tuto funkci lze integrovat také do OS Android pomocí integrování určitých nástrojů či aplikací, nahrazujících ty systémové. Push notifikace jsou v systému iOS poháněna službou APNS, jak již bylo uvedeno v kapitole 2.1.



Služba APNS je základem pro doručování push notifikací na všechna zařízení, která kdy společnost Apple Inc. vydala. Je kompatibilní se zařízeními iOS, watchOS, tvOS a macOS.

Registrace zařízení pomocí APNS proběhne automaticky, jakmile je na zařízení uživatele daná aplikace nainstalována. Po procesu registrace obdrží vlastník aplikace jedinečný token zařízení - klíč pro kombinaci aplikace a zařízení, který identifikuje konkrétní aplikaci, spuštěnou na konkrétním zařízení. Tento token umožňuje službě APN doručovat zprávy pouze do zařízení, která by jej měla přijímat.

Po registraci vytvoří APNS šifrované a stabilní IP spojení mezi aplikačním serverem a APN. Po navázání tohoto spojení získá vlastník aplikace přístup k nastavení, které mu umožní notifikace odesílat a přijímat.

Pro zobrazení příchozí notifikace nemusí být aplikace spuštěna, neboť ji zpracuje připojení s APN, které probíhá na pozadí. Pokud je zařízení vypnuté, služba APN si notifikaci ponechá a pokusí se ji doručit později.

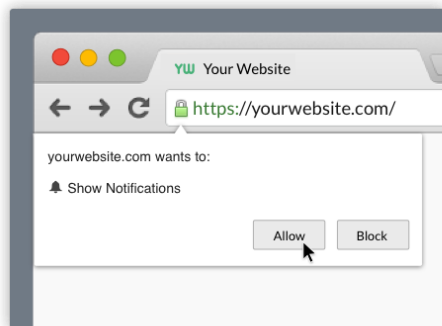
Pro práci s APNS musí mít vlastník aplikace založený profil a musí zprovoznit SSL certifikát pro svůj aplikační server. Ten je z hlediska kybernetické bezpečnosti důležitou součástí jakéhokoliv softwaru, který přenáší osobní údaje.

2.6. Notifikace z jiných platforem

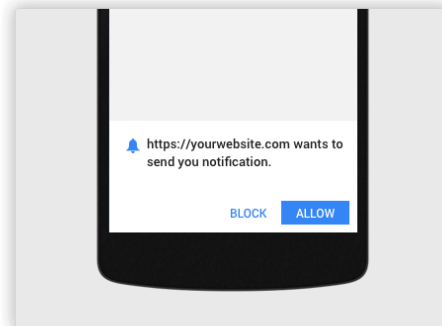
Kromě implementací notifikačních služeb do samotných aplikací pro mobilní zařízení se s notifikacemi běžně setkáváme i u jiných platforem. Notifikace je možné realizovat různými způsoby v podstatě všude tam, kde svým způsobem probíhá nějaká vzájemná komunikace mezi serverem a klienty a nebo kde probíhá monitoring určitých kritických proměnných či vlastností systému.

Veškerá upozornění či oznámení v reálném světě se svým způsobem také dají nazývat notifikacemi, i když u nich zrovna není nutnost využívat internetového připojení se serverem dané "aplikace" nebo cloud MNS serverem. Mezi takové stavy lze řadit třeba monitorování tlaku v pneumatikách aut, semaforey na křižovatkách, čísla pater ve výtahu atd.

Při pohledu z IT hlediska se v poslední době například uplatňují web push notifikace, které fungují na stejném principu jako notifikace pro aplikace na mobilních zařízeních, avšak s tím rozdílem, že jsou implementovány pro konkrétní webové prohlížeče.



Desktop Opt-in



Mobile Website Opt-in

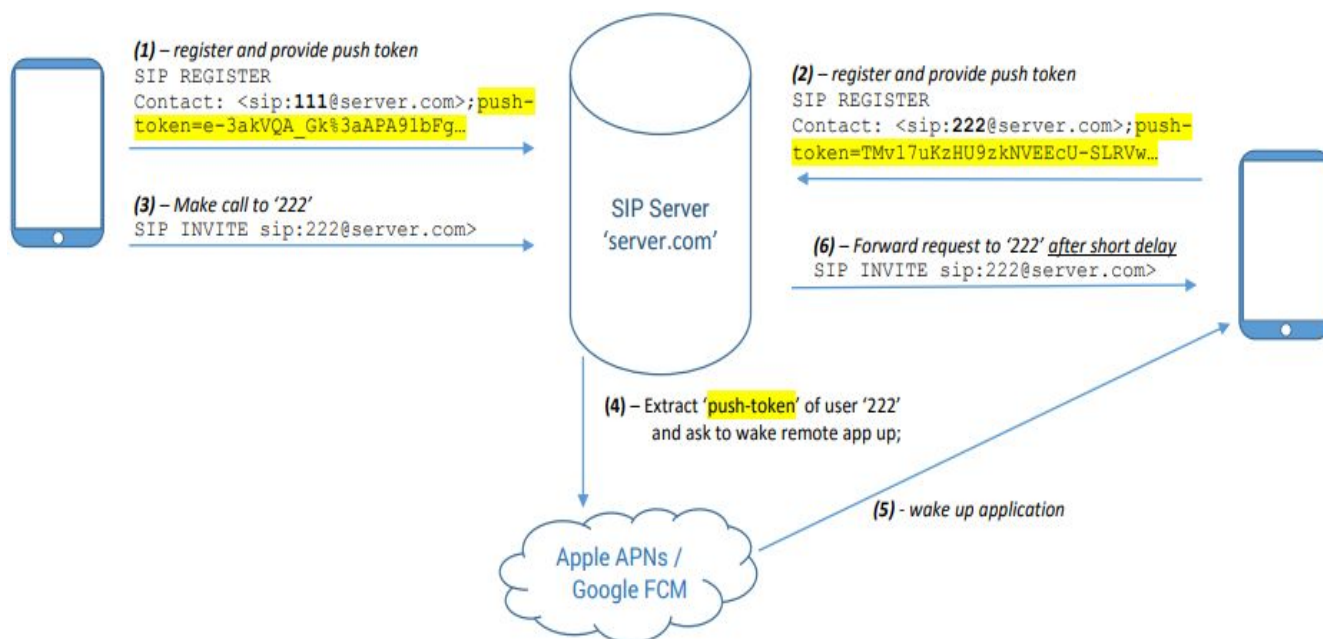
Dobrým příkladem pak dále může být taková VoIP telefonie, kde uživatel může být upozorněn na příchozí hovor právě díky použití push notifikace.

