

**VYSOKÁ ŠKOLA POLYTECHNICKÁ JIHLAVA**

Katedra technických studií

Obor Aplikovaná informatika

**Analýza a implementace systému pro  
instant messaging**

bakalářská práce

Autor: Ondřej Schrek

Vedoucí práce: Ing. Marek Musil

Jihlava 2017

Vložený papír se zadáním **Vysoká škola polytechnická Jihlava**

Tolstého 16, 586 01 Jihlava

## **ZADÁNÍ BAKALÁŘSKÉ PRÁCE**

Autor práce: **Jan Novák**  
Studijní program: Elektrotechnika a informatika  
Obor: Aplikovaná informatika  
Název práce: **Název Práce**  
Cíl práce: Práce se.

**Jméno vedoucího BP**  
vedoucí bakalářské práce

**Jméno vedoucího katedry**  
vedoucí katedry  
Katedra elektrotechniky a informatiky

## **Abstrakt**

Předmětem této práce je analýza a implementace systému umožňující komunikaci typu instant messaging. Cílem práce je zhodnocení vlastního způsobu návrhu a implementace, použitých nástrojů a služeb, taktéž srovnání vybraných aplikací, které jsou v současné době dostupné na trhu. Práce také popisuje a představuje možnosti systému Firebase, což je cloudové backend řešení pro mobilní a webové aplikace od společnosti Google.

Praktická část práce se skládá z návrhu a realizace aplikace pro mobilní platformu Android, která slouží pro koncové uživatele aplikace, nakonfigurování služby Firebase a vytvoření webové aplikace, která funguje jako administrační rozhraní celého systému.

## **Klíčová slova**

Java, Android, Firebase, cloud, instant messaging

## **Abstract**

Zde bude anotace práce anglicky.

## **Key words**

Java, Android, Firebase, cloud, instant messaging

Prohlašuji, že předložená bakalářská práce je původní a zpracoval/a jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem v práci neporušil/a autorská práva (ve smyslu zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů, v platném znění, dále též „AZ“).

Souhlasím s umístěním bakalářské práce v knihovně VŠPJ a s jejím užitím k výuce nebo k vlastní vnitřní potřebě VŠPJ.

Byl/a jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje AZ, zejména § 60 (školní dílo).

Beru na vědomí, že VŠPJ má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom/a toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem VŠPJ, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených vysokou školou na vytvoření díla (až do jejich skutečné výše), z výdělku dosaženého v souvislosti s užitím díla či poskytnutí licence.

V Jihlavě dne

.....

Podpis

## **Poděkování**

*Na tomto místě bych rád poděkoval svému vedoucímu práce Ing. Marku Musilovi za konzultace, poskytnutí tématu a možnosti vytvářet ho pod jeho vedením.*

# Obsah

1	Úvod.....	9
2	Instant messaging.....	10
2.1	Existující softwarová řešení .....	10
2.1.1	Facebook Messenger.....	10
2.1.2	WhatsApp .....	11
2.1.3	Skype .....	12
2.1.4	Slack.....	12
3	Technologická řešení .....	14
3.1	Protokol XMPP .....	14
3.2	Návrh vlastní implementace.....	15
4	Služby typu mBaaS.....	18
4.1	Porovnání současných poskytovatelů .....	18
4.1.1	Firebase .....	18
4.1.2	Kinvey.....	19
4.1.3	Backendless .....	19
4.2	Shrnutí.....	20
5	Firebase.....	21
5.1	Analytics .....	21
5.2	Develop .....	21
5.2.1	Cloud Messaging .....	21
5.2.2	Cloud Functions .....	22
5.2.3	Authentication.....	23
5.2.4	Realtime Database .....	23
5.2.5	Storage .....	24
5.2.6	Hosting.....	24

5.2.7	Test Lab .....	24
5.2.8	Crash Reporting .....	25
5.3	Grow.....	25
5.3.1	Notifications.....	25
5.3.2	Remote Config.....	25
5.3.3	App Indexing .....	26
5.3.4	Dynamic Links.....	26
5.3.5	Invites.....	26
6	Požadavky na systém .....	27
6.1	Požadavky na klientskou aplikaci .....	27
6.2	Požadavky na serverovou aplikaci .....	27
7	Návrh mobilní aplikace.....	28
7.1	Schéma aktivit a fragmentů.....	28
7.2	Návrh schématu databáze.....	29
7.2.1	Sekce dat pro aplikaci .....	29
7.2.2	Sekce dat pro systém.....	29
8	Návrh webové administrace.....	30
9	Realizace mobilní aplikace .....	31
9.1	Nastavení projektu .....	31
9.2	Přihlašování.....	32
9.3	Menu aplikace .....	34
9.4	Přátelé.....	35
9.5	Zprávy .....	36
9.6	Schůzky .....	40
9.7	Skupinové konverzace .....	42
10	Realizace webové administrace .....	44
10.1	Přihlašování .....	44

10.2	Přidání uživatele .....	45
10.3	Nasazení na Firebase .....	46
11	Závěr .....	47
12	Seznam použité literatury .....	48
13	Seznam obrázků .....	49
14	Seznam použitých zkratk .....	51
	Přílohy.....	52
15	Obsah přiloženého CD .....	52



# 1 Úvod

Tématem této bakalářské práce je analýza a následná implementace jednoduchého systému pro rychlý přenos zpráv tzv. instant messaging. V dnešní době se jedná o velmi rozšířený segment aplikací, který má již dlouhou tradici mezi uživateli a také tento segment v posledních letech zažil významný technologický pokrok, například díky společnostem jako jsou Facebook, Google, Skype a mnoha dalším, které se aktivně podílejí na zlepšování současných řešení.

Zajímavým aspektem v realizaci vlastního systému je použití Firebase, což je cloudové řešení typu *backend-as-service*, které slouží v systému jako backend mobilní aplikace. Jedním z cílů tohoto projektu je popis systému Firebase, mimo jiné také formou vlastní realizace systému. Oboje současně poskytne ucelené informace o tomto způsobu řešení. Jelikož v současné době se zatím nevyskytuje moc ucelených informací, zkušeností či případně případových studií o používání systému Firebase.

Vlastní systém se skládá ze dvou segmentů: mobilní klientské aplikace pro Android a serverové aplikace, která je realizována pomocí systému Firebase, jenž poskytuje rozhraní android aplikaci tak vnějším systémům pro integraci. K administraci systému bude vytvořena webová aplikace.

Aplikace může nalézt smysl v nějaké menší organizaci, která potřebuje škálovatelné a otevřené řešení pro osobní komunikaci a šíření hromadných zpráv mezi zaměstnanci. Stejně tak může dobře posloužit jako součást komplexnější aplikace, či jen zůstat jako studijní ukázka s dokumentací pro studenty.

Klientská Android aplikace je vyvíjena v jazyce Java. Webová administrace bude vyvíjena pomocí frameworku Angular2.

## **2 Instant messaging**

Instant messaging představuje komunikaci mezi dvěma či více uživateli v reálném čase přes počítačovou síť např. internet. Komunikace může probíhat v textové formě, ale v dnešní době se můžeme setkat s komunikačními programy, které rozšiřují funkčnost o zasílání multimediálních souborů, hlasu či videa. Jedná se o flexibilnější komunikaci, než pomocí emailu, který není primárně určen pro komunikaci v reálném čase. Jednou z dalších výhod oproti emailu je možnost sledování přítomnosti uživatele.

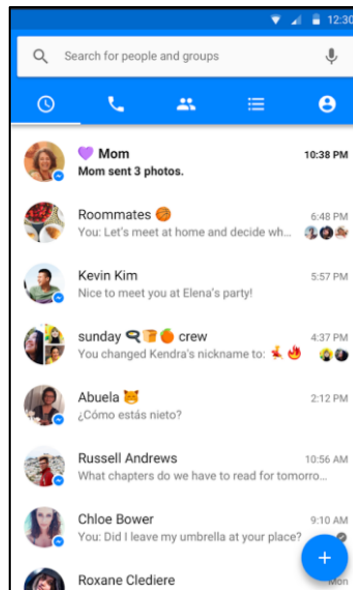
### **2.1 Existující softwarová řešení**

Na současném trhu existuje již mnoho řešení pro instant messaging. V této části práce zhodnotím systémy, které jsem si vybral a se kterými mám již osobní zkušenost a mohu je subjektivně zhodnotit. Testovat budu mobilní verze aplikací, konkrétně na platformě Android.

#### **2.1.1 Facebook Messenger**

Sociální síť Facebook provozuje aplikaci Messenger, která slouží jako komunikační prostředek mezi uživateli sociální sítě. Aplikace je poměrně masivně rozšířená. První verze aplikace byla vydána v srpnu 2011. Aplikace prochází častými updaty a změnami UI. Messenger umožňuje posílání zpráv uživateli či skupině, uskutečňování hovorů, nebo zasílání multimediálního obsahu.

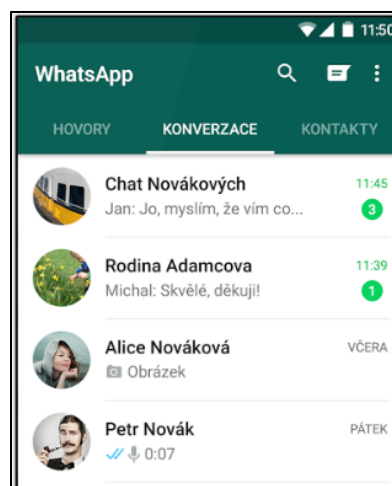
Primární účel aplikace je rozšíření funkčnosti sociální sítě. Při užívání aplikace musíme navíc počítat s podmínkami a politikou, kterou Facebook má např. v souvislosti s autorskými právy, filtrování obsahu atd. Pokud navíc přihlédneme k častým updatům aplikace, které výrazně mění user experiences, lze tvrdit, že to není vhodná komunikační aplikace pro firmu, která hledá stabilní řešení, je spíše vhodnější např. pro zájmovou skupinu či osobní použití.



Obrázek 1: Ukázka aplikace Messenger

### 2.1.2 WhatsApp

Aplikace umožňující výměnu zpráv, zakládání skupinových konverzací a přenosu multimediálních souborů. Jedná se o velmi minimalistickou aplikaci. Mezi její zvláštnosti patří identifikace pomocí telefonních čísel resp. synchronizace kontaktů aplikace s kontakty v telefonu (tel. čísla). Aplikace byla vydána v roce 2009. Aplikaci bych doporučil spíše jednotlivcům, pro firemní účely se moc nehodí, už jen z důvodu existence pouze mobilní verze aplikace.

