Mendelova univerzita v Brně
Provozně ekonomická fakulta

Mobilní aplikace pro majitele psů

Bakalářská práce

Vedoucí práce:
doc. Ing. Oldřich Trenz, Ph.D.

Petra Cendelínová

Brno 2017

Tímto bych chtěla poděkovat doc. Ing. Oldřichovi Trenzovi, Ph.D. za vedení práce a také Ing. Janu Kolomazníkovi, Ph.D. za cenné rady při konzultacích. Také bych chtěla ocenit rady poskytnuté prostřednictvím konzultační služby jakpsati@pef.mendelu.cz. Dále bych chtěla poděkovat rodině za podporu při studiu a dokončování práce. Poděkování také patří přátelům, kteří mi pomáhali s testováním aplikace.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Mobilní aplikace pro majitele psů**
vypracovala samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*
Jsem si vědoma, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.
Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

2. ledna 2017 .....

Abstract

Cendelínová, P. Mobile application for dog owners. Bachelor thesis. Mendel university in Brno, 2017.

This thesis deals with the design and development of a mobile application for the Android operating system. The main objectives of the application is to provide useful information for the dog owners, to warn about important dates. A brief overview about the Android operating system and methods of creating applications in this system are described in the thesis.

Keywords: Android, Cloud computing, Firebase, Material Design, MBaaS, UML

Abstrakt

Cendelínová, P. Mobilní aplikace pro majitele psů. Bakalářská práce. Mendelova univerzita v Brně, 2017.

Práce se zabývá návrhem a vývojem mobilní aplikace pro operační systém Android. Hlavním cílem aplikace je poskytovat potřebné informace majitelům psů, upozorňovat na důležité termíny. V práci je popsán stručný přehled o OS Android, cloudovém řešení a způsob vývoje v něm.

Klíčová slova: Android, Cloud computing, Firebase, Material Design, MBaaS, UML

Obsah

Seznam zkratek

AOT Ahead of time

API Application Programming Interface

ART Android Runtime

BaaS Backend as a Service

DEX Dalvik Excutable

dp Density-independent pixels

dpi Dots per inch

DVM Dalvik Virtual Machine

FAB Floating Action Button

HTTP HyperText Transfer Protocol

IaaS Infrastructure as a Service

JIT Just In Time

JSON Java Script Object Notation

LIFO Last In First Out

MB Megabajt

obr. obrazovky

OS Operační systém

PaaS Platform as a Service

PDA Personal Digital Assistant

SaaS Software as a Service

SMS Systém krátkých zpráv

SQL Structured Query Language

UI User Interface

URL Uniform Resource Locator

Seznam obrázků

1 Úvod a cíl práce

1.1 Úvod

Elektronika včetně mobilních telefonů se stala nedílnou součástí našich životů. Mnoho z nás vlastní tzv. chytré telefony, které kromě základních funkcí jako volání a psání SMS zpráv, disponují řadou nadstandardních vymožeností. Není tedy velkým překvapením, že je chytrý mobilní telefon důležitým pomocníkem při každodenních situacích například připomenutí důležitého termínu, propojení s e-mailovou schránkou či navigace k hledanému místu.

Díky rozvoji mobilních zařízení došlo k rozmachu v oblasti vývoje mobilních aplikací. K důležitým předpokladům pro používání mobilních aplikací patří operační systém. Na současném trhu existuje řada různých typů mobilních aplikací od výukových až po antivirové programy. Jak již bylo řečeno, aplikace pro mobilní zařízení, zasahují široké spektrum zájmů nejen v soukromém, ale i ve firemním sektoru. V současné době si je spousta firem vědoma velkého vlivu mobilních aplikací a z tohoto důvodu nabízí své produkty či služby jejich prostřednictvím. Navzdory vysokému množství aplikací, toto odvětví informatiky bude mít stále vysoký potenciál zapříčiněním obtížné nahraditelnosti mobilních zařízení v blízké budoucnosti.