Implementace příchozích hovorů je ve VoIP aplikacích následovná:

Aplikace neustále běží na pozadí, posílá SIP REGISTER/keep-alive pakety a je připravena přijímat SIP INVITE zprávy ze vzdálených zařízení. Tento mód využívá nemalé množství baterie zařízení a spotřebovává také nezanedbatelnou část výpočetního výkonu.

Push oznámení umožňují tyto nežádoucí vlastnosti eliminovat přesunutím funkce neustálého poslechu z aplikace na stranu serveru. Aplikace se spustí, zaregistruje se na serveru a na telefonu ji lze vypnout. Když přijde hovor, VoIP server odešle oznámení do zařízení uživatele. Ten pak má možnost aplikaci aktivovat přijmutím hovoru nebo odmítnutím.

Obrázek níže demonstruje sekvenci zpráv mezi klienty a servery:

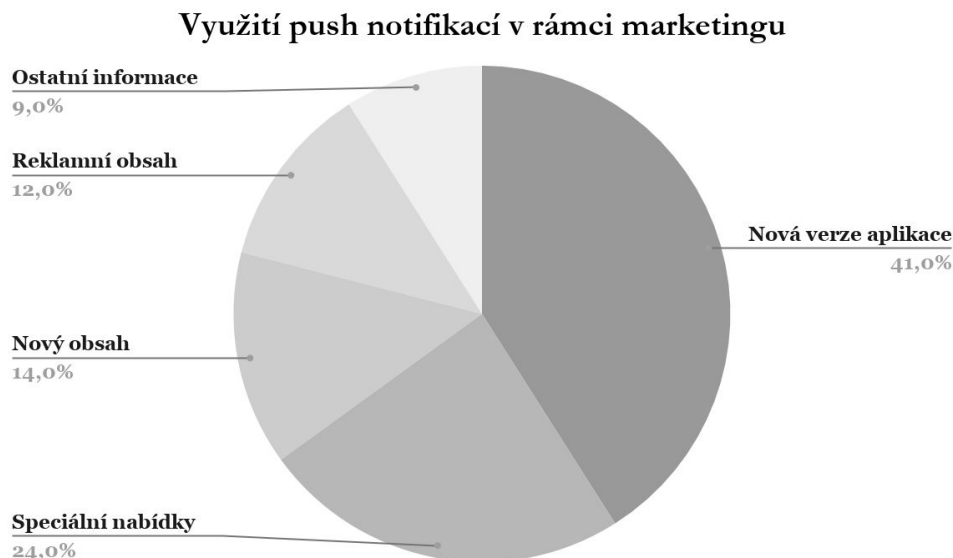


Implementace požaduje provedení změn na straně klienta i serveru, kde uživatel musí poskytnout push token svého zařízení v požadavku SIP REGISTER jako parametr hlavičky "Contact". Před přeposláním požadavku SIP INVITE volanému uživateli SIP server využije token volaného a předá jej MNS serveru s požadavkem o probuzení vzdálené VoIP aplikace.

2.7. Porovnání mobilních notificačních zprostředkovatelů

Efektivní vývoj notificačních technologií zajišťuje mimo jiné své značce správnou reklamu a potenciál stát se mocným marketingovým nástrojem.

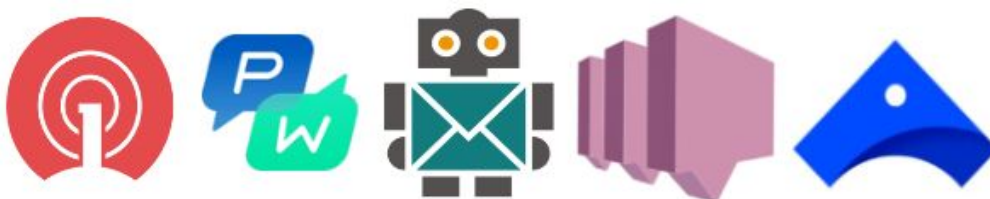
Z grafu níže lze vypožorovat, na co většinou push notifikace v rámci marketingu uživatele upozorňují.



Koncepce push notifikací pronikla do oblasti marketingu velice rychle. Mezi efektivitou a nadměrnou dotěrností notifikací zde však existuje tenká hranice. V první řadě by měla notifikace poskytnout uživateli cenné informace a usilovat o spolupráci. Proto je velmi důležité i vhodné načasování doručení tohoto oznámení společně s poskytnutím relevantního obsahu. Dostat upozornění od rozvážkové služby ve 3 hodiny ráno o tom, že zítra večer budou slevy na určitý druh jídla asi není ten úplně správný a zejména logický přístup. Někteří prodejci například využívají technologii GPS k odesílání notifikací, kdy uživatel je poblíž prodejního místa.

2.7.1. Platformy třetích stran

V roce 2020 bychom nicméně měli očekávat, že na trhu je kromě již zmíněného FCM mnoho dalších vydavatelů. Někteří se totiž zaměřují na menší společnosti, zatímco jiní na globální značky, pokud bychom hovořili o notificačních službách a nástrojích vydavatelů k propagaci určité služby či produktu. Následuje srovnání několika populárních platforem z hlediska toho, co nabízejí a proč se vyplatí je využívat.



OneSignal

Jedná se o multiplatformní službu, umožňující tvůrcům obsahu poskytovat personalizované notifikace v mobilních aplikacích. Tato služba zjednodušuje celkový proces zasílání push notifikací a umožňuje vytvářet vlastní šablony a automatické notifikace v závislosti na použitém systému. V roce 2018 využívalo tuto službu více jak 100 000 mobilních aplikací napříč platformami Android a iOS.

Jedná se o bezplatnou službu. Existují zde však funkce prémiové funkce, jejichž cena se liší v závislosti na požadavcích uživatele.

PushWoosh

Tato platforma umožňuje posílat kromě textu i videa, obrázky a další obsah. Obsahuje také užitečné funkce jako třeba privátní cloud úložiště, který poskytuje vysokou rychlost odesílání upozornění, segmentaci marketingových kampaní atd.

Prvních 14 dní užívání služby je zdarma. Dále je však třeba platit (50-500\$ za měsíc) v závislosti na tom, kolik zařízení chce uživatel pro službu využít. Díky vysoké funkčnosti podporuje PushWoosh libovolný jazyk pro nativní komunikaci s uživatelem, automatické odesílání notifikací nebo provádění A/B testů.

PushBots

Tato služba umožňuje sledovat pokročilé statistiky o notifikacích, což v podstatě znamená, že lze vidět, jak lidé na notifikace reagují. Opět i zde je první 2 týdny free verze a dále pak 29-45\$ v závislosti na vybraném balíčku. PushBots nabízí velmi jednoduchou instalaci a integraci. Statistiky dále obsahují přehledné grafy a analýzy, které pomáhají vývojářům vidět úroveň úspěšnosti poskytovaných notifikačních služeb.