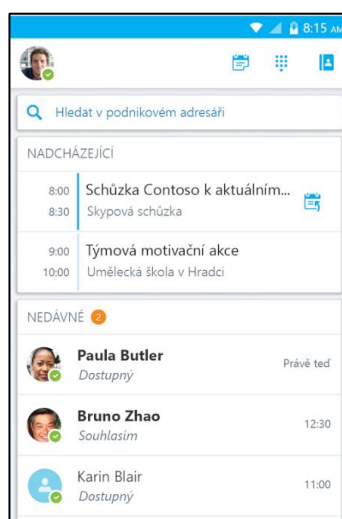


Obrázek 2: Ukázka aplikace WhatsApp

### 2.1.3 Skype

Aplikace umožňující primárně hlasové či video hovory, výměnu zpráv, pořádání skupinových konverzací a přenos multimediálních souborů. Poměrně jednoduchá aplikace s příjemným UI. Výhodou je dostupnost klienta na PC, či online ve webovém prohlížeči. Skype nabízí i customizované řešení pro firmy. První verze se objevila v roce 2003 (Android 2013).

Skype používá vlastní protokol ke komunikaci, který je privátní. Skype lze označit jako vhodné řešení pro firmy i ostatní uživatele. Skype řešení pro firmy je však zpoplatněno (*platí se za každého uživatele*).



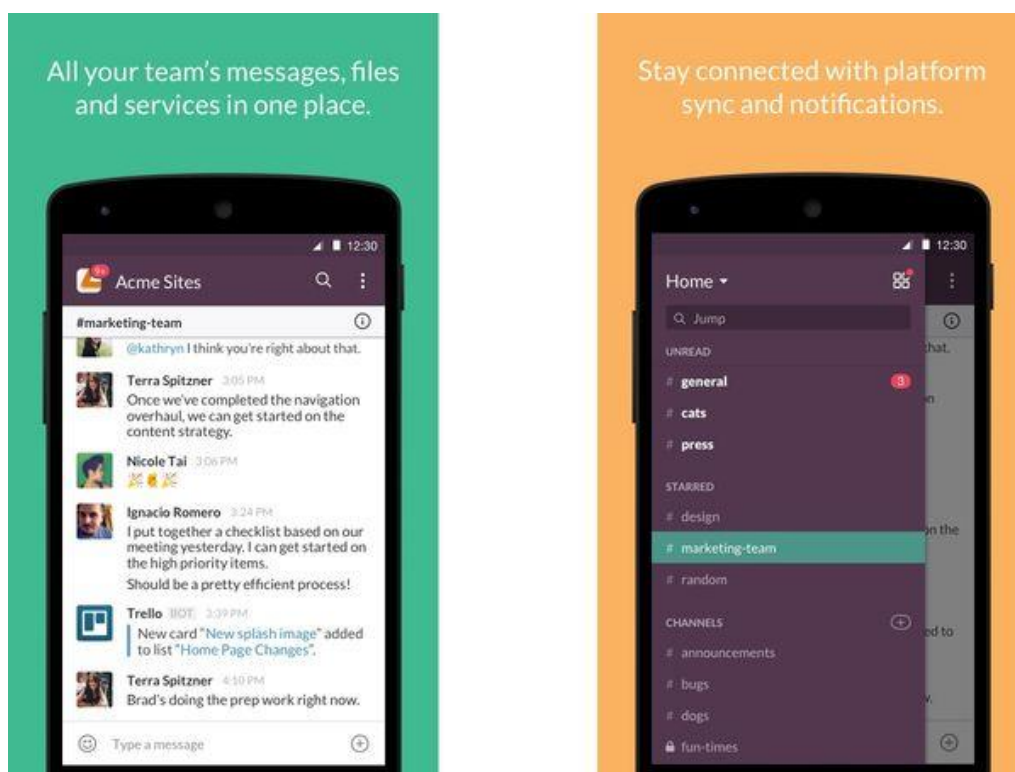
Obrázek 3: Ukázka aplikace Skype

### 2.1.4 Slack

Aplikace speciálně vyvinuta pro týmovou kooperaci, umožňuje posílání zpráv a uskutečňování hovorů mezi uživateli. Slack také umožňuje pohodlně tvořit informační kanály a týmy, ke kterým se uživatelé mohou připojovat a odebírat novinky. Team/kanál má nástěnku na které mohou členové přidávat zprávy či jiný obsah.

Výhodná je i možnost integrace různých informačních zdrojů např. Github, Bitbucket, Twitter. Aplikace je velmi intuitivní a má velmi zdařilou koncepci. Další výhodou Slacku je možnost poměrně velké škálovatelnosti díky API. Aplikace Slacku je dostupná na mnoha platformách. První verze se objevila v roce 2013.

Slack se primárně hodí pro firmy, pro ostatní uživatele pravděpodobně nebude mít velké využití. Nevýhodou Slacku je poměrně vysoká cenová náročnost. V porovnání se Skype, který začíná již na 50 Kč za uživatele měsíčně se Slack pohybuje v částkách od 170 – 323 Kč za uživatele měsíčně. Slack však umožňuje používání ve free režimu s celou řadou omezení, tou nejpálčivější je pouze 10 tisíc zpráv, starší zprávy se odmazávají.



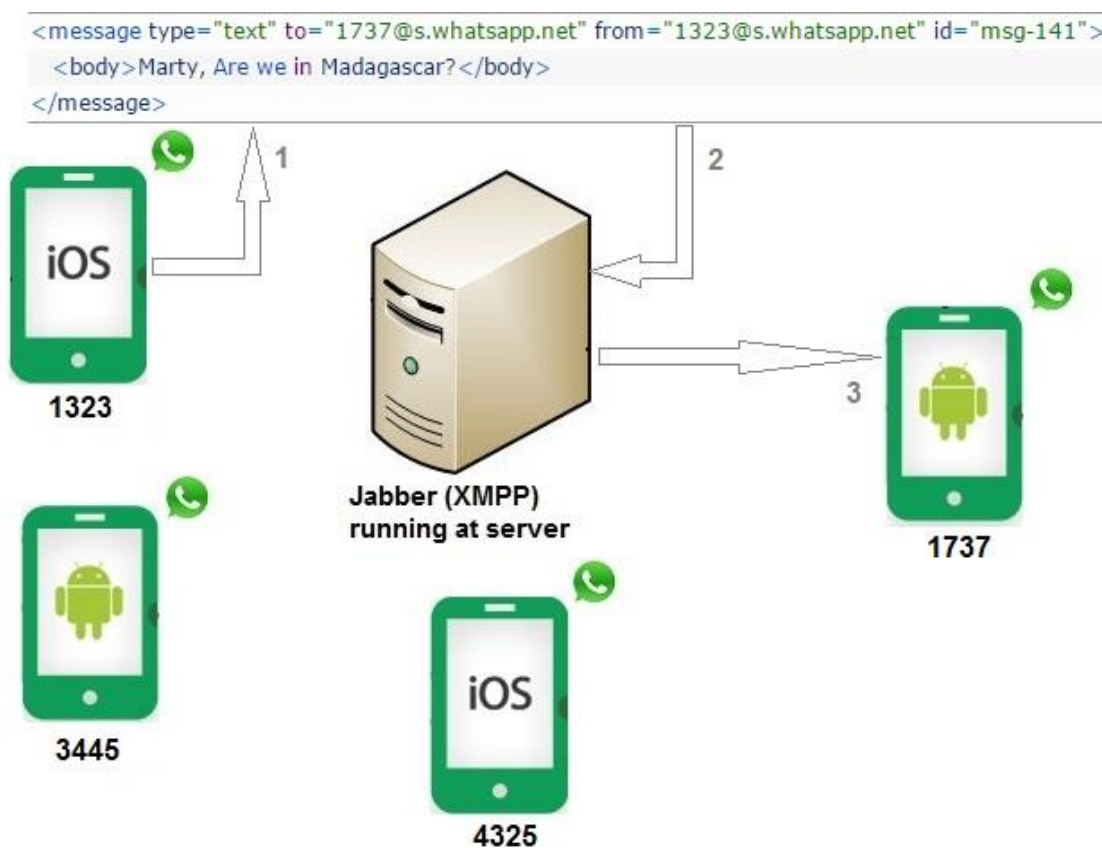
Obrázek 4: Ukázka aplikace Slack

### 3 Technologická řešení

Princip fungování většiny systému poskytující IM lze popsat jako server, či skupinu serverů přes, které probíhá komunikace klientských aplikací. Klient odešle zprávu na server, který zajistí její doručení adresátovi. Většina zmiňovaných aplikací ukládá historii komunikace na server, například Whatsapp je výjimkou, která uchovává komunikaci šifrovaně na klientském zařízení. Většina výše zmíněných aplikací používá ke komunikaci protokol XMPP.

#### 3.1 Protokol XMPP

XMPP (Extensible Messaging and Presence Protocol) je open source komunikační protokol, který ke komunikaci používá formát XML. Primárně slouží k zasílání zpráv typu instant messaging, avšak již od začátku je počítáno s možným rozšířením například o VOIP služby. První zmínky o protokolu se objevují již v roce 1998, v roce 2004 byl protokol standardizován. Protokol je založen na architektuře klient-server. Klienti nekomunikují přímo, ale přes decentralizované servery (podobně jako email). Uživatel má možnost volby serveru, připojí se k tomu, ke kterému má důvěru. Ve světě XMPP neexistuje centrální server, který by spojoval uživatele, avšak díky proprietární implementaci XMPP protokolu ve výše uvedených aplikacích nemusí být dogmaticky dodržovány všechny standardy protokolu. Uživatelé musí být jednoznačně identifikovatelní, většinou pomocí tzv. JID (*username@server.com*).



Obrázek 5: Ukázka komunikace WhatsApp pomocí XMPP

### 3.2 Návrh vlastní implementace

Během zkoumání stávajících řešení bylo třeba hledat způsoby, jakými by bylo možné vytvořit vlastní systém pro IM, který bych byl schopen vyvinout vlastními silami a přitom aby byl dostatečně robustní.