1.2 Cíl práce

Cílem této práce je vytvořit mobilní aplikaci pro majitele psů dostupnou pro platformu Android. Aplikace poslouží majitelům psů, kteří si chtějí usnadnit a zdokumentovat společný život se svým psem. Aplikace jim umožní získat potřebné informace o nejbližším veterináři, chovatelských potřebách, ale také o různých psích plemenech. Uživatel bude moci využívat aplikaci k připomínaní důležitých termínů, jako je návštěva veterináře či podání léku psovi. Mimo jiné majitel psa může přizvat další nadšené pejskaře ke společnému věnčení. Celá aplikace bude pojata jako sociální síť pejskařů, kde mohou sdílet své zkušenosti, radosti a zážitky strávené se svými psy.

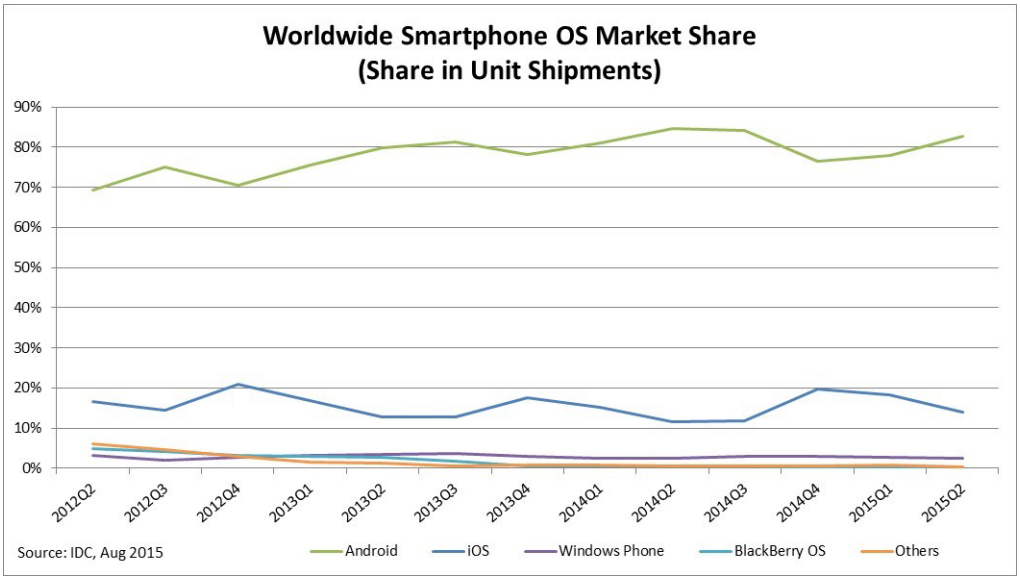
2 Literární přehled

2.1 Operační systém Android

Android je rozsáhlý operační systém vytvořený společností Google. V současné době je vyvíjen organizací Open Handset Alliance, mezi něž patří mimo jiné „velcí hráči“ mobilní branže jako Google, HTC, Samsung a další. Android je založený na open source platformě, což v překladu znamená otevřený zdrojový kód. Pod pojmem „otevřený kód“ si lze představit snadnou dostupnost a to jak technickou, tak licenční. Jinými slovy, uživatel může využívat či upravovat zdrojový kód dle své libosti.

Základem OS je Linuxové jádro, které zajišťuje zabezpečení systému jako celku. Mezi jeho funkce patří například správa paměti a procesů. Následně aplikace k těmto funkcím jádra přistupují pomocí API (Ujbányai, 2012, s. 13).

Díky podpoře více platforem může OS Android fungovat v zařízení různých firem. I z tohoto důvodu Android patří mezi nejpopulárnější operační systémy. Jak naznačuje obrázek 1 Android dlouhodobě zaujímá největší podíl na celkovém trhu mobilních OS. Ve druhém čtvrletí roku 2015 používalo 82.8 % uživatelů právě Android. V současné době je vyvíjen především pro chytré telefony, tablety a PDA.