Amazon SNS

Koordinace doručování notifikací předplaceným uživatelům je jedna z vlastností této platformy. Díky jednoduchému API mohou vývojáři posílat notifikace napříč platformami. Zákazníci, využívající i Amazon Web Services mohou posílat až 1 milion notifikací měsíčně zdarma. Po překročení limitu se platí 0,5\$ za každý další odeslaný a 0,5\$ za každý další přijatý milion oznámení.

Mezi vlastnosti této platformy patří ve zkratce jednoduchá škálovatelnost, přizpůsobitelnost a navigace a možnost odesílat oznámení přímo z cloudu při snadné navigaci.

Urban Airship

Velké oblibě se těší i tato platforma. Zahrnuje vícekanálové zasílání zpráv, prediktivní analýzu a sjednocené profily zákazníků. Co se týče ceny, úvodní balíček je zdarma a dále se cena pohybuje mezi 100-350\$/měsíc. Real-time podpora, in-app notifikace a jednoduchá implementace jsou další z důvodů, proč používat právě tuto platformu.

Výše zmíněné notifikační platformy jsou jedny ze široce využívaných služeb v rámci zasílání notifikací a upozorňování uživatelů na důležité informace. Některé z nich představují pro jednotlivce díky svým vlastnostem a ceně spíše alternativní řešení a jsou vhodnějším řešením pro firemní organizace.

Pokud tedy uživatel nemá žádné speciální požadavky a chce integrovat notifikace do své aplikace, vystačí si v případě OS iOS s APNS, o kterém již pojednává kapitola 2.5, respektive s Firebase Cloud Messaging, které podporuje jak OS iOS tak OS Android (viz. kapitola 2.3).

3. Praktická část

Dle zadání se v této části projektu zaměřuji na otestování notifikací OS Android. Pro testování jsem využil poskytnuté mobilní zařízení OneNote 7T (Android 10) a také školní virtuální server, dostupný přes SSH, přes který probíhá generování a odesílání notifikací službě FCM.

Co se týče různých typů notifikací, nejprve se zabývám klasickými aplikačními notifikacemi, u kterých ukazují možnosti vlastních úprav. Následuje testování VoIP push notifikací, jejichž doba doručení je oproti klasickým notifikacím okamžitá. Co se týče klasických notifikací, uživatel musí nejprve notifikaci obsloužit, než může přidružená aplikace provádět akce. Oproti tomu VoIP push notifikace aplikaci otevře sama, tedy bez procesu obsloužení uživatele.

Jak jsem se již zmínil v teoretické části, architektura zasílání notifikací pro OS Android je zřejmá z obrázku níže. Pro realizaci této architektury jsem využil notifikační službu Firebase Cloud Messaging.

3.1. Princip

Celá architektura se v podstatě skládá ze čtyř částí:



Provider: Virtuální server (2001:718:1001:2c6::112)

- slouží k sestavování parametrů a obsahu notifikací
- posílá tyto data společně s uživatelským tokenem FCM serveru
- vůči FCM serveru se autorizuje pomocí serverového API tokenu

Cloud: FCM server

- obsahuje serverový API token, který provider potřebuje znát
- registruje se zde uživatelský token
- odesílá notifikace konkrétnímu zařízení s názvem balíčku aplikace

Zařízení: OneNote 7T

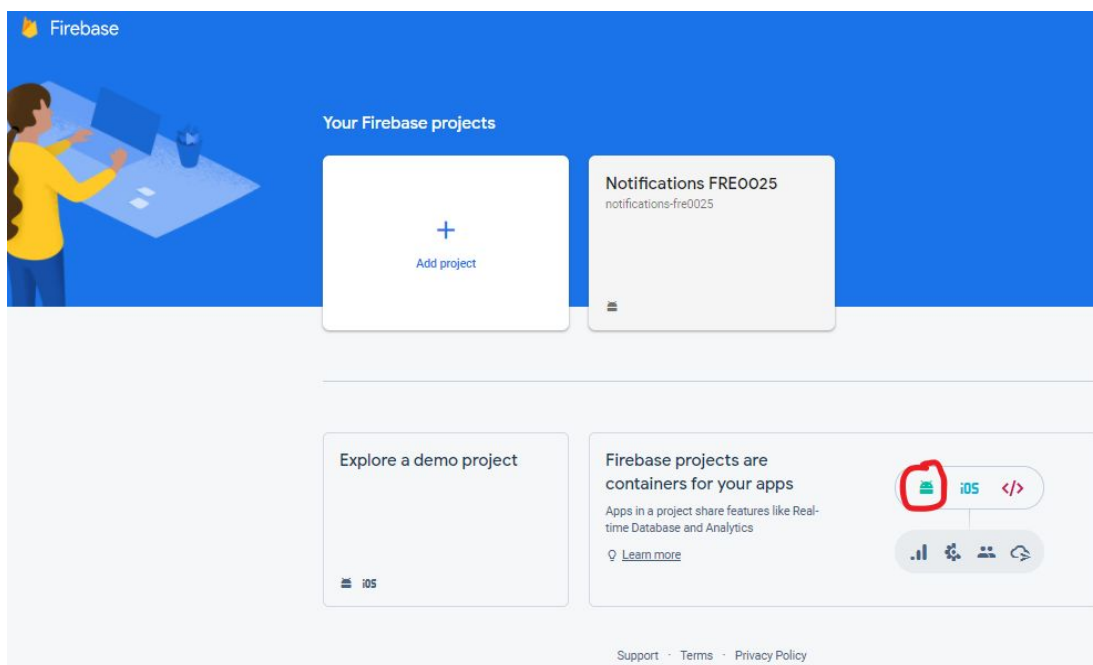
- vyhledá balíček přidružené aplikace a otevře ji po kliknutí na notifikaci

Klientská aplikace, vyvinutá v Android Studio

- obsahuje vygenerovaný uživatelský token, kterým se registruje na FCM
- vytváří prostředí pro přijetí a zobrazení příchozích notifikací
- může zobrazit další data, přenesená v notifikacích

3.2. Integrace služby Firebase Cloud Messaging

Pro začátek je potřebné založit si přes <https://console.firebase.google.com/u/0/> projekt (je nutné být přihlášený přes svůj Google účet) a v něm si registrovat svou aplikaci uvedením aplikačního balíčku.



✕ Add Firebase to your Android app

1 Register app

Android package name ⓘ

App nickname (optional) ⓘ


Dalším krokem je stáhnutí konfiguračního souboru, obsahujícího mj. informace o našem balíčku a jeho vložení do adresářové struktury projektu podle návodu.