První návrh bylo použití webových služeb, což pod drobnohledem nebyl úplně ideální přístup, než jak se zpočátku jevil. Počáteční výhoda byla poměrně jasná a to jednoduchost řešení. Systém by měl server, který by byl přístupný pod veřejnou adresou a nabízel by určitou paletu potřebných služeb pro klienta. První úskalí spočívalo v samotném faktu, že webové služby nejsou primárně určené pro systémy v reálném čase, jako další problém byla samotná efektivita síťového provozu z klientských aplikací, spočívající v neustálých dotazech na stav/aktualizace. Řešení by

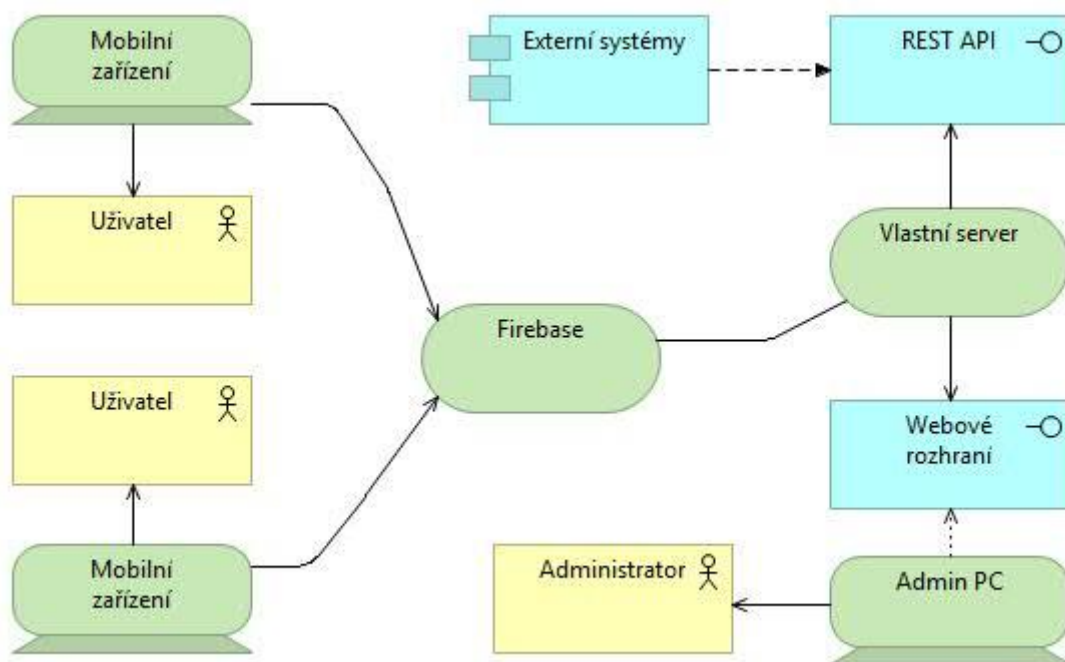
mohlo být například tunelování pomocí VPN ze serveru, který by sám posílal nové události na zařízení, avšak realizace by byla velmi náročná.

Druhá možnost řešení byla pomocí XMPP protokolu v němž funguje velká spousta reálně nasazených aplikací. Výhoda by byla v poměrně dostupné paletě klientských knihoven pro tvorbu XMPP klienta. Horší situace je však při realizaci serveru XMPP, který se ve většině případů používá jako již existující robustní open source serverové řešení, ke kterému se integrují customizované součásti. Realizace XMPP serveru na „zelené louce“ by byla časově náročná a velmi neefektivní v porovnání s použitím existujícího open source řešení. Pro naše použití je open source řešení poměrně zbytečně robustní, složitě customizovatelné a využili bychom jen malou část jeho funkčnosti.

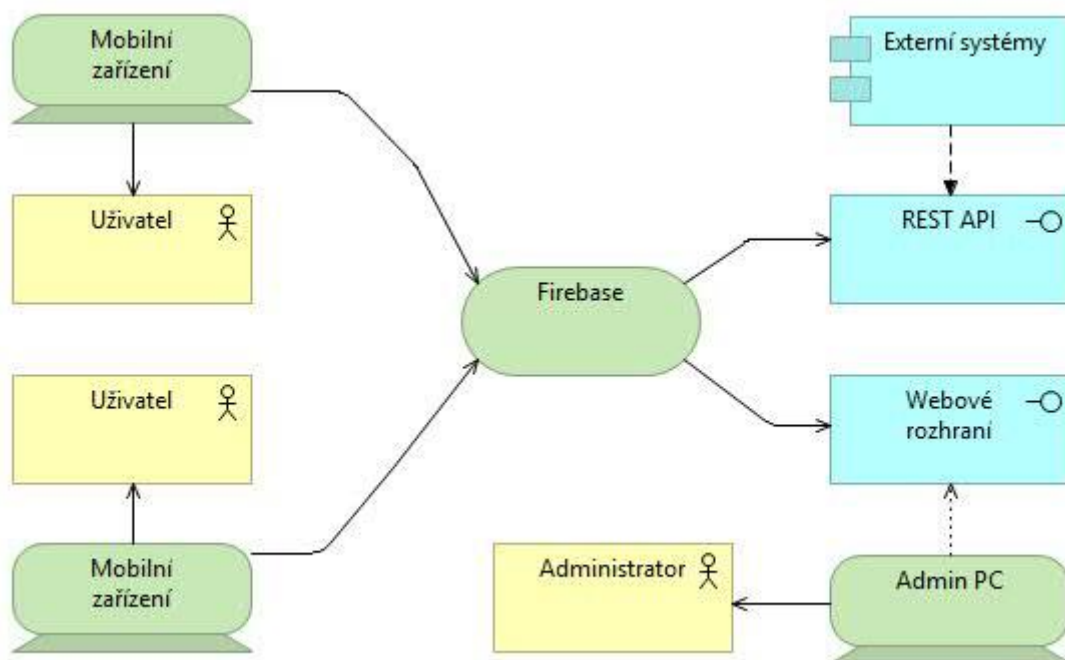
Třetí a finální technologické řešení bylo použití systému typu mBaaS (mobile Backend as a Service). Toto řešení zažívá v posledních letech poměrně strmý vzestup, jelikož většina potřebné funkcionality na klientských zařízeních je již implementována v cloudu a vývojář je již pouze využívá ve své aplikaci. Tato strategie může poměrně výrazně ušetřit náklady a čas na vývoj serverové části aplikace. Často se může stát, že si plně vystačíme s poskytovanými službami na cloudu a naše aplikace má podporu robustního backendu, bez jediného řádku serverového kódu. Což umožňuje malému týmu, či jednotlivci tvořit poměrně rozsáhlé a komplexní mobilní a webové aplikace.

Výhoda tohoto řešení pro nás je, že nás odstíní od komunikace mezi zařízením a serverem. Aplikace bude bez problému fungovat i ve veřejné síti internetu. Mezi další výhody patří, že nebudeme muset vlastnit server s veřejnou adresou, takže aplikaci může mít nasazenou i organizace, která nemá vybudovanou IT infrastrukturu se servery.





Obrázek 6: Návrh vlastního systému (původní Firebase)



Obrázek 7: Návrh vlastního systému (aktualizovaný Firebase)

## 4 Služby typu mBaaS

První mBaaS služby se začali objevovat v roce 2011. Většina poskytovatelů v současné době nabízí poměrně stejnou paletu základních služeb. Poskytují cloudové úložiště pro web či mobilní zařízení, registraci a přihlášení včetně integrace sociálních sítí, push notifikace, analytické nástroje pro mobilní aplikaci. Výhodou je, že služby jsou primárně určeny pro použití v reálném čase.

Mezi nejvýraznější výhody použití mBaaS služeb patří úspora zdrojů v projektu, „oživení“ aplikace, sběr dat, škálovatelnost. K nevýhodám patří proprietární uzamčení tzv. vendor lock, nemožnost kompletní kontroly, možnost nedostupnosti služeb.

Mezi známé uživatele těchto služeb ve svých aplikacích patří např. eBay, Warner Bros., Udacity.

### 4.1 Porovnání současných poskytovatelů

Dnes se můžeme setkat s mnoha poskytovateli backend řešení pro mobilní aplikace. V této části bych rád představil nejznámější zástupce a porovnal jejich výhody a nevýhody. Vybral jsem si tři zástupce a to Firebase, Kinvey, Backendless.

#### 4.1.1 Firebase

Společnost byla založena roku 2011, od roku 2014 ji provozuje společnost Google. Platformy, které Firebase nativně podporuje a jsou pro ně odpovídající SDK: iOS, Android, Web (Angular, Java Script atd.), C++, Unity, Java (pouze pro server). Databáze je zde realizována jako JSON objekt, který můžeme rozšiřovat přidáváním potomků. Firebase neobsahuje geografické služby.

Firebase již umožňuje tvoření serverového kódu (známé také jako Parse Cloud Code). K tomu slouží Functions, které implementujeme v Node.JS a slouží například jako trigger realtime DB, případně jako obsluha REST API. Dále můžeme použít customizovaný server komunikující s Firebase, pro tyto účely je knihovna pro jazyk Java. Firebase má možnost definovat uživatelské role pro real-time databázi a rolím přiřazovat oprávnění.

Firestore poskytuje poměrně příznivou cenovou politiku. Většina aplikací si vystačí s tarifem Spark, který je zdarma. Google zde nenabízí žádné úlevy pro studenty, open source, s odvoláním na štědrost tarifu Spark. Další tarif je Flame a pak následuje vlastní customizovaný tarif Blaze, který je přesně na míru uživatele a platí skutečně jen za to, co používá.

#### **4.1.2 Kinvey**

Společnost byla založena roku 2010. Platformy, které Kinvey nativně podporuje a jsou pro ně odpovídající SDK: iOS, Android, Web (Angular, Node.js atd.), Java. Databáze je zde realizována jako JSON objekt, který můžeme rozšiřovat přidáváním potomků.

Pro DBMS je použita MongoDB. Kinvey také neobsahuje geografické služby, avšak objekty mohou mít položku geologic property. Kinvey umožňuje tvoření serverového kódu (PCC) pomocí JavaScriptu, nebo triggerů či předdefinovaných funkcí. Kinvey má vertikálně odstupňované oprávnění např. pro databázové záznamy a určené vlastnictví např. pro celou třídu záznamů atd.

Kinvey také nabízí free tarif, který je v porovnání s Firestore poměrně střídmy a placené varianty se nedají škálovat dle potřeby.

#### **4.1.3 Backendless**

Společnost byla založena roku 2012. Platformy, které Backendless nativně podporuje a jsou pro ně odpovídající SDK: iOS, Android, Java Script, Java a jako jediná z vybraných Windows Phone (.NET). Databáze je zde realizována jako klasická relační databáze používající MySQL jako svůj DBMS. Backendless obsahuje geografické služby pomocí relace geopointů a datových objektů.

Backendless umožňuje tvoření serverového kódu pomocí jazyka Java, případně se dají použít trigger, předdefinované funkce či timery. Backendless má řízení přístupů na úrovni uživatelských rolí a uplatňování vlastnictví na data a soubory.

Backendless poskytuje poměrně složitou cenovou politiku. Avšak na menší aplikace se lze vejít do Free limitu.

## 4.2 Shrnutí

Souhrnně lze říct, že každý poskytovatel má ve své řešení silné a slabé stránky. U Firebase je značná nevýhoda, že nepodporuje desktopové aplikace, jelikož SDK pro Javu je určeno pouze na serverovou aplikaci, díky přítomnosti privátního klíče v aplikaci. U ostatních problém spočíval v poměrně složité cenové politice, která je poměrně nepřehledná a částka se u složitější aplikace může poměrně prudce zvýšit.

Pro realizaci systému jsem si vybral Firebase, který i přes zmíněné nevýhody poskytuje štědré free limity. Firebase má pod správou společnost Google, tudíž se dá očekávat dobrá dostupnost a poměrně jisté a bezpečné uložení dat. Mezi další výhody patří i to, že funkčnost se díky vlastnictví Googlem má tendenci rychle rozšiřovat, tudíž je zde oproti jiným poskytovatelům dlouhodobější perspektiva pro aplikaci. O tom svědčí i to, že během tvorby bakalářské práce, přibyl zásadní segment funkčnosti a to Cloud Functions.