Obrázek 1: Porovnání podílů jednotlivých OS na trhu
(Převzato z IDC, 2015)

2.1.1 Historie Androidu

Android patří mezi jeden z nejmladších operačních systémů. V roce 2003 byla založena firma Android Inc. zabývající se vývojem aplikací pro mobilní zařízení. O dva roky později se již tato společnost stala stoprocentní dceřinou společností Google. Důležitý zlom nastal 5. listopadu roku 2007, kdy Andy Rubin a jeho tým oficiálně představili operační systém založený na Linuxovém jádře. V tento stejný den byla také založena organizace Open Handset Alliance, která se stará o vývoj Androidu dodnes (Ujbányai, 2012, s. 14).

2.1.2 Verze operačního systému Android

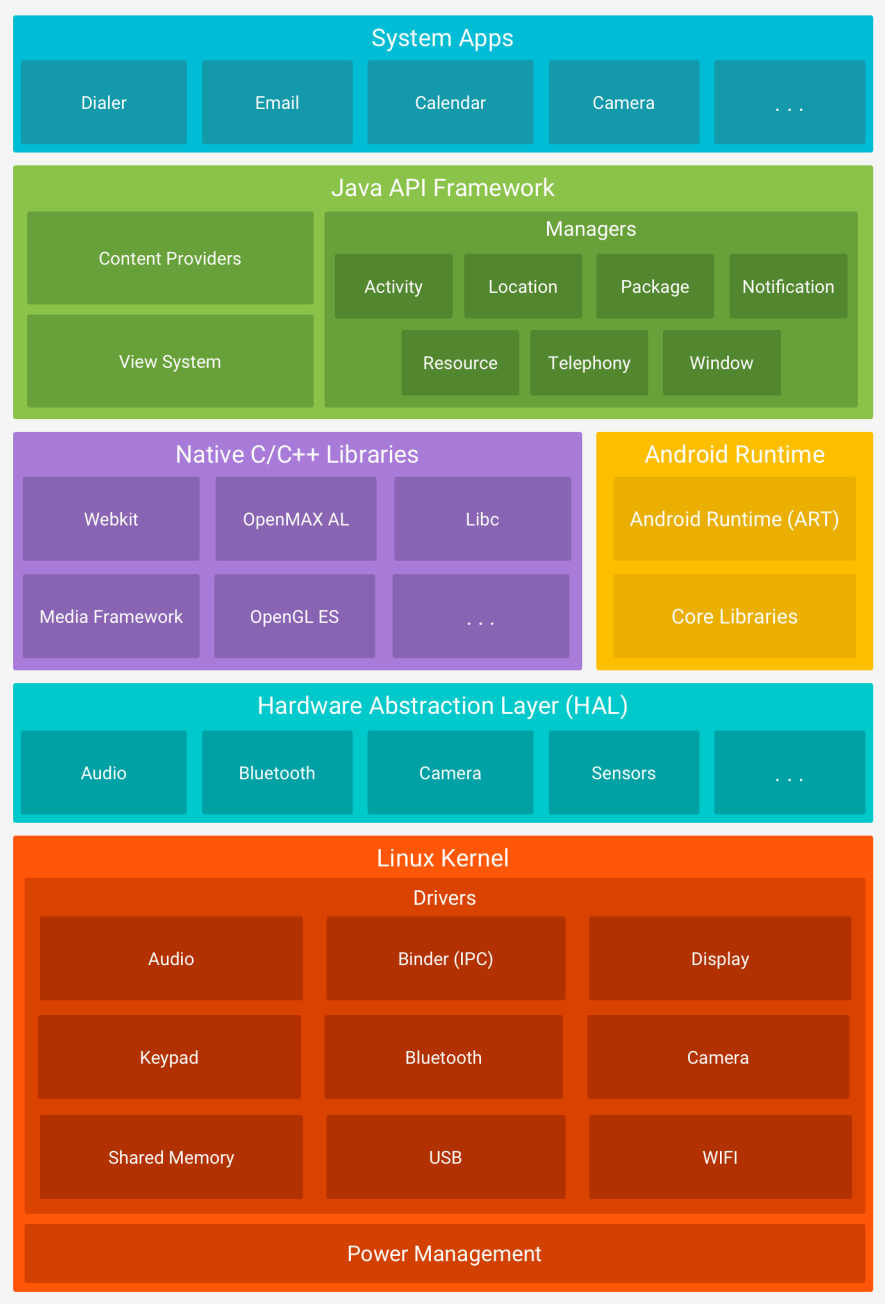
Od vzniku první verze 1.0 Android prošel řadou změn, oprav a vylepšení. Tabulka 1 zobrazuje vývoj verzí operačního systému Android od verze 2.2 až po nejnovější verzi 7.0 s názvem Nougat. Z uvedené tabulky lze usoudit, že nejrozšířenější verzí je verze zvaná Marshmallow s API 23. Zajímavostí je pojmenování jednotlivých verzí, přičemž každá nová verze má svůj přívlastek dle sladkostí.