2 Download config file Instructions for Android Studio below | [Unity](#) [C++](#)

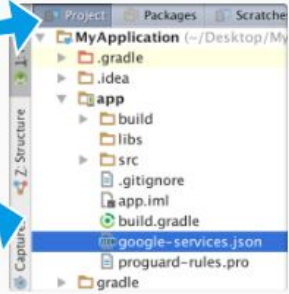
[Download google-services.json](#)

Switch to the Project view in Android Studio to see your project root directory.

Move the google-services.json file you just downloaded into your Android app module root directory.



google-services.json



Previous [Next](#)

Pro integraci služby FCM do aplikace “Notifications”, vyvíjené v Android Studiu, je nyní nutné zajistit podporu této služby. Tato integrace zahrnuje přidání konkrétních pluginů a závislostí do souborů sestavovacího nástroje Gradle.

3

Add Firebase SDK

Instructions for Gradle | [Unity](#) | [C++](#)

The Google services plugin for [Gradle](#) loads the `google-services.json` file you just downloaded. Modify your build.gradle files to use the plugin.

Project-level build.gradle (<project>/build.gradle):

```
buildscript {
    repositories {
        // Check that you have the following line (if not, add it):
        google() // Google's Maven repository
    }
    dependencies {
        ...
        // Add this line
        classpath 'com.google.gms:google-services:4.3.4'
    }
}

allprojects {
    ...
    repositories {
        // Check that you have the following line (if not, add it):
        google() // Google's Maven repository
    }
}
```

Java Kotlin

App-level build.gradle (<project>/<app-module>/build.gradle):

```
apply plugin: 'com.android.application'
// Add this line
apply plugin: 'com.google.gms.google-services'

dependencies {
    // Import the Firebase BoM
    implementation platform('com.google.firebase:firebase-bom:26.0.0')

    // Add the dependency for the Firebase SDK for Google Analytics
    // When using the BoM, don't specify versions in Firebase dependencies
    implementation 'com.google.firebase:firebase-analytics'

    // Add the dependencies for any other desired Firebase products
    // https://firebase.google.com/docs/android/setup#available-libraries
}
```

By using the Firebase Android BoM, your app will always use compatible Firebase library versions. [Learn more](#)

Finally, press "Sync now" in the bar that appears in the IDE:

Gradle files have changed since last sync

Sync now

Previous Next

Pro úspěšné přidání podpory FCM do aplikace je pak nutné aplikaci s integrovanými závislostmi výše pouze spustit. Pokud se na stránce objeví oznámení o úspěšném ověření aplikace ze strany FCM, je možné přejít k dalšímu kroku.

✓

Register app

Android package name: com.fre0025.notifications, App nickname: FRE0025_Notifications

✓

Download config file

✓

Add Firebase SDK

4

Run your app to verify installation

✓

Congratulations, you've successfully added Firebase to your app!

Previous

Continue to console

3.3. Generování uživatelského tokenu

Uživatelský token je unikátní identifikátor, který v sobě definuje:

- ID zařízení, které dostane notifikaci
- Aplikace v zařízení, která dostane notifikaci

Tento token lze získat procesem, běžícím na pozadí aplikace, která tím pádem musí být napřed alespoň jednou spuštěna. Token se může také změnit např. při přeinstalování aplikace nebo smazání dat aplikace. Následující část kódu demonstruje získání uživatelského tokenu:

```
FirebaseInstanceId.getInstance().getInstanceId()
    .addOnCompleteListener(new OnCompleteListener<InstanceIdResult>() {
        @SuppressWarnings("SetTextI18n")
        @Override
        public void onComplete(@NonNull Task<InstanceIdResult> task) {
            if (task.isSuccessful()) {
                String token = Objects.requireNonNull(task.getResult()).getToken();
                Log.d(TAG, "onComplete: Token: " + token);
                tv_token.setText("FCM client token generated:");
                tokenText.setText(token);
            } else {
                tv_token.setText("Token generation failed\n");
            }
        }
    });
```

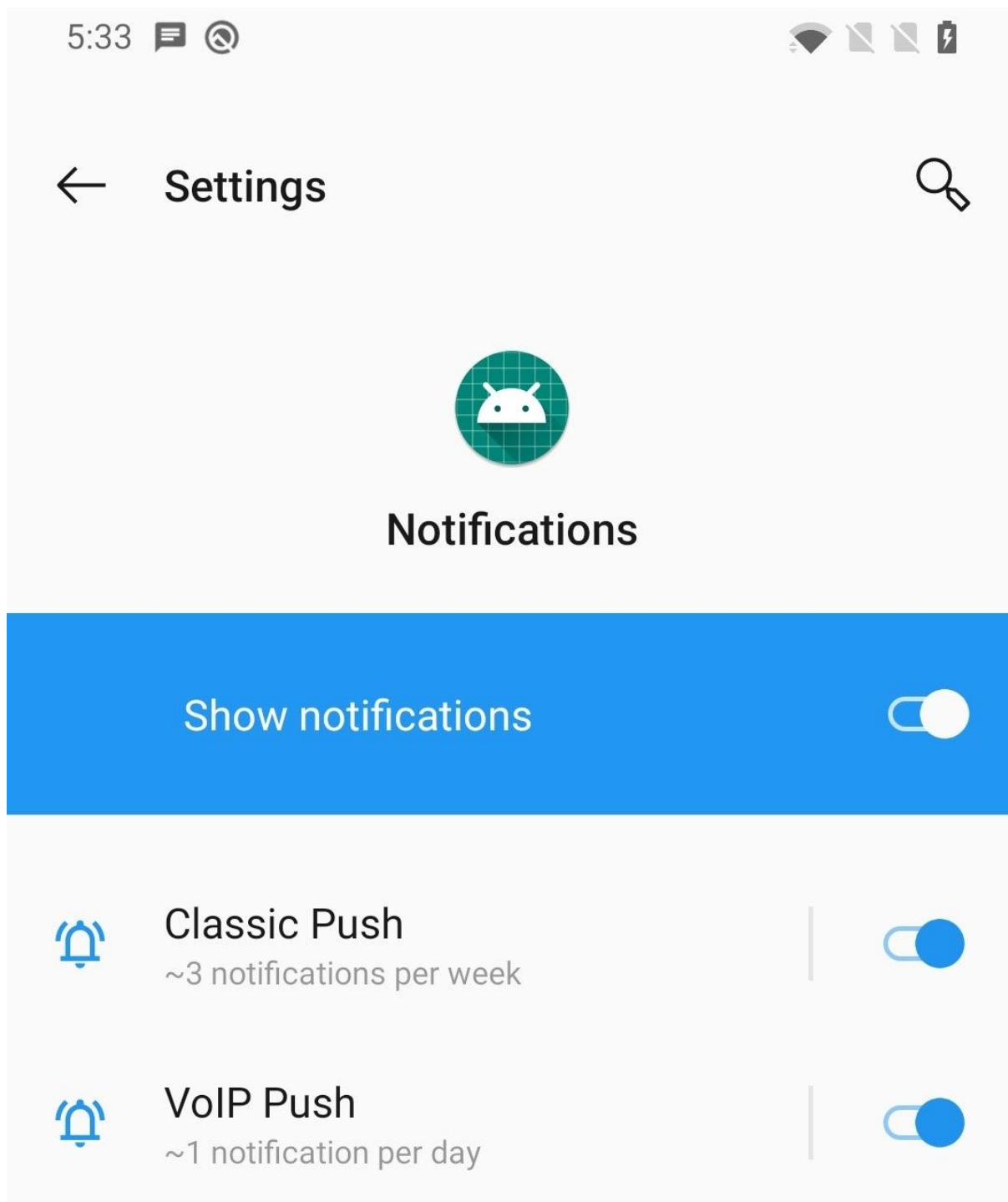
Na první pohled je zřejmé použití deprecated funkce, nicméně dostupná literatura uvádí, že i tahle možnost je stále správná. Cílem této funkce je získat instanci aplikace, která je spojená s backendem FCM a přidání listeneru abychom věděli, kdy se úloha dokončí. Po získání tokenu se uloží do proměnné a také se vypíše na obrazovku a do logu. Pokud tento token nyní poskytneme FCM cloudu, který umožňuje posílat notifikace, bude schopen komunikovat i s naším zařízením, respektive aplikací.