## 5 Firebase

V této části práce budou představeny funkce systému Firebase, které se vztahují nějakým způsobem k realizované aplikaci. Funkce se dělí do čtyřech skupin: Analytics, která poskytuje data o používání aplikace, druhou je Develop, která poskytuje funkce především vývojářům. Třetí skupina Grow poskytuje funkce pro podporu virálnosti aplikace. Čtvrtá skupina Earn poskytuje prostředky pro monetizaci aplikace, například pomocí reklamy v aplikaci.

### 5.1 Analytics

Základní modul, který musí být vždy přítomný v aplikaci, byť se analytické funkce nepoužívají. Tvoří totiž jádro Firbase. Analytics umožňuje odesílat až 500 různých událostí z aplikace, každá událost může obsahovat až 25 doplňujících atributů. V základu jsou některé běžné události předdefinovány. Analytics nefunguje real-time, ale v rámci úspory baterie zařízení, odesílá data v dávkách zhruba po jedné hodině. Služba má poměrně propracované možnosti analýzy dat v konzoli aplikace.

Analytics jako takové jsou bez omezení, co se týče kapacity uchovávaných dat, avšak takzvané BigQuery pro customizované dotazování již zdarma provádět nelze. V naší aplikaci budeme sledovat zájem uživatelů o jednotlivé části a četnost používání aplikace.

### 5.2 Develop

Tato skupina funkcí bude pro naši aplikaci stěžejní, jelikož nám jde primárně o funkčnost aplikace.

#### 5.2.1 Cloud Messaging

Funkce umožňující zasílání zpráv do mobilních zařízení z webového rozhraní Firebase, nebo FCM serveru. Zprávy se dělí na notificační (limit 2kB) a datové (limit 4kB) zprávy. Zprávy lze posílat na jednotlivá zařízení, skupiny zařízení, nebo zařízení zapsaná k odběru určitého tématu. V aplikaci tato funkčnost může být použita při zasílání hromadných informací a upozornění na novou zprávu.

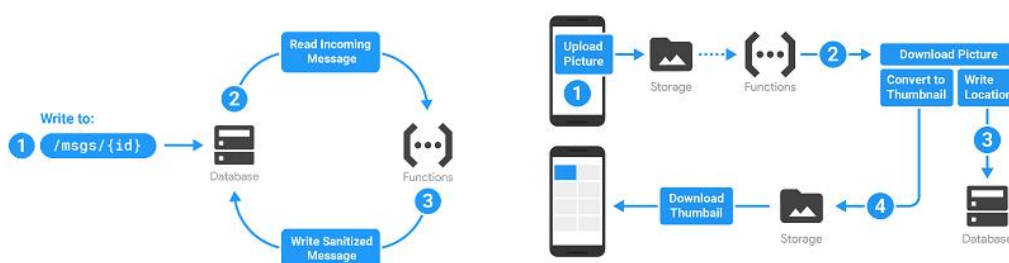


Obrázek 8: Schéma FCM

## 5.2.2 Cloud Functions

Tato novinka umožňuje pohodlné rozšiřování funkčnosti Firbase pomocí psaní malých skriptů v Node.JS, aniž bychom museli budovat vlastní server jak tomu bylo doposud. Tato novinka je dosud v betaverzi, tudíž všechna funkčnost není dostatečně zadokumentována a potenciál není využit na plno.

Využití nalezene pro realizaci triggerů nad databází, které mohou vyvolávat akce nejen v databázi, ale v celé Firebase. Jako komplexní příklad lze uvést nahrání obrázku z klienta. Funkce uloží obrázek na Firebase Storage a současně zapíše údaje do databáze. Pomocí Functions můžeme realizovat REST API (http triggers), triggeru na autentifikaci, analytiku.



Obrázek 9: Ukázky využití Functions

### 5.2.3 Authentication

Tato funkčnost umožňuje integraci přihlašování, která podporuje i přihlašování z jiných sítí. Momentálně podporuje Firebase Google, Facebook, Github, Twitter účty, případně lze poskytnout uživatelům i anonymní účet podle potřeby.

Autentizaci provádí Firebase, který poskytuje vývojáři stejný způsob přístupu k údajům, z různých sítí. Po přihlášení je k dispozici jméno, email, url profilového obrázku uživatele a také identifikátor sítě ze které se uživatel přihlásil. Každému uživateli Firebase přidělí jedinečný user-id. Po úspěšném přihlášení se údaje o uživateli uloží do databáze.

Firebase také umožňuje správu uživatelských session napříč webem a mobilními aplikacemi. V naší aplikaci bude funkce využita pro přihlašování, pomocí emailu a hesla, Facebooku a sítě Google.

### 5.2.4 Realtime Database

Tuto funkčnost bych se nebál označit jako vlajkovou loď celého systému Firebase. Jedná se o poměrně sofistikovaný systém NoSQL databáze, který umožňuje synchronizaci dat mezi zařízeními v reálném čase. Například změna či přidání záznamu se okamžitě projeví v okně webového prohlížeče i na mobilní aplikaci, bez jakékoliv implementace navíc, ze strany vývojáře.

Databáze je také optimalizována pro práci v offline režimu, který u mobilních zařízení nastává poměrně často a nečekaně, data se v offline režimu ukládají na lokální cache a po získání připojení se automaticky se synchronizují. Databáze umožňuje řízení přístupu jednotlivým uživatelům, či skupinám uživatelů pomocí pravidel, umožňující reflektovat uživatelské parametry (např. user-id) a jiné proměnné v databázi.

Byť je databáze koncipována jako JSON objekt, můžeme ovlivňovat strukturu ukládaných dat pomocí širokého spektra podmínek. V realizované aplikaci bude databáze hrát klíčovou roli a bude se o ni opírat většina funkcí aplikace a celého systému.

Databáze v naší aplikaci bude provozována na tarifu Spark, tudíž se na ní vztahují určitá omezení. Databáze může mít maximálně 100 současně běžících připojení, 1GB velikost uložště, traffic 10GB měsíčně, databáze také neumožňuje automatickou záloh. Žádný z těchto limitů se naší aplikace zásadně nedotkne.

### **5.2.5 Storage**

Uložiště, které slouží pro ukládání uživatelského obsahu. Oproti klasickému cloudovému úložišti má několik výhod. Umožňuje ochranu před neoprávněným přístupem např. nepřihlášenému uživateli aplikace. Všechny přenosy, které probíhají mezi uložštěm a zařízením jsou zabezpečené.

Stejně jako u databáze je zde ošetřeno chování při výpadku připojení, tak aby po opětovném připojení pokračovalo stahování dále. Storage podobně jako databáze obsahuje v našem tarifu určitá omezení.

Velikost uložště je 5GB, denní traffic je 1GB a jsme limitováni i počtem operací 20k upload a 50k download. Žádné z těchto omezení by se nemělo dotknout aplikace. Storage může být použita v aplikaci jako uložště pro multimediální zprávy mezi uživateli

### **5.2.6 Hosting**

Hosting představuje klasický webhosting, který neobsahuje žádnou specifickou vlastnost, která by zde stála za zmínku. V aplikaci Hosting nebude použit. Hosting je limitován 1GB uložště a trafficem 10GB za měsíc.

### **5.2.7 Test Lab**

Funkce umožňující otestování naší aplikace na různých zařízeních. Testy mohou být robotické, kdy Firebase objevuje jednotlivé části aplikace a testuje je náhodným způsobem, nebo můžeme dodat vlastní testovací scénáře. Výsledky testů jsou podloženy logy, videem a snímky obrazovky. Aplikace bude využívat základní robotické testování, které by mohlo objevit chyby a snížit čas potřebný na manuální testování aplikace,



které by se soustředilo pouze na testování běžného chování uživatele. V našem tarifu jsme omezeni 15 testy denně.

### **5.2.8 Crash Reporting**

Umožňuje reportovat pády aplikace v reálném provozu a shromažďovat je do webové konzole aplikace. Součástí reportu je kompletní výpis stack trace, tudíž můžeme lépe lokalizovat pád a jeho příčinu. Do aplikace si také můžeme přidat naše vlastní podpůrné logy, které případně uvidíme při pádu. Tato celá funkčnost je plně zdarma. V naší aplikaci bude tato funkčnost použita.

## **5.3 Grow**

Tuto část funkčnosti představím ve stručnosti, jelikož realizovaný systém nebude využívat nic ze skupiny Grow. Avšak znalost těchto funkcí je velmi vhodná z pohledu marketingu a samotného šíření aplikace, protože dokážou zvýšit viralitu aplikace.

### **5.3.1 Notifications**

Notifications jsou nadstavbou pro výše popsany Cloud Messaging, poskytuje nástroje na zacílení notifikací na určitou skupinu, např. z Analytics. Použití Notifications je vhodné například na cílené nabídky, reklamu, nebo poskytování bonusů pro konkrétní segment uživatelů aplikace.

### **5.3.2 Remote Config**

Remote Config umožňuje nastavovat vzdáleně aplikacím určité parametry, aniž by uživatelé aplikace museli provést update na novější verzi. Může se například hodit postupné uvolňování nové funkčnosti do ostrého provozu, případně při velké chybovosti funkčnost deaktivovat aniž by uživatel musel řešit update na opravenou verzi aplikace.

### **5.3.3 App Indexing**

App Indexing umožňuje propojení aplikace a vyhledávače Google, pokud aplikace obsahuje stejný obsah (klíčová slova) jako webová stránka, je uživatel z mobilního zařízení odkázán speciálním odkazem (Android App Link) do aplikace, ve které se otevře hledaný obsah.

### **5.3.4 Dynamic Links**

Úzce souvisí s Android App Indexing, díky této funkci můžeme odkazovat např. z webu na určitou konkrétní aktivitu aplikace, výhodou je, že pokud uživatel danou aplikaci nemá, tak se automaticky přesměruje na získání aplikace, po nainstalování se aplikace otevře na požadované aktivitě.

### **5.3.5 Invites**

Umožňuje uživateli zasílání pozvání k instalaci aplikace svým přátelům, např. ze soc. sítí pomocí emailu, či sms.

## 6 Požadavky na systém

V této části budou popsány hranice realizovaného systému a požadavky na něj. Definovány jsou základní požadavky, které se realizují v první iteraci vývoje. Rozšířené požadavky patří k dalším iteracím vývoje systému, případně jako námět pro studenty k samostatné realizaci.

### 6.1 Požadavky na klientskou aplikaci

Aplikace bude fungovat na mobilní platformě Android, aplikace podporuje od verze 4.4 KitKat, čímž pokryjeme až 85%<sup>1</sup> používaných zařízení v současné době. Primární použití bude na mobilních zařízeních, aplikace tudíž nebude optimalizována na větší obrazovky zařízení např. tablety.

1. Posílání zpráv mezi uživateli
2. Přihlašování
3. Možnost mít přátele
4. Schůzky (Broadcasty)
5. Skupinové konverzace

### 6.2 Požadavky na serverovou aplikaci

Serverová aplikace bude realizována pomocí služby Firebase. Touto volbou můžeme garantovat nefunkční požadavek v podobě dobré škálovatelnosti a dostupnosti serverové aplikace. Pro nastavení systému bude sloužit webová administrace vyvinuta v Angular2. V našem systému bude sloužit pouze pro vytvoření nových uživatelů mobilní aplikace.