Tabulka 1: Přehled verzí OS Android k 5. 12. 2016 (Převzato z Android Developers, 2016c)

Číslo verze	**Název**	**API**	**Distribuce**
2.2	Froyo	8	0.1 %
2.3.3–2.3.7	Gingerbread	10	1.2 %
4.0.3–4.0.4	Ice Cream Sandwich	15	1.2 %
4.1.x	JellyBean	16	4.5 %
4.2.x		17	6.4 %
4.3		18	1.9 %
4.4	Kitkat	19	24.0 %
5.0	Lollipop	21	10.8 %
5.1		22	23.2 %
6.0	Marshmallow	23	26.3 %
7.0	Nougat	24	0.4 %

2.1.3 Architektura OS Android

Architektura OS Androidu je založena na spolupráci šesti vrstev, přičemž každá z nich má na starosti určitou funkcionalitu. Obrázek 2 představuje fungování jednotlivých vrstev.



Obrázek 2: Architektura OS Android
(Převzato z Android Developers, 2016b)

Linux Kernel

Jak uvádí Lacko (Lacko, 2015, s. 59) ve své knize nejnižší vrstvou představuje upravené Linuxové jádro. Zabezpečuje správu paměti, správu procesů a základní síťové operace.

Hardward Abstraction Layer

Tato vrstva poskytuje rozhraní mezi hardwarovou vrstvou a vyššími softwarovými vrstvami. Pro každou hardwarovou komponentu je vytvořen modul a ten je v případě použití načten (Android Developers, 2016a).

Native C/C++ Libraries

Následující vrstva Libraries (knihovny) poskytuje řadu nativních knihoven napsaných v jazyce C a C++. K dispozici jsou například knihovny pro práci s webovým obsahem či mapou (Ujbányai, 2012, s. 18).

Android Runtime

Jak popisuje Lacko další vrstvu tvoří virtuální stroj Dalvik Virtual Machine DVM a základní Java knihovny. DVM je jistou analogií JVM na klasických počítačích, ale bere v potaz možnosti napájení a menší paměť mobilních zařízení (Lacko, 2015, s. 60). Dalvik je založen na JIT (Just in time) komplikaci. Zdrojový kód se překládá do Dalvik bytecode (soubory v příponou DEX) za běhu aplikace. Tyto soubory jsou dále zpracovány tímto virtuálním strojem.