3.4. Tvorba notifikačního kanálu

Od Android verze 8.0 je pro zobrazení notifikací nutné jejich přiřazení do notifikačních kanálů. Pro vyvíjenou aplikaci je proto tedy nutné vytvořit vlastní notifikační kanál, kterému lze nastavit chování již přímo při naprogramování (ne pouze v Nastavení zařízení). Notifikační kanál vytvoříme následovně:

```
private void createNotificationChannelForClassic() {
    Uri soundUri = Uri.parse(
        "android.resource://" + getApplicationContext().getPackageName() + "/" + R.raw.insight);
    AudioAttributes audioAttributes = new AudioAttributes.Builder()
        .setContentType(AudioAttributes.CONTENT_TYPE_SONIFICATION)
        .setUsage(AudioAttributes.USAGE_ALARM)
        .build();
    NotificationChannel channel = new NotificationChannel
        (FCM_CHANNEL_ID1, name: "Classic Push", NotificationManager.IMPORTANCE_HIGH);
    channel.setDescription("Classic notification channel");
    channel.enableVibration(true);
    channel.setSound(soundUri, audioAttributes);
    channel.setLockscreenVisibility(Notification.VISIBILITY_PUBLIC);
    NotificationManager manager = getSystemService(NotificationManager.class);
    if (manager != null)
        manager.createNotificationChannel(channel);
}
```


Z obrázku lze vidět, že do konstruktoru třídy NotificationChannel se vkládá ID kanálu, jeho jméno a také důležitost. V tomto případě jsem ji nastavil na HIGH (viz. kapitola 2.2.3). Dále jde kanálu nastavit mimo jiné notificační dioda zařízení, vibrace či zobrazení notifikace při zamčeném zařízení. Tento kanál slouží pro klasické notifikace. Pro notifikace VoIP jsem vytvořil odlišný notificační panel, který se liší především použitím jiného vyzváněcího tónu. V nastavení zařízení jsou pak pro danou aplikaci vytvořené notificační kanály vidět jako aktivní.



3.5. Přijímání notifikací

Notifikace se musejí přijímat i když zařízení momentálně nepoužíváme nebo když aplikaci nemáme otevřenou. Tím pádem je potřeba založit ne aktivitu, ale službu, která tak bude běžet na pozadí a v případě příchozí komunikace ji zařízení vyvolá. Tato služba se musí (obdobně jako každá aktivita) uvést do souboru AndroidManifest.xml.

```
<service android:name=".FCMMessageReceiverService"
    android:exported="false">
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT" />
        <action android:name="com.google.firebase.INSTANCE_ID_EVENT" />
    </intent-filter>
</service>
```

V této službě je nutné implementovat funkci `onMessageReceived`, která přijímá příchozí zprávy:

```
@Override
public void onMessageReceived(@NonNull RemoteMessage remoteMessage) {
    super.onMessageReceived(remoteMessage);
    Log.d(TAG, msg: "onMessageReceived: called");
    Log.d(TAG, msg: "onMessageReceived: MessageReceived from: " + remoteMessage.getFrom());

    //GET DATA FROM REQUEST
    Map<String, String> data = remoteMessage.getData();
    String notificationType = data.get("type");
    title = data.get("title");
    text = data.get("body");
    String image = data.get("image");
    Bitmap bitmap = getBitmapFromURL(image);
    NotificationCompat.BigPictureStyle bigPicStyle = new NotificationCompat.BigPictureStyle().bigPicture(bitmap);
    String color = data.get("color");
    soundUri1 = Uri.parse("android.resource://" + getPackageName() + "/" + R.raw.insight);
    soundUri2 = Uri.parse("android.resource://" + getPackageName() + "/" + R.raw.call);

    Intent myIntent = new Intent(getApplicationContext(), MainActivity.class);
    PendingIntent pendingIntentClassic = PendingIntent.getActivity(getApplicationContext(), requestCode: 0, myIntent, PendingIntent.FLAG_ONE_SHOT);
    //BUILD NOTIFICATIONS
    if(notificationType.equals("classic")) {
        NotificationCompat.Builder notificationBuilder =
            new NotificationCompat.Builder( context: this, FCM_CHANNEL_ID1)
                .setSmallIcon(R.drawable.ic_chat)
                .setLargeIcon(BitmapFactory.decodeResource(getResources(), R.drawable.ic_animal))
                .setContentTitle(title)
                .setContentText(text)
                .setAutoCancel(true)
                .setVibrate(new long[]{0, 100, 200, 300})
                .setSound(soundUri1)
                .setColor(Color.parseColor(color))
                .setStyle(bigPicStyle)
                .setContentIntent(pendingIntentClassic);

        NotificationManager notificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
        Notification notification = notificationBuilder.build();
        notificationManager.notify(CLASSIC_NOTIFICATION_ID, notification);
    } else if(notificationType.equals("voip")) {
        Intent intent = new Intent(FULL_SCREEN_ACTION, uri: null, packageContext: this, NotificationReceiver.class);
        PendingIntent pendingIntentVoip = PendingIntent.getBroadcast( context: this, requestCode: 0, intent, PendingIntent.FLAG_UPDATE_CURRENT);
        AlarmManager alarmManager = (AlarmManager) this.getSystemService(Context.ALARM_SERVICE);
        if (alarmManager != null)
            alarmManager.set(AlarmManager.RTC_WAKEUP, System.currentTimeMillis(), pendingIntentVoip);
        NotificationManagerCompat.from(this).cancel(VOIP_NOTIFICATION_ID); //cancel last notification for repeated tests
    }
}
```

Z přichozích dat zde získám položky “title”, “body”, “image”, “color” a “sound”, které se pak použijí přímo pro sestavení notifikace, kterou již uživatel uvidí na svém zařízení. Společně s těmito parametry existuje velké množství dalších parametrů.