---

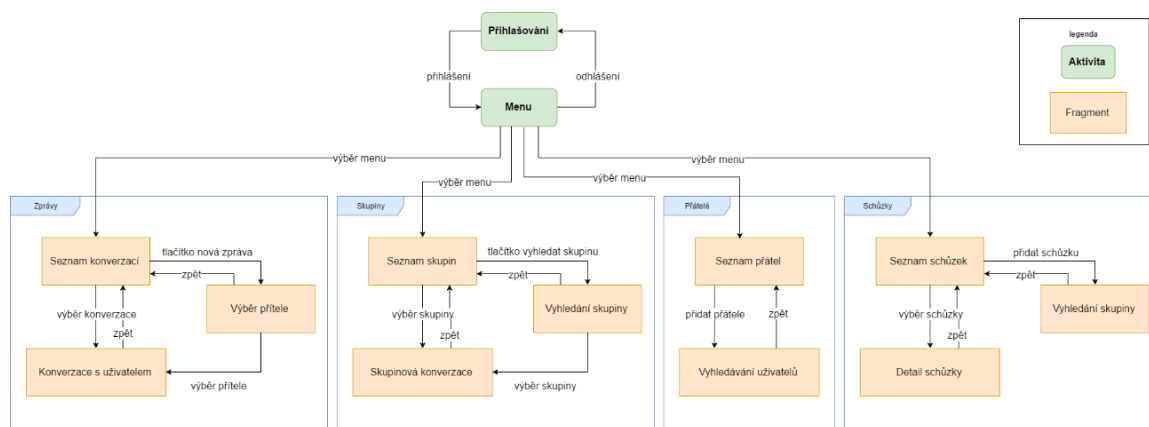
<sup>1</sup> <https://developer.android.com/about/dashboards/index.html> k datu 5.1 2017

## 7 Návrh mobilní aplikace

V této části bude popsán úvodní návrh před samotnou realizací systému.

### 7.1 Schéma aktivit a fragmentů

Nejprve sestavíme podle požadavků scénář chování aplikace, který bude složen z jednotlivých aktivit a fragmentů a vzájemné vazby mezi nimi. Aplikace bude využívat fragmenty, díky tomu může být v budoucnu lépe udržitelná a snáze se bude rozšiřovat o další funkčnost.



Obrázek 10: Schéma aktivit a fragmentů

Úvodní obrazovka, která se zobrazí nepřihlášenému uživateli jako první, bude aktivita k přihlášení. Po úspěšném přihlášení se uživateli zobrazí aktivita Menu, která bude sloužit jako kontejner pro fragmenty. Díky tomu budeme mít menu pouze v jedné aktivitě a to samé platí pro aplikační lištu, vyvarujeme se zvýšené granularitě kódu aplikace a případné úpravy či rozšíření budou velmi snadno proveditelné.

Sada fragmentů zpráv umožňuje uživateli zobrazit seznam konverzací, odesílání zpráv uživatelům a prohlížení jednotlivých konverzací.

Sada fragmentů skupiny umožňuje uživateli komunikovat v rámci dané skupiny s více uživateli najednou, jde v podstatě o analogickou funkčnost jako v případě sady zpráv, akorát v rámci více uživatelů aplikace.

Sada fragmentů přátelé umožňuje uživateli vyhledávat a přidávat nové přátele.

Sada fragmentů schůzky umožňuje uživateli plánovat schůzky se svými přáteli a prohlížet si již vytvořené schůzky.

## 7.2 Návrh schématu databáze

Firebase real-time databáze je dokumentová databáze, reprezentována jako velký JSON objekt. Při návrhu je třeba minimalizovat hloubku zanoření i za cenu redundance dat, jelikož databáze ztrácí poměrně rychle výkon při nedodržování ploché struktury (doporučuje se hloubka 3-4, maximálně umožňuje 32).

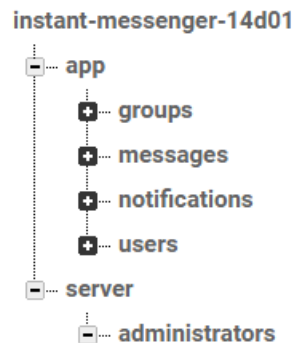
Na nejvyšší úrovni rozdělíme objekt na data pro aplikaci (*app*) a pro systém (*server*). Dále rozdělíme data pro aplikaci na jednotlivé sekce a to skupiny (*groups*), zprávy (*messages*), notifikace (*notifications*), uživatelé (*users*). Data pro systém budou obsahovat sekci administrátoři (*administrators*). Další jemnější modelování jednotlivých sekcí bude probíhat během samotného vývoje aplikace.

### 7.2.1 Sekce dat pro aplikaci

- *skupina* obsahuje informace o skupinách (zprávy, členové, informace)
- *zprávy* obsahuje zprávy pro uživatele systému (analogie k poštovní schránce)
- *notifikace* obsahuje všechny události, které má Firebase notifikovat na zařízeních
- *uživatelé* obsahuje uživatele v systému a informace vztahující se k nim

### 7.2.2 Sekce dat pro systém

- *administrátoři* obsahuje ty uživatele, kteří mají přístup do webové administrace



Obrázek 11: schéma databáze

## 8 Návrh webové administrace

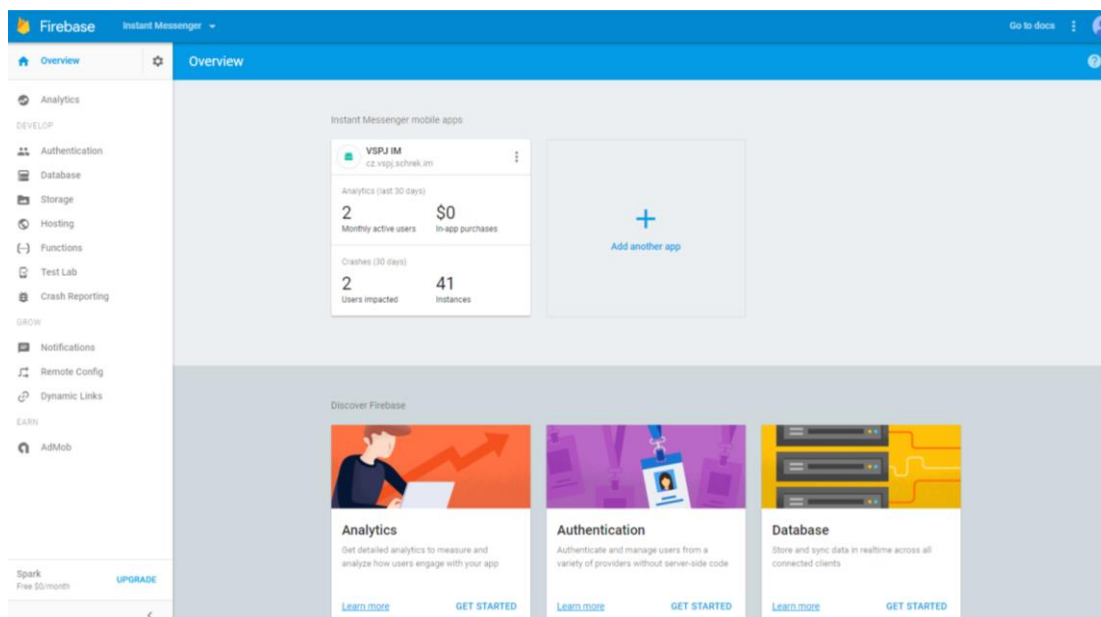
Webová administrace bude sloužit zatím pouze k přidávání uživatelů do systému. Webová aplikace bude obsahovat úvodní stranu k přihlášení a po přihlášení bude přesměrován na hlavní stránku, která bude mít záložkovou horizontální navigaci mezi stránkami.

Webová aplikace bude realizována v Angular2, tudíž jako single-page aplikace což je v případě administrace vhodný návrh. Pro nastýlování bude použit Bootstrap, abychom minimalizovali čas při vývoji a přitom zachovali určitou kvalitu designu.

Webová aplikace bude hostována přímo na Firebase Hosting.

## 9 Realizace mobilní aplikace

Během tvorby mobilní aplikace budou využity standardní vývojové nástroje pro Android (Android Studio). Pro verzování rozšířený systém verzí Git. Před samotný započítím vývoje bylo potřeba založit nový Firebase projekt. K založení je potřeba vlastnit účet od Google, následně na stránce <https://console.firebase.google.com> založíme nový Firebase projekt.

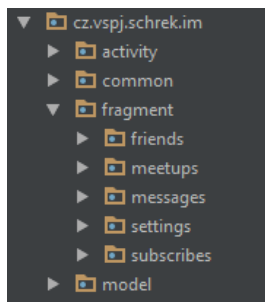


Obrázek 12: Ukázka konzole Firebase

### 9.1 Nastavení projektu

Aplikace bude podporovat pouze mobilní zařízení, nebude umožňovat rotaci (pouze vertikální layout). Všechny specifické požadavky napíšeme do android manifestu, kde také hned na začátku definujeme práva, která má aplikace získat.

Dále si definujeme strukturu balíčků v aplikaci, tak aby jednotlivé fragmenty a aktivity, případně pomocné třídy byly přehledně roztrženy. V balíčku *aktivity* jsou všechny aktivity v projektu. *Commons* sdružuje třídy, které jsou sdíleny napříč projektem, tak abychom kód spravovali na jediném místě a nezvětšovali granularitu kódu. *Fragment* shromažďuje fragmenty aplikace, které jsou dále tříděny dle účelu, jelikož je v aplikaci fragmentů značný počet. Posledním balíčkem je *model*, ve kterém jsou entitní třídy používané v projektu.



Obrázek 13: Ukázka členění balíčků

Také musíme přidat závislosti do Gradle pro Firebase a spojit ho s našim vytvořeným projektem v konzoli. V našem případě nainportujeme moduly *core*, *auth*, *database*. Pro přiřazení k našemu projektu musíme v konzoli získat, konkrétně v nastavení, sekce general, stáhnout google-service.json, který umístíme do složky *app* a aplikace ho v této lokaci bude očekávat po spuštění. Tímto se podařilo zavést do projektu Firebase a spojit ho s projektem v konzoli.

## 9.2 Přihlašování

Pro tvorbu první aktivity mobilní aplikace nejdříve musíme nakonfigurovat modul Authentication ve Firebase konzoli. Pro naši aplikaci zvolíme přihlašovací metodu pomocí emailu. Při výběru přihlašování emailu musíme počítat s přijetím určitých pravidel, například uživatelská jména musí být ve tvaru emailové adresy a jsou kladeny požadavky na bezpečnost hesla. Při nedodržení výše uvedených pravidel nedojde k vytvoření uživatelského účtu. V naší aplikaci tyto požadavky nepředstavují zásadní problém.

Samotná realizace aktivity spočívala v navržení rozložení obrazovky, celý layout je zabalen do relativního pozicování, které nám umožní přesně umístit prvky v kontejneru. Upozornění budou realizována pomocí tzv. Toastů (vyskakovací bublina ve spod obrazovky).