V případě zařízení s verzí Androidu 5.0 a výše je DVM nahrazen novým virtuálním strojem Android Runtime (ART). V roce 2013 byl poprvé představen ART ve verzi 4.4 Kitkat. Jednalo se pouze o testovací účely, ale o rok později zcela nahradil původní virtuální stroj DVM. ART přináší Ahead-of-time kompilaci (AOT). K přeložení zdrojového kódu dojde pouze jednou, a to již při instalaci aplikace. Výhodou AOT je snížení nároků na baterii mobilního zařízení. Nevýhodou je prodloužení instalace vyžadující vyšší paměťové nároky (Vitas, 2013).

Java API Framework

Z pohledu vývojáře je nejdůležitější vrstvou Application Framework. Umožňuje vývojářům přistoupit k službám systému a komponentám, které mohou dále využít ve svých aplikacích. K dispozici je například Content Providers (pro přístup k datům jiných aplikací). Další službou je Notification Manager sloužící k zobrazení upozornění ve stavové liště. Prostřednictvím služby Activity Manager lze spravovat životní cyklus aktivity (Android Developers, 2016a).

System Apps

Android poskytuje základní systémové aplikace jako emailový klient, kontakty a internetový prohlížeč. Funkcí těchto aplikací může vývojář využít ve svých vlastních aplikacích (Android Developers, 2016a).

2.1.4 Komponenty Android aplikace

Dle Ujbányai (Ujbányai, 2012, s. 37) existují tři typy aplikací. První z nich může běžet na popředí, příkladem tohoto typu mohou být hry na mobilních zařízení. Druhým typem se myslí aplikace pracující na pozadí, jedná se například o aplikace sledující telefonní hovory. Posledním typem jsou aplikace s přerušovanou činností, většinu času pracují na pozadí, ale zároveň komunikují s uživatelem prostřednictvím upozornění na určité události. Každá aplikace se skládá z komponent. Jak dokazuje Ujbányai (Ujbányai, 2012, s. 29) se jedná o následující komponenty:

- **Aktivity (Activity)**
  Za základní pilíře uživatelského rozhraní se považují aktivity, které představují jednu obrazovku aplikace. Většina aplikací se skládá z více aktivit, které jsou mezi sebou provázány. Při startu je zobrazena aktivita, která byla určena jako hlavní. Jedna aktivita může provolávat jinou, pokud k tomuto dojde, předchozí aktivita se pozastaví a umístí do zásobníku. Zásobník funguje na principu LIFO, to v praxi znamená, že poslední vložená aktivita bude obnovena jako první.
- **Služby (Services)**
  Služby na rozdíl od aktivit jsou navrženy, aby prováděly dlouhotrvající operace na pozadí bez interakce s grafickým uživatelským rozhraním.
- **Poskytovatelé obsahu (Content Providers)**
  Tato komponenta má na starosti práci s daty a jejich zpřístupnění pro všechny aplikace z důvodu neexistence jednoho univerziálního uložiště, ke kterému by měly všechny balíčky systému Android přístup.
- **Záměry (Intents)**
  Záměr lze chápat jako operaci, která se musí vykonat. Může se například jednat o zobrazení webové stránky či vytočení čísla pro telefonní hovor. Záměry se rozdělují do dvou skupin, a to implicitní a explicitní. V případě implicitního záměru není definováno, která aplikace akci provede. Uživatel tedy vybírá aplikaci, která záměr provede. Explicitní záměr již dopředu definuje aplikaci, která akci provede. Výběr provádí samotná platforma, která udržuje informaci ve formě klíč-hodnota, kde klíč představuje akci, a hodnota aplikaci, která bude reagovat na danou akci.
- **Přijímače (Broadcast Receivers)**
  Jedná se o komponenty, které slouží k naslouchání oznámení z vnějšku i zevnitř aplikace. Na základě typu oznámení patřičně reaguje, příkladem může být reakce na obdrženou SMS zprávu. Existují dva typy, první je systémový broadcast a druhým typem je vlastní napsaný vývojářem.
- **Oznámení (Notifications)**
  V aplikaci mohou nastat určité situace, o niž by měl být uživatel informován. K tomuto účelu aplikace využívá oznámení, které mohou být zobrazeny trojím