Po získání parametrů pak vytvářím Intent, který po kliknutí na notifikaci odkáže uživatele na úvodní aktivitu aplikace. Následně probíhá sestavení těla notifikace, ke kterému se postupně přidávají jednotlivé parametry. Důležité je přidělit notifikaci danému notifikačnímu kanálu (v tomto případě klasickým notifikacím). Po sestavení “těla” notifikace se pak vytvoří notifikační manažer, který notifikaci pod určitým ID publikuje systému, který ji vyvolá.

Ve spodní části útržku kódu se pak řeší VoIP PUSH notifikace, kde se vytváří Intent, který odkazuje uživatele do třídy NotificationReceiver. Ta řeší vyvolání tzv. full screen notifikace. Nakonec je zde vytvořen AlarmManager, který má možnost vyvolat aplikaci z režimu zamknutého displeje.

3.6. Přijetí obsahu, které mohou notifikace obsahovat

Kromě samotného obsahu, který nám notifikace zobrazuje, může obsahovat i další data, která nicméně nejsou uživateli zobrazena. Následující kód je jedním z možných řešení, jak tyto data získat.

```
//ADDITIONAL LOGS
if(remoteMessage.getData().size() > 0) {
    Log.d(TAG, msg: "onMessageReceived: Data Size: " + remoteMessage.getData().size());
    for (String key : remoteMessage.getData().keySet()) {
        Log.d(TAG, msg: "onMessageReceived Key: " + key + " Data: " + remoteMessage.getData().get(key));
    }
    Log.d(TAG, msg: "onMessageReceived: Data: " + remoteMessage.getData().toString());
}
```

Tato část kódu je také v metodě onReceivedMessage. Tímto způsobem je možné získat veškerá data (včetně dříve zmíněných parametrů, tvořících tělo notifikace), která obsahoval HTTP požadavek pro vytvoření notifikace. Data jsou ve formátu JSON - tedy hodnota : klíč. Nastavení těchto hodnot na straně odesílatele požadavku je popsáno v kapitole 3.9.

3.7. Nastavení akčních tlačítek

Notifikace mohou obsahovat také akční tlačítka, která obsahují funkce, jak s danou notifikací naložit nebo obsloužit ji. Z testovacího hlediska jsem v tomto projektu přidal k notifikaci tlačítko “Dismiss”, které notifikaci zavře. Pro tento proces je potřebné tělu notifikace přidat položku

```
.addAction(R.drawable.ic_delete, "Dismiss", dismissIntent)
```

která potřebuje jako parametr PendingIntent. Ten musí být deklarován takto:

```
Intent showIntent = new Intent(this, MainActivity.class);
```

```
showIntent.putExtra(getPackageName(), CLASSIC_NOTIFICATION_ID);
```

```
PendingIntent dismissIntent =
```

```
PendingIntent.getActivity(this, 0, showIntent, PendingIntent.FLAG_UPDATE_CURRENT);
```

Odkazuje nás přitom k otevření hlavní aktivity, do které je nyní třeba přidat ošetření do funkce onCreate:

```
NotificationManager manager =
```

```
(NotificationManager) getSystemService(NOTIFICATION_SERVICE);
```

```
if(getIntent() != null && getIntent().hasExtra(getPackageName())){
```

```
    manager.cancel(CLASSIC_NOTIFICATION_ID);
```

```
    finish();
```

```
}
```

Pokud uživatel tedy klikne na akční tlačítko s daným nastavením a uvedením kontrolní hodnoty (putExtra), v aktivitě se pak zjišťuje, zda Intent obsahuje tuto konkrétní hodnotu. Pokud ano, provede se část kódu, ve které se zruší notifikace s daným ID (pokud existuje).

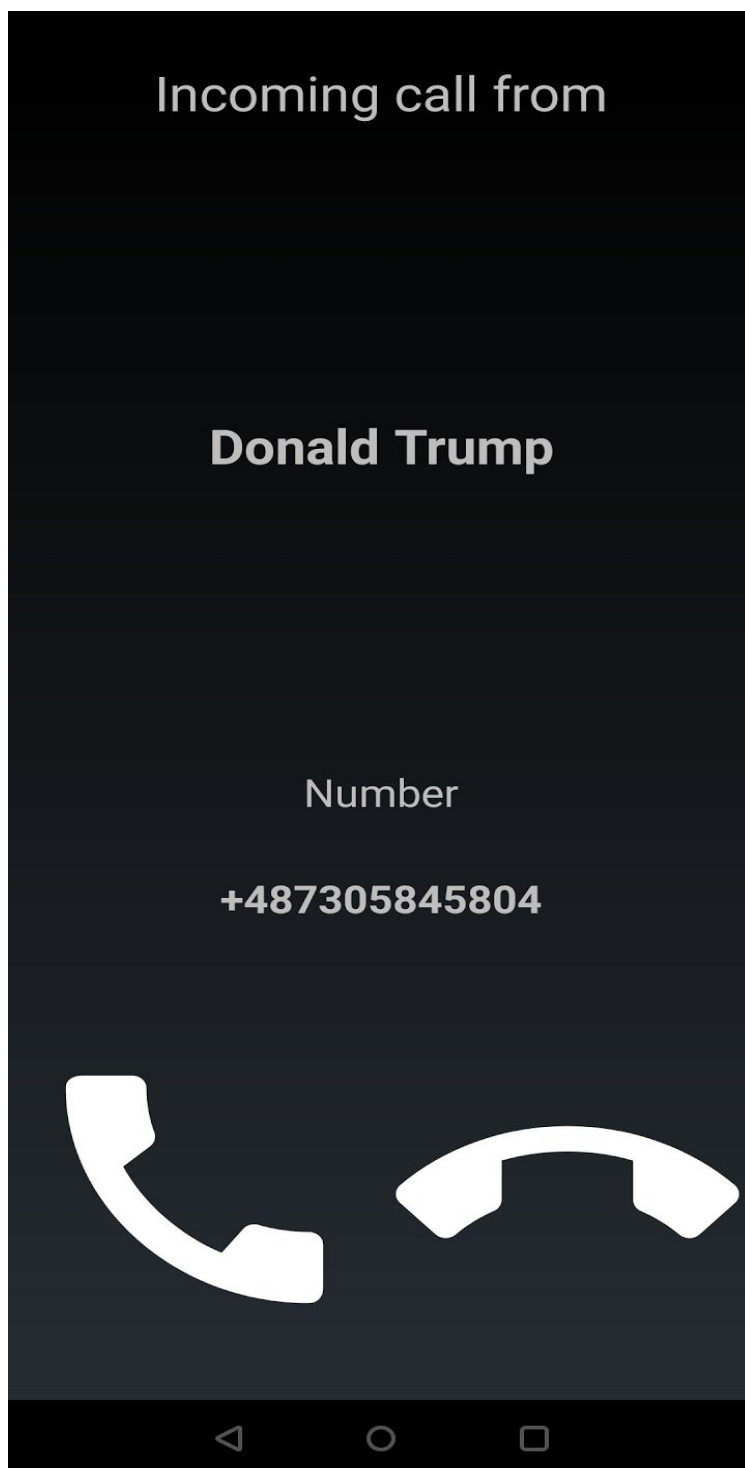
3.8. Nastavení zobrazení klasických push notifikací vs VoIP push