Kromě klasického napojení a obsluhy elementů layoutu musíme v aktivitě zavést samotný Firebase a zavést autentifikaci. To realizujeme pomocí volání statické metody *Firebase.initializeApp* v metodě *onCreate*, která je překrytá v aktivitě. Dále je potřeba přidat listener na změny stavu přihlášení a to provedeme pomocí instančních metod *addAuthStateListener* opět v aktivitou překrytých metodách *onStart* a *onStop*.



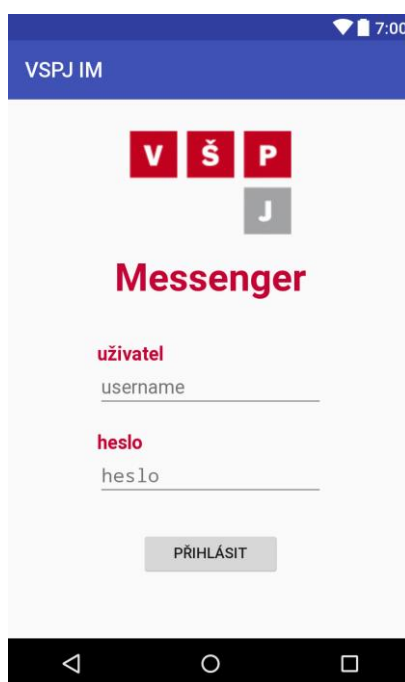
```

private FirebaseAuth.AuthStateListener authListener = (firebaseAuth) → {
    FirebaseUser user = firebaseAuth.getCurrentUser();
    if (null != user) {
        Utils.setUserOnlineState(user.getId());
        startActivity(new Intent(getApplicationContext(), MainActivity.class));
        finish();
    }
};

```

Obrázek 14: Ukázka kódu listeneru

Z ukázky kódu listeneru je patrné, že ke vpuštění do menu je potřebné, aby se změnil stav přihlášení a také byl přiřazen do Firebase uživatel, který se přihlásil. Obsluha chybných přihlášení je řešena přímo callbackem z metody přihlášení. Výhodou tohoto řešení je, že se nemísí logika obsluhy různých stavů, které mohou nastat při přihlášení a samotná autentifikace a následné přesměrování do menu.



Obrázek 15: Layout přihlášení do aplikace

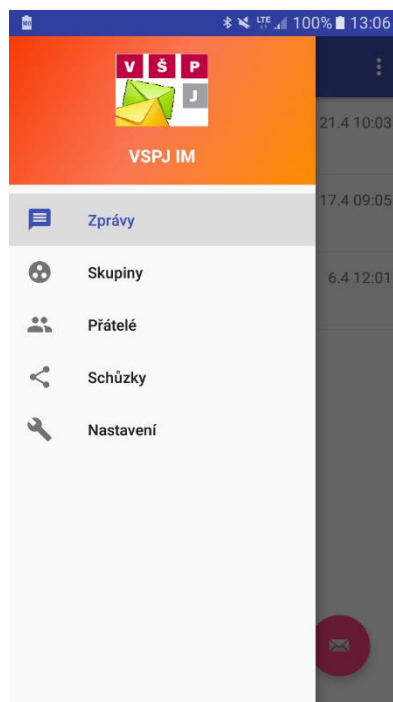
### 9.3 Menu aplikace

Menu aplikace je koncipované pomocí fragmentu a lišty menu, která je umístěna v levé části obrazovky a dle potřeby ji můžeme skrývat a tahem prstu či tlačítkem zobrazit. Jako základní fragment pro zobrazení jsem zvolil zprávy, které budou uživatelé nejčastěji využívat.

Pro samotnou realizaci menu jsem zvolil řešení pomocí šablon, které poskytuje Android Studio. Díky tomu se mi vygenerovala základní kostra, kterou jsem upravil vlastní grafikou. Výhodou tohoto řešení je poměrně srozumitelný kód, ve kterém se snadno dělají úpravy pro vlastní přizpůsobení, a zároveň splňuje standard pro Android zařízení, tak že máme jistotu korektního zobrazení napříč zařízeními a samotnými verzemi Androidu.

Jako první krok jsem zvolil implementaci tzv. toolbaru, což je horní lišta aplikace. Přidal jsem do ni kontextové menu, které obsahuje možnost odhlášení z aplikace. Aktivita musí obsahovat stejnou konstrukci, kterou jsem zmiňoval již při přihlášení, s tím rozdílem že voláme metodu pro odhlášení a uživatele přesměrujeme na přihlašovací stránku. Je to jediná možnost jak se odhlásit, jelikož pokud aplikaci ukončíme či minimalizujeme, stále zůstává přihlášený uživatel a aplikace se spustí přímo v menu.

Další nezbytnou věcí, kterou bylo nutné implementovat, byla obsluha back-stacku aplikace, což představuje zásobník, na který se ukládají fragmenty/aktivity, které jsme otevřeli, a při stisku tlačítka zpět se zásobník vyprazdňuje. Implementoval jsem následující metody: *replaceFragment* vyprázdní zásobník a vloží fragment z parametru, *pushFragment* vloží fragment na vrchol zásobníku a *popBackStack* vyprazdňující zásobník.



Obrázek 16: Menu aplikace

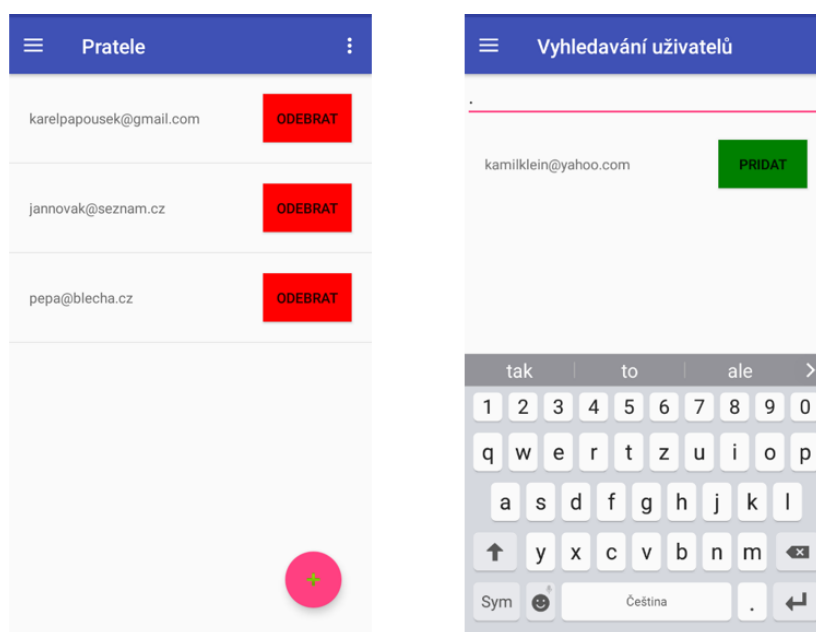
## 9.4 Přátelé

První modul, který byl realizován, je správa přátel uživatele aplikace. K jeho realizaci nepotřebujeme žádné jiné prerekvizity. Prvním krokem bylo vytvořit v databázi strukturu uživatelského účtu. Uživatele v databázi je zapouzdřen pomocí unikátního klíče vygenerovaného pomocí Firebase. Uvnitř obsahuje dvě položky a to seznam přátel, složený z klíče a emailu přítele a informací o vlastním účtu uživatele.



Obrázek 17: Záznam uživatelů v databázi

Samotný modul se skládá ze dvou fragmentů, první slouží jako seznam přátel s možností smazání, druhý fragment slouží k vyhledávání uživatelů v systému. Fragment pro zobrazení přátel je realizovaný jako obyčejný seznam s vlastním layoutem položek, který při stisku tlačítka odebrat odstraní záznam v databázi. Fragment vyhledávání obsahuje také seznam, který se aktualizuje podle změny ve vyhledávací liště. Zde bylo nutné vytvořit funkci, která získá uživatele ze systému a současně vyfiltruje účty, které má uživatel již přidáné. Seznam přátel se uchovává ve statické položce třídy *LoggedUser*, která obsahuje i další informace o uživateli a aktualizuje se každým přihlášením. Vyhledávání uživatelů má také upravený toolbar, stejně jako všechny fragmenty druhé úrovně v aplikaci.



Obrázek 18: Rozložení obrazovek

## 9.5 Zprávy

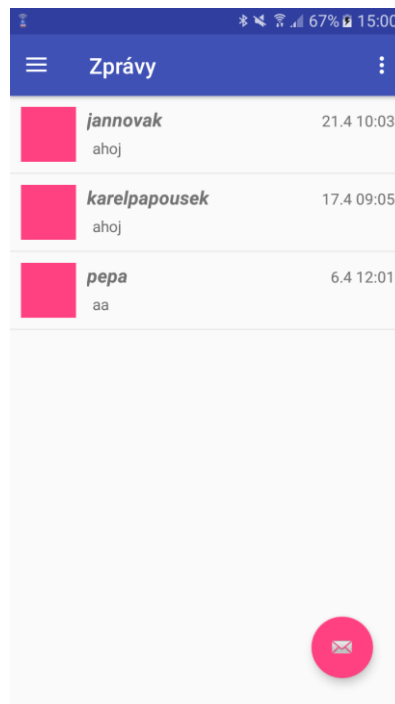
Modul zasílání zpráv mezi uživateli se skládá ze tří fragmentů a to seznamu konverzací, konverzace a vyhledání přítele. Zprávy jsou uživateli doručovány do schránky, každý uživatel má u sebe uloženy obdržené zprávy tj. jeho odeslané zprávy jsou u adresáta ve schránce. Schránka uživatele je v databázi označena jeho unikátním klíčem a uvnitř jsou

klíče jednotlivých odesílatelů. Uvnitř jsou jednotlivé zprávy, jejichž klíč je časové razítko vložení do databáze. Každá zpráva obsahuje položky *read*, která uchovává stav zobrazení, dále pak typ zprávy, který umožňuje například rozšíření o jiný obsah, než jsou textové zprávy a poslední je *value*, což je samotný obsah zprávy.



Obrázek 19: Struktura ukládání zpráv

Seznam konverzací je realizován jako seznam, jehož řádek je složen z ikony přítele, jeho doménového jména a poslední zprávy s časem odeslání. Ve fragmentu je implementován listener, který sleduje schránku uživatele a v případě doručení zprávy aktualizuje údaje v seznamu.