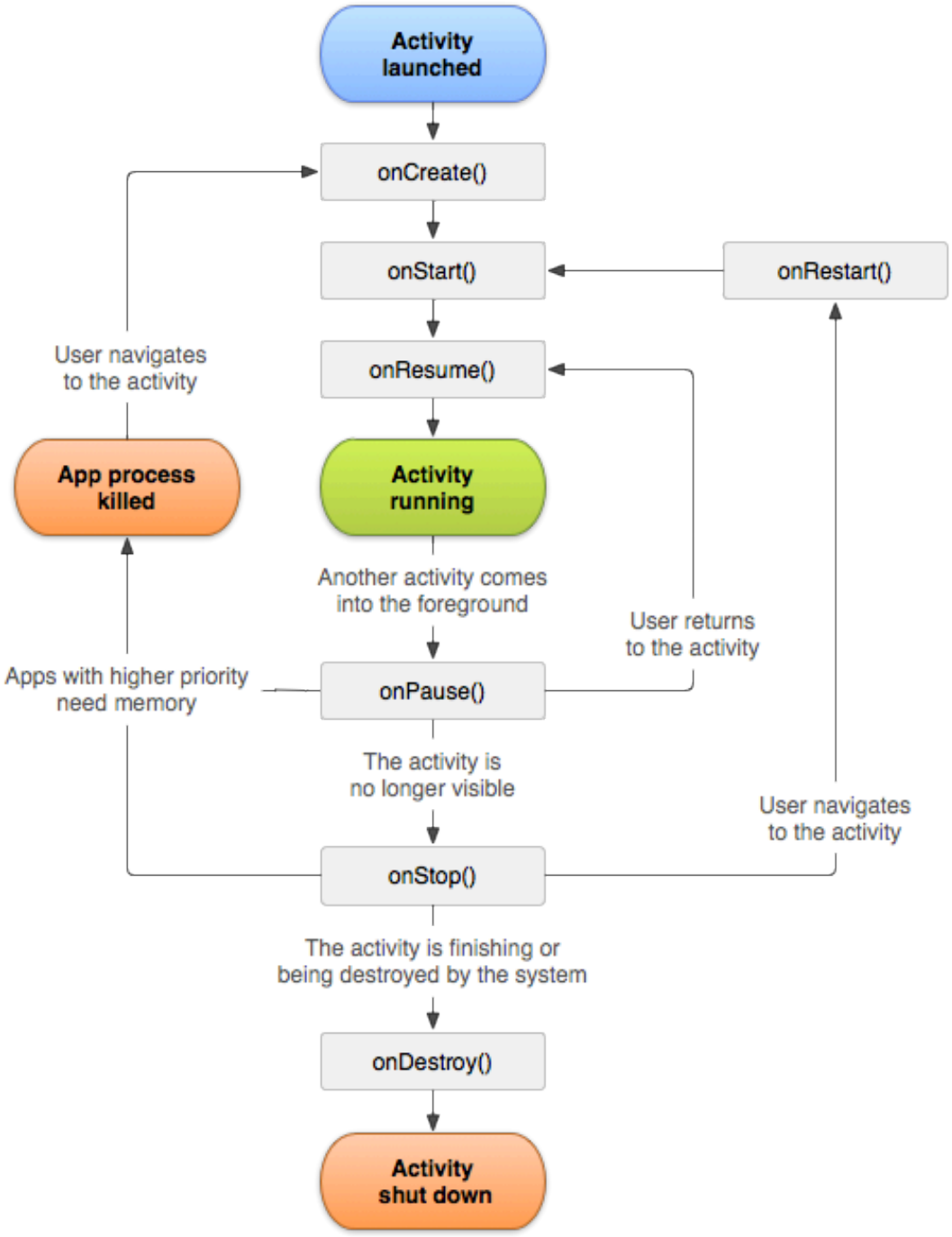
způsobem. První typ zvaný Toast zobrazí sdělení pro uživatele na pracovní ploše okna a následně zmizí. Dalším způsobem oznámení o vzniklé události je ve stavovém řádku v případě, kdy aplikace pracuje v pozadí. Posledním typem je oznámení formou dialogového okna s potřebnou interakcí od uživatele.