Pro představu je dobré zobrazit předchozí implementaci příchozích notifikací také v grafické reprezentaci.

Klasické notifikace mají standardní rozložení a zobrazí se ve formě upozornění na notifikační liště a je jedno, co uživatel zrovna se zařízením dělá nebo zda je aplikace v režimu foreground, background nebo není zapnutá vůbec. Tohoto stavu je docíleno stranou vyvolání HTTP požadavku na straně serveru, kde je doporučeno vkládat veškeré parametry notifikace pod JSON klíč "data" (nikoliv "notification", kdy notifikace se zobrazovala pouze ve foreground režimu).



Jelikož VoIP notifikaci chceme zobrazit ve formě příchozího hovoru ve formě aplikace přes celou obrazovku, je nutné vytvořit aktivitu IncomingCall, která obsahuje rozložení obrazovky pro obsluhu tohoto typu notifikace. Tato obrazovka by se měla uživateli zobrazit i při zamčené obrazovce. Pokud uživatel právě provádí se zařízením interakci, zobrazí se nejprve heads-up notifikace, kde jejím rozkliknutím se zobrazí tato obrazovka.



3.9. Nastavení defaultních parametrů pro notifikace

V rámci testování jsem se setkal také s problémem, kdy aplikace ignorovala nastavený vlastní notifikační panel a používala místo něj panel “Miscellaneous”, tedy defaultní kanál. To samé platilo v případě neuvedení ikony notifikace nebo její barvy. Řešením bylo přidání následujících řádků do souboru “AndroidManifest.xml”, které definují vlastní parametry zobrazování notifikací pro danou aplikaci:

```
<meta-data
    android:name="com.google.firebase.messaging.default_notification_icon"
    android:resource="@drawable/ic_sync" />
<meta-data
    android:name="com.google.firebase.messaging.default_notification_color"
    android:resource="@color/colorFcmNotification" />
<meta-data
    android:name="com.google.firebase.messaging.default_notification_channel_id"
    android:value="@string/fcm_channel"/>
```

V tomto stavu je klientská aplikace již schopna využívat nastavení vlastního notifikačního kanálu a přijímat notifikace ze strany serveru, pokud server zná uživatelský token.

3.10. Řešení posílání notifikací

Když již máme stranu pro přijímání notifikací připravenou, je nyní na místě mít i stranu, která notifikace odešle. Nejprve jsem si myslel, že bude stačit FCM konzole, která je součástí služby FCM cloud, nicméně tvoření notifikací je tam příliš neefektivní a navíc značně omezené vzhledem k tomu, co FCM HTTP Legacy protokol nabízí.

Místo FCM serveru jsem se rozhodl nejprve využít aplikaci Postman, která umožňuje generování HTTP požadavků na vzdálenou URL s přidáním hlaviček a vlastního těla požadavku. Tento přístup byl mnohem komplexnější a rychlejší než vytváření notifikací přes FCM konzoli, nicméně i zde přišlo omezení v podobě možnosti využívat Postman pouze na OS Windows a také nemožnost zahrnout SSH tunel.

Nakonec jsem využil školní virtual server, do kterého se přihlašuji pomocí VPN a SSH tunelu. Na tomto serveru jsem si nainstaloval službu cURL, která provádí mj. to samé co Postman. Největší výhodou této implementace je možnost uložit si celý HTTP požadavek do skriptu, kde mám možnost upravovat jednotlivá JSON pole notifikace pomocí argumentů a z toho plynoucí rychlost odesílacího procesu, kdy stačí napsat jeden jednoduchý příkaz. Dalším kritériem výběru cURL bylo zakomponování virtuálu s SSH, vychází ze zadání projektu.


```
root@server: /home/student
GNU nano 2.9.3 notifications.sh

#!/bin/bash
DEVICE_TOKEN="f1syIP4cSKO9Mh123X51iC:APA91bH8QAz3T5vWfuk2-PJtR6GPtSqprDrUOe77P2b_xpFNU4kG9S1_jvt2wNX39wIf5Lcs1Y7JYcPNJcXNFP9270r1RjG4b7ugaoSGU-D8HNHOe-ZBvVIs"
TYPE=$1
TIME=$(date +%d-%m-%Y_%H:%M:%S)
TITLE=$2
BODY=$3

if [ $4 = "vsb" ]; then
    IMG="https://i.ytimg.com/vi/suCm2edqq0Y/maxresdefault.jpg"
elif [ $4 = "img" ]; then
    IMG="https://dirtybourbonrivershow.files.wordpress.com/2015/11/sunny-beach-wallpaper-hd.jpg?w=680"
fi

if [ $TYPE = "classic" ]; then
    SOUND="insight.mp3"
elif [ $TYPE = "voip" ]; then
    SOUND="call.mp3"
fi

curl --location --request POST 'https://fcm.googleapis.com/fcm/send' \
--header 'Authorization: key=AAAAATX16FNw:APA91bFN9bIwVLiH2oWv7COeqfBarvLMq5m9EWdXz2iVfO6pK9HY-DBktCRjKq35ABD3-zpPcjvXcvWNMzR5UVBxMnDf71FL6-h9MJFv2MJCF_tS07E$' \
--header 'Content-Type: application/json' \
--data-raw '{
  "to": "'"$DEVICE_TOKEN"'",
  "data": {
    "type": "'"$TYPE"'",
    "mydata1": "'"$TIME"'",
    "mydata2": "Another custom data..",
    "title": "'"$TITLE"'",
    "body": "'"$BODY"'",
    "image": "'"$IMG"'",
    "sound": "'"$SOUND"'",
    "color": "#FF0000"
  }
}'
printf "\n"
```

Začátek skriptu obsahuje proměnné token zařízení, typ, název, text a obrázek notifikace, které musí uživatel zapsat jako argumenty skriptu. Další proměnná je zapsání času jako položka dalšího obsahu notifikace (viz kapitola 3.6). Pro demonstraci následuje jednoduchý výběr obrázku, který se pro notifikaci použije. Zde je důležité uvést, že se však nejedná o ikonku notifikace na notifikační liště, ale o bitmap obrázek, který je zobrazen přímo v notifikaci. Dále následuje výběr vyzvánění, kdy pro klasickou notifikaci zazní krátký zvuk “insight.mp3” a pro příchozí hovor vyzvánění “call.mp3”. Tyto dva zvukové soubory jsou ve podadresáři “raw” samotné klientské aplikace.