Obrázek 20: Seznam konverzací

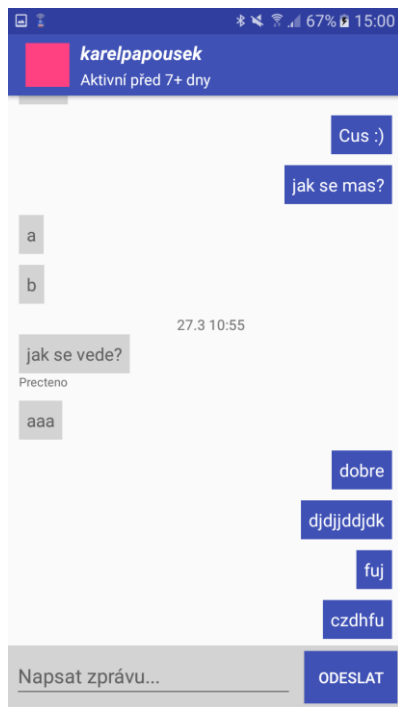
Samotný detail konverzace obsahuje upravenou lištu, která obsahuje ikonu uživatele, jeho doménové jméno a status kdy byl naposledy aktivní.

Status je uložen v sekci uživatele pod položkou *status*, který obsahuje id zařízení a čas změny. Id zařízení se aktualizuje při přihlášení do aplikace, status a čas změny se aktualizují v Main aktivitě, konkrétně v metodách *onResume* a *onPause*, tudíž tehdy pokud zobrazíme, nebo skryjeme aplikaci. Díky tomu může uživatel sledovat stav aktivity přítele v detailu konverzace.

Samotné zprávy se načtou z vlastní a ze schránky přítele, poté dojde ke sloučení podle časových razítek a aplikace je zobrazí v seznamu, který má upravený layout tak aby se zprávy od přítele zobrazili vlevo šedě a zprávy uživatele vpravo modře.

Při kliknutí na zprávu se zobrazí detaily a to datum, který je převáděn z časového razítka do formátu podle stáří zprávy a také status značící přečtení zprávy. Načítání probíhá stylem lazy-loading, protože samotné formátování času záleží na stáří zprávy a k výpočtu používáme aktuální čas, který aplikace získává z Firebase, tímto by vznikala při načítání zbytečně velká komunikace k získání dat, která potřebujeme až na vyžádání.

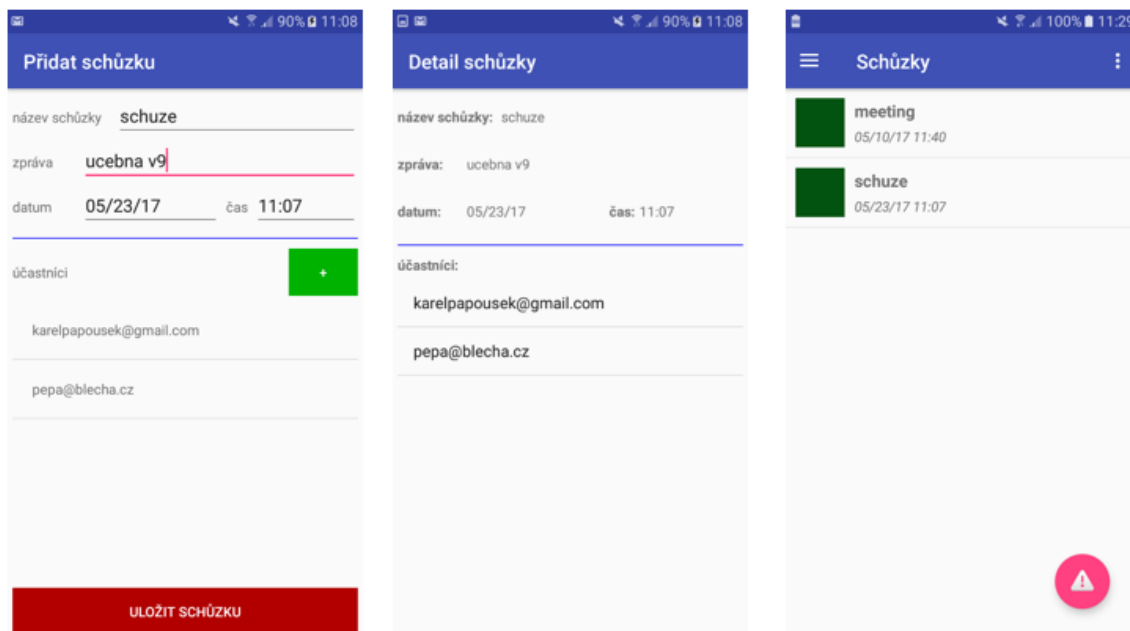
Jediné úskalí, které při realizaci zasílání zpráv nastalo, bylo tehdy, když uživatel zahájí konverzaci s jiným uživatelem, tj. odešle mu první zprávu. V tu chvíli nastane problém a to ten, že uživatel, který odeslal zprávu, ji neuvidí ve vlastním seznamu konverzací. Při načítání seznamu konverzací potřebujeme získat seznam uživatelů, se kterými probíhala komunikace. Při samotném vytváření seznamu uživatelů se vychází z vlastní schránky uživatele, která obsahuje přijaté zprávy a následně se dohledává ve schránkách odesílatelů. Tím pádem pokud odešleme první zprávu, tak je pouze u adresáta, ale odesílatel nemá ve schránce nic. Proto bylo nutné zavést speciální zprávu, která se po odeslání první zprávy vytvoří uživateli záznam ve vlastní schránce. Zpráva má časové razítko s hodnotou -1 a obsahuje typ systém a hodnotu *start\_conversation*. Tímto dosáhneme viditelnosti při načítání seznamu konverzací odesílatele první zprávy. Výhodou tohoto řešení ukládání zpráv na schránky pro příjem je, že sledujeme pouze schránku uživatele (nízká režie) a pak cíleně dohledáváme schránky protějšků a další výhoda spočívá v neredundanci zpráv (úspora dat).



Obrázek 21: Detail konverzace

## 9.6 Schůzky

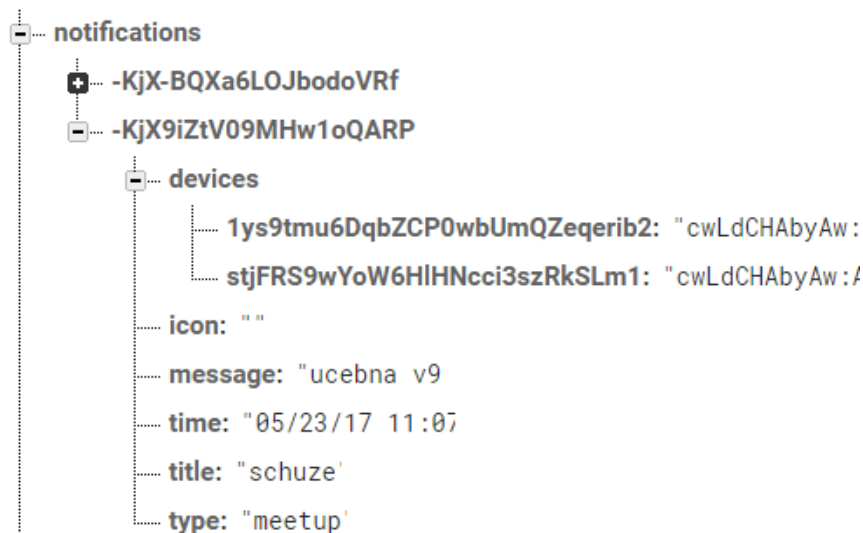
Modul schůzek se povedlo částečně rozpracovat, v mobilní aplikaci jsou vytvořeny aktivity pro zobrazení všech naplánovaných schůzek, detail schůzky a vytvoření nové schůzky.



Obrázek 22: Obrazovky modulu schůzky

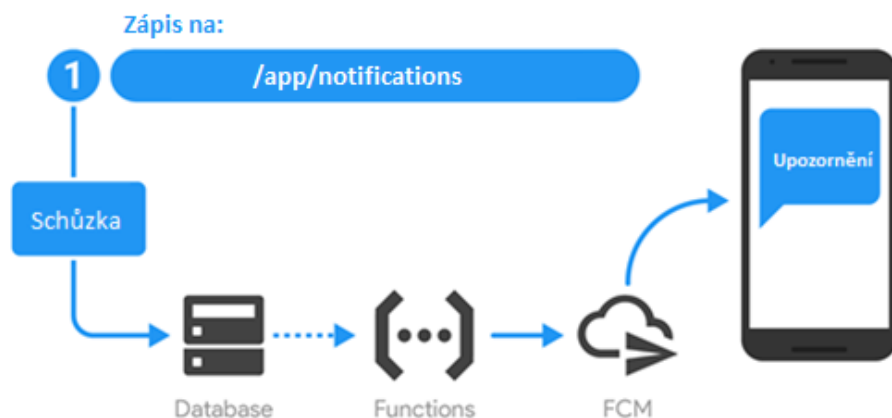
Při vytvoření schůzky se vytvoří záznam v databázi v sekci *notification*, s typem *meetup*, tak abychom v případě rozšíření měli rozlišené typy notifikací. Dále záznam obsahuje položku *devices*, která sdružuje uživatelské id a klíč zařízení spárovaný s daným uživatelem. Díky tomu můžeme vytvořit notifikaci, notifikace se totiž směřují na id zařízení.





Obrázek 23: Ukázka schůzky v databázi

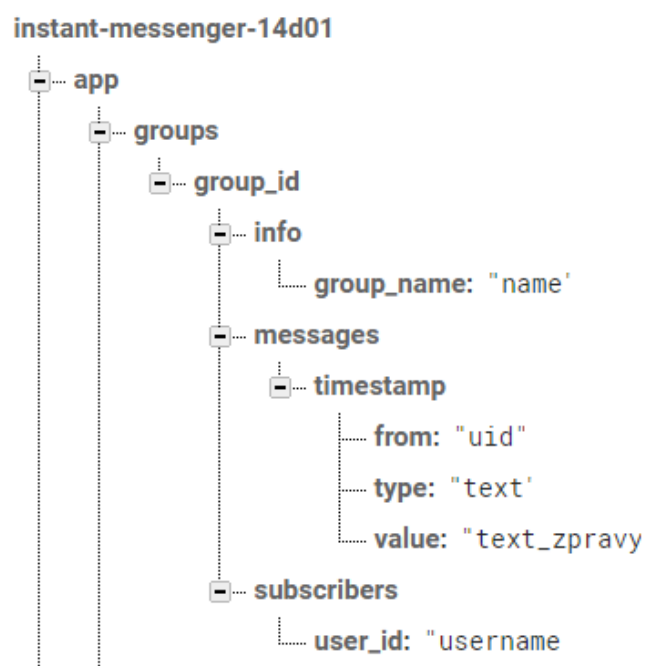
Dále je třeba nad databází nasadit trigger, který by při přidání záznamu vytvořil v systému notifikaci. Dnes je již možné využít pro toto službu Firebase Functions a pomocí skriptu v Node.JS tvořit notifikace, místo nutnosti vytvoření FCM serveru. V aplikaci jsem nevyužil Functions, jelikož neovládám Node.JS a navíc v současné době není kompletně zadokumentovaná funkčnost co se týká automatické tvorby notifikací. Na následujícím obrázku je schematicky naznačena možná realizace upozorňování na schůzky, případně analogicky na broadcast zprávy, ke kterým by bylo nutné vytvořit pomocí Functions REST API, které by bylo vystaveno na veřejných adresách Firebase.