2.1.5 Životní cyklus aktivity

Jak již bylo řečeno výše, aktivita představuje jednu obrazovku aplikace s uživatelským rozhraním. Allen vysvětluje (Allen, 2013, s. 241), že aktivita se může nacházet v jednom ze čtyř stavů, a to:

- **Aktivní**: Aktivita byla spuštěna uživatelem, běží v popředí a komunikuje s uživatelem.
- **Pozastavená**: Aktivita je spuštěná, běží, ale část její obrazovky překrývá určité upozornění. Tento stav může například nastat, když zařízení detekuje příchozí hovor a upozorní uživatele na jeho přijetí.
- **Zastavená**: Jedná se o stav aktivity, při němž je aktivita skrytá za jinými aktivitami a vložena do zásobníku. Lze se k ní tedy kdykoliv bezproblémově vrátit.
- **Mrtvá**: V tomto případě se může jednat o aktivitu, která nebyla ještě spuštěna uživatelem nebo byla násilně ukončena.

Jak naznačuje obrázek 3 při přechodu mezi jednotlivými stavy aktivity, jsou volány metody získané děděním z objektu Activity.



Obrázek 3: Životní cyklus aktivity
(Převzato z Android Developers, 2015b)

Metoda onCreate()

Tato metoda je volána při prvním spuštění aktivity. V těle metody se na pozadí konfiguruje vše potřebné k běhu aktivity jako například uživatelské rozhraní, získání dat. V případě, že aktivita již byla spuštěna a následně překryta jinou aktivitou, při jejím dalším spuštění se zavolá metoda onCreate s parametrem Bundle získaný z metody onSaveInstanceState(). Vždy je po ní volána metoda onStart(), uvádí Allen (Allen, 2013, s. 242).

Metoda onStart()

Voláním metody onStart() se stává aktivita viditelnou pro uživatele. Může být následována metodou onResume() při přechodu do popředí či metodou onStop(), kdy se aktivita stává neviditelnou pro uživatele (Android Developers, 2015a).

Metoda onResume()

Tato metoda je volána ve chvíli, kdy aktivita má komunikovat s uživatelem, popisuje Ujbányai (Ujbányai, 2012, s. 41).

Metoda onPause()

Pokud aktivita přestává být na popředí, zavolá se metoda onPause(), typickým příkladem jejího spuštění je přesun uživatele do jiné aktivity například stisknutím tlačítka Home. V této metodě se obvykle řeší uložení aktuálních dat, uvádí Lacko (Lacko, 2015, s. 68).

Metoda onStop()

Metoda onStop() se volá ve chvíli, kdy aktivita je dlouhodoběji neviditelná pro uživatele a tudíž je třeba ji zastavit. Může po ní následovat metoda onRestart(), která obnoví její běh nebo jí ukončí metoda onDestroy(), jak je uvedeno na webových stránkách Android Developers (Android Developers, 2015a).

Metoda onRestart()

Dle Ujbányai (Ujbányai, 2012, s. 41) je metoda onRestart() volána v situaci, kdy aktivita byla zastavena a je potřeba ji obnovit.

Metoda onDestroy()

Na opačném konci životního cyklu je metoda onDestroy(), která může být volána při ukončování aktivity, například při volání metody finish() nebo vynuceným ukončením aktivity (Allen, 2013, s. 242).