Následuje již samotný příkaz “curl” s odpovídajícími přepínači, které jsou povinné. Je jasné, že se jedná o HTTP požadavek typu POST, který je nutné odeslat na adresu fcm.googleapis.com/fcm/send, která se uvádí v oficiální dokumentaci pro FCM Legacy HTTP protokol. Dále je třeba uvést hlavičku “Authorization”, která obsahuje token API serveru - tedy serveru, který tuto notifikaci přepošle do našeho mobilního zařízení. Další je pak hlavička, informující, že obsah dat je typu JSON.

Další položkou už jsou samotná JSON data, která obsahují uživatelský token a samotný obsah notifikace. Právě zde je místo, kde se dají upravovat parametry odeslané notifikace. Druhou věcí je strana klienta, která musí mít chování těchto parametrů implementovaná také, jinak se zobrazí defaultní parametry pro konkrétní notifikační kanály.

Demonstrace

<https://drive.google.com/drive/folders/1gtlWrQmPTCtEYBsfXiLV9wbpzPf-2x7c?usp=sharing>

4. Závěr

Předmětem tohoto projektu bylo vyzkoušet si, na jakém principu fungují notifikace na mobilních zařízeních nejprve z teoretického hlediska a následně i praktického.

V teoretické části jsem na úvod objasnil, co to vůbec notifikace jsou a kde se s nimi můžeme setkat. Popsal jsem princip fungování notifikací včetně uvedení jednotlivých částí sítě, kterými musí notifikační data projít společně s uvedením základního rozdělení notifikací na tzv. “push” a “in-app”. Následoval popis jednotlivých významných rysů a problematiky notifikací pro operační systémy Android a iOS včetně popisu jejich notifikačních architektur. Dále jsem se zmínil o dalších možnostech zobrazování notifikací, kde jsem uvedl VoIP PUSH nebo WebPush notifikace. Před zakončením teoretické části projektu jsem srovnal několik nejznámějších komerčních notifikačních platforem z hlediska toho, co nabízí běžnému uživateli v porovnání s volně dostupnými prostředky, určenými k vývoji pro individuální projekty.

Praktická část se zaměřuje na implementaci vlastní aplikace, vytvořené v Android Studiu, která převádí teoretické poznatky do praxe. Nejtěžší částí projektu bylo porozumět souvislostem mezi volně dostupnými útržky kódu, roztroušenými po internetu, které většinou vysvětlovaly konkrétní problém, ale už ignorovaly zakomponování do okolního kontextu.

Po shromáždění dostatku informací jsem se rozhodl, že využiji volně dostupnou notifikační platformu Firebase Cloud Messaging, přes kterou nakonec směřovaly všechny mé notifikační požadavky. Po výběru platformy bylo nutné vytvořit klientskou aplikaci, která bude notifikace přijímat. Do této aplikace musely být zakomponovány závislosti a pluginy právě FCM platformy. Prvním “záchytným bodem” v rámci celkového dokončení praktické části bylo přijímat zařízením nejjednodušší notifikace přímo z FCM konzole, která slouží k posílání notifikací na daná zařízení. To se mi povedlo, nicméně pro splnění požadavků zadání projektu to bylo nedostatečné. Využil jsem proto “vlastní server” v podobě virtuálního serveru, přiděleného školou pro výuku, na kterém jsem si vytvořil skript, který FCM platformě posílá data, ze kterých lze vytvořit notifikace již v mnohem více uživatelsky upravitelné formě než v případě online FCM konzole.

Co se týče samotných typů notifikací, pracoval jsem s dvěma základními, a to klasickými push a VoIP PUSH. Pro každý typ jsem vytvořil individuální implementaci, kterou jsem se snažil v praktické části práce popsat.

Nakonec uvádím odkaz na videa, která demonstrují funkční odeslání notifikací serverem přes SSH tunel FCM službě a zobrazení těchto notifikací na zapůjčeném zařízení OneNote 7T.

Reference:

1. <https://medium.com/@Imaginnovation/app-development-decisions-push-notifications-vs-in-app-messaging-5d7ac1e56233>
2. <https://xtremepush.com/the-complete-guide-to-push-notifications-for-app-and-web/>
3. <https://idapgroup.com/blog/push-notifications-mechanism-for-ios-android/>
4. <https://developer.apple.com/design/human-interface-guidelines/ios/system-capabilities/notifications/>
5. <https://yalantis.com/blog/push-notifications-mechanism-outline-ios-and-android/>
6. <https://stormotion.io/blog/top-push-notification-services/>
7. <https://www.cleveroad.com/blog/push-notification-platform-comparison-see-the-main-benefits#localytics>
8. https://voipsip SDK.com/Download/docs/ABTO_Receiving_VoIP_incoming_calls_via_Push_Notifications.pdf
9. https://firebase.google.com/docs/cloud-messaging/android/first-message#java_1
10. <https://www.simplifiedcoding.net/firebase-cloud-messaging-tutorial-android/#What-is-Firebase-Cloud-Messaging>
11. <https://code.tutsplus.com/tutorials/how-to-get-started-with-push-notifications-on-android--cms-25870>
12. <https://stackoverflow.com/questions/42273699/how-to-stack-firebase-cloud-messaging-notifications-when-the-application-is-not/43914028#43914028>
13. <https://developer.android.com/training/notify-user/build-notification#SimpleNotification>
14. <https://developer.android.com/training/basics/firstapp/starting-activity#BuildIntent>
15. <https://www.tutlane.com/tutorial/android/android-notifications-bigtextstyle-bigpicturestyle-inboxstyle>
16. <https://stackoverflow.com/questions/53684967/c-sharp-fcm-the-remote-server-returned-an-error-400-bad-request>
17. <https://medium.com/@apoorv487/testing-fcm-push-notification-through-postman-terminal-part-1-5c2df94e6c8d>