Obrázek 24: Návrh realizace zasílání notifikací

## 9.7 Skupinové konverzace

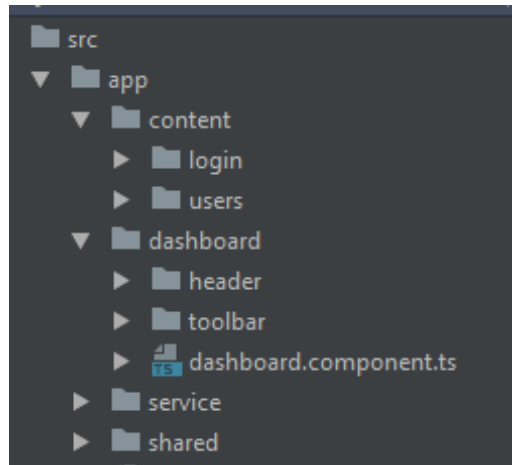
Skupinové konverzace by se realizovali podobně jako zasílání zpráv mezi dvěma uživateli, akorát s tím rozdílem, že by se zprávy uchovávali v jedné položce. Skupina bude uložena pod *group\_id*, které generuje samotné Firebase, tak aby nedošlo ke kolizi. Skupina obsahuje položku *info* uvnitř které jsou informace o skupině, jako je například název, popis. Dále obsahuje položku *messages*, která uchovává všechny zprávy, samotná struktura zprávy by byla analogická, stejně jako u klasické zprávy, akorát s tím že by neobsahovala položku *read* a navíc by přibyla položka *from*, která by jednoznačně identifikovala odesílatele zprávy pomocí jeho id. Poslední položkou skupiny je *subscribers*, která obsahuje účastníky skupinové konverzace.



Obrázek 25: Návrh schématu skupinové konverzace

## 10 Realizace webové administrace

Webová administrace slouží ke správě uživatelů, ale její funkčnost se může rozšířit například na správu databáze, tvorbu skupin, plánování událostí. Momentálně je umožněno vytvořit nového uživatele. Samotná webová administrace je vytvořena pomocí JavaScriptového frameworku Angular2, který se primárně hodí pro tvorbu single-page aplikací.



Projekt je strukturován dle obrázku výše, ve složce *content* se nacházejí jednotlivé obsahy sekcí a v jejich podsložkách komponenty, složka *dashboard* obsahuje komponenty stránky, *service* obsahuje služby např. pro komunikace s Firebase a *shared* pro modely entit.

### 10.1 Přihlašování


Přihlašování je realizováno proti Firebase Authentication, ale s tím že v databázi uchováváme o uživateli navíc informaci, zda-li patří mezi administrátory, což samotný Firebase bohužel neumožňuje, proto je nutné po klasickém ověření proti Firebase, ověřit zda se uživatel nachází mezi administrátory. Informace o administrátorských účtech jsou uloženy v sekci databáze *server/administrators* pod Firebase uid uživatele.

```

signinUser(user: User) {
  firebase.auth().signInWithEmailAndPassword(user.username, user.password)
    .then(userInfo => {
      firebase.database().ref('/server/administrators/' + userInfo.uid).once('value')
        .then(snapshot => {
          if (snapshot.val() != null) {
            this.isAdmin = true;
            this.router.navigate(['/users']);
          } else {
            this.isAdmin = false;
            this.logout();
          }
        })
    })
    .catch(error => {
      var errorCode = error.code;
      var errorMessage = error.message;
    })
});
}

```

Obrázek 26: Ukázka přihlašování s ověřením



Obrázek 27: Přihlašovací obrazovka webové administrace

## 10.2 Přidání uživatele

Vytváření uživatele je realizováno pomocí formuláře, do kterého vyplníme uživatelské jméno ve tvaru obvyklém pro email, a dále jsou tu také položky pro zadání hesla. Při vytváření uživatelského účtu musíme vytvořit účet v modulu Authentication a po vytvoření účtu, vytvoří záznam v databázi (*app/users*)

```

createNewUser(user: User) {
  firebase.auth().createUserWithEmailAndPassword(user.username, user.password)
    .then(userInfo => {
      firebase.database().ref('app/users/' + userInfo.uid).set({
        friends: {id_firend: "id_firend"},
        info: {username: user.username}
      });
      console.log('user created!');
    })
    .catch(function (error) {
      console.log(error);
      var errorCode = error.code;
      var errorMessage = error.message;
    });
}

```

Obrázek 28: Ukázka vytvoření uživatele a záznamu v databázi



The screenshot shows the VSPJ IM web application interface. At the top, there is a logo with the letters 'V', 'Š', 'P' and a small 'J' below them. The title 'VSPJ IM' is displayed. Below the title, there is a navigation bar with links: 'Uživatelé', 'Schůzky', 'Rozhnutí', and 'Nastavení'. A red 'Odhlásit se' button is on the right. The main content area is titled 'Přidat uživatele' and contains a form with three input fields: 'uživatel', 'heslo', and 'potvrzení hesla'. A green 'Přidat uživatele' button is at the bottom right of the form.

Obrázek 29: Formulář pro vytvoření uživatele

### 10.3 Nasazení na Firebase

Webová aplikace je nasazena přímo na Firebase hostingu, pro nahrání jsem použil nástroj Firebase-cli, který umožní nahrání z konzole po zadání klíče a spárování s daným projektem. Samotná aplikace je dostupná z adresy projektu, případně ji můžeme přesměrovat na jinou doménu. Mezi výhody patří například možnost deploy managementu.

## 11 Závěr

Aplikaci se podařilo dle vytyčených cílů realizovat, případně teoreticky navrhnout jejich tvorbu. Aplikaci z hlediska funkčnosti nelze porovnávat s komerčními aplikacemi, jelikož rozsah nelze pokrýt individuálním vývojem. Výhodou aplikace je otevřený kód a možnost libovolné customizace, což komerční systémy poskytují pouze v omezené míře a za poplatek. Aplikace se hodí pro studijní účely, případně jako základní skeleton pro specifické úpravy přímo na míru dané organizaci.

Podrobný úvodní návrh usnadnil samotnou realizaci systému, rozdělil ji do fází, které na sebe navazovali. Při samotné realizaci jsem narazil na několik problémů, které jsem nemohl v úvodní studii odhadnout např. při získávání času ze serveru, avšak veškeré nedostatky Firebase, které se vyskytli, bylo možné lehce eliminovat. Mezi jednu z nevýhod Firebase spatřuji to, že veškerý serverový kód je prováděn na klientském zařízení, což v případě některých typů aplikací může být zásadní bezpečnostní riziko. V době kdy jsem začínal navrhovat systém, samotný Firebase, oproti jiným backendless providerům neposkytoval serverové skripty a bylo nutné vytvořit extra server navíc, což ztrácelo smysl myšlenky backendless. Avšak situace se rapidně změnila a Firebase začal nabízet funkci Cloud Functions, která umožňuje psaní skriptů na serveru a tím umožnit zabezpečené spouštění klíčových funkcí aplikace přímo na serveru. Původní návrh s tímto vůbec nepočítal, a proto některé funkce aplikace jsou složitěji realizované např. tvorba nového uživatele.

## 12 Seznam použité literatury

1. KADLEC, Jiří. *REST a webové služby v jazyce Java*. Brno, 2010.
2. SOMMERVILLE, Ian. *Softwarové inženýrství*. Brno: Computer Press, 2013. ISBN 978-80-251-3826-7.
3. NOVÁ, Jana. *Využití webových služeb a práce s nimi*. Plzeň, 2013.
4. Firebase guide. *Firebase* [online]. [cit. 2017-01-16]. Dostupné z: <https://firebase.google.com/docs>
5. WEISS, Petr a Marek RYCHLÝ. *ARCHITEKTURA ORIENTOVANÁ NA SLUŽBY, NÁVRH ORIENTOVANÝ NA SLUŽBY, WEBOVÉ SLUŽBY*. Brno, 2007.
6. EELES, Peter a Peter CRIPPS. *Architektura softwaru*. Brno: Computer Press, 2011. ISBN 978-80-251-3036-0.
7. Backend as a service comparsion. In: *SlideShare* [online]. [cit. 2017-01-16]. Dostupné z: <http://www.slideshare.net/serhiysnizhny/backend-as-a-service-comparison>
8. LACKO, Ľuboslav. *Vývoj aplikací pro Android*. Brno: Computer Press, 2015. ISBN 9788025143476.
9. ALLEN, Grant. *Android 4: průvodce programováním mobilních aplikací*. Brno: Computer Press, 2013. ISBN 9788025137826.



## 13 Seznam obrázků

Obrázek 1: Ukázka aplikace Messenger .....	11
Obrázek 2: Ukázka aplikace WhatsApp .....	11
Obrázek 3: Ukázka aplikace Skype .....	12
Obrázek 4: Ukázka aplikace Slack .....	13
Obrázek 5: Ukázka komunikace WhatsApp pomocí XMPP .....	15
Obrázek 6: Návrh vlastního systému (původní Firebase) .....	17
Obrázek 7: Návrh vlastního systému (aktualizovaný Firebase) .....	17
Obrázek 8: Schéma FCM .....	22
Obrázek 9: Ukázky využití Functions .....	22
Obrázek 10: Schéma aktivit a fragmentů .....	28
Obrázek 11: schéma databáze .....	30
Obrázek 12: Ukázka konzole Firebase .....	31
Obrázek 13: Ukázka členění balíčků .....	32
Obrázek 14: Ukázka kódu listeneru .....	33
Obrázek 15: Layout přihlášení do aplikace .....	33
Obrázek 16: Menu aplikace .....	35
Obrázek 17: Záznam uživatelů v databázi .....	35
Obrázek 18: Rozložení obrazovek .....	36
Obrázek 19: Struktura ukládání zpráv .....	37
Obrázek 20: Seznam konverzací .....	38
Obrázek 21: Detail konverzace .....	39
Obrázek 22: Obrazovky modulu schůzky .....	40
	49

Obrázek 23: Ukázka schůzky v databázi .....	41
Obrázek 24: Návrh realizace zasílání notifikací .....	42
Obrázek 25: Návrh schématu skupinové konverzace .....	43
Obrázek 26: Ukázka přihlašování s ověřením .....	45
Obrázek 27: Přihlašovací obrazovka webové administrace .....	45
Obrázek 28: Ukázka vytvoření uživatele a záznamu v databázi .....	46
Obrázek 29: Formulář pro vytvoření uživatele.....	46

## **14 Seznam použitých zkratek**

IM – Instant Messaging

JID – Jabber Identifier

mBaaS – Mobile Backend as a Service

SMS – Short Message Service

VOIP – Voice over Internet Protocol

VPN – Virtual Private Network

XMPP – Extensible Messaging and Presence Protocol

## **Přílohy**

### **15 Obsah přiloženého CD**

Na přiloženém CD se v kořenovém adresáři nachází tato bakalářská práce ve formátu *bakalarska\_prace.pdf* s jednoduchým návodem *navod.txt* pro obsluhu programu.