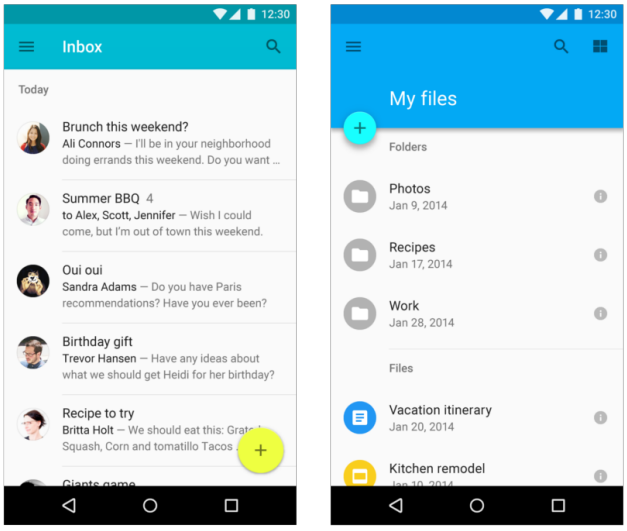
2.2 Android Material Design

Material Design je známý jako designový jazyk se snahou spojit principy dobrého designu s inovací, technologií a vědou napříč všemi zařízeními nezávislými na velikosti či typu (Google, 2016f). Před uveřejněním Material Design neexistovala pravidla sjednocující uživatelské rozhraní. V roce 2014 na konferenci Google I/O tým designérů firmy Google přišel s novým nápadem, který by spojoval uživatelské rozhraní jejich aplikací a poskytl tak lepší a jednotnou interakci s uživatelem (Thornsby, 2016, s. 124).

Základem všeho byla změna pohledu týmu designérů na design jako technickou záležitost. Začali uvažovat o něm jako o existujících objektech v reálném světě s danými fyzickými vlastnostmi. Při jeho návrhu se autoři nechali zejména inspirovat prací s papírem a inkoustem (Bohn, 2014). Z tohoto důvodu Material Design využívá prvky reality a co nejvěrohodněji se je snaží převést do virtuální podoby. Především je kladen důraz na práci se stíny a světly prostřednictvím 3D prostoru. V praxi to znamená, že existují osy x (šířka), y (výška) a z (hloubka). Osa z indikuje efekt přibližování nebo oddalování od uživatele. Významnou roli v návrhu Material Designu také hrají animace, které pomáhají uživateli pochopit, co se právě děje v aplikaci (Fulcher, 2014).

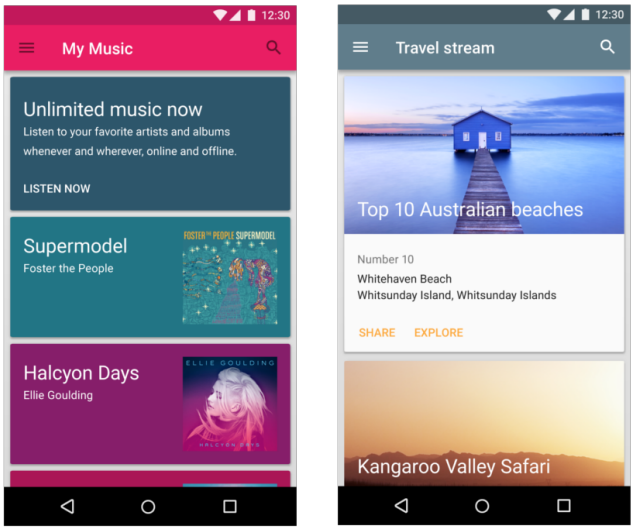
Kromě nového designu stávajících komponent, Material Design přinesl také řadu nových. Za zmínku stojí List, Card, Bottom Sheet, Floating Action Button (FAB).

List patří k jednomu ze způsobů jak zobrazit data stejného typu. Představuje jeden sloupec obsahující stejně velké řádky s textem a obrázkem. Pokud je text delší než 3 řádky, je vhodné využít Card (Google, 2016a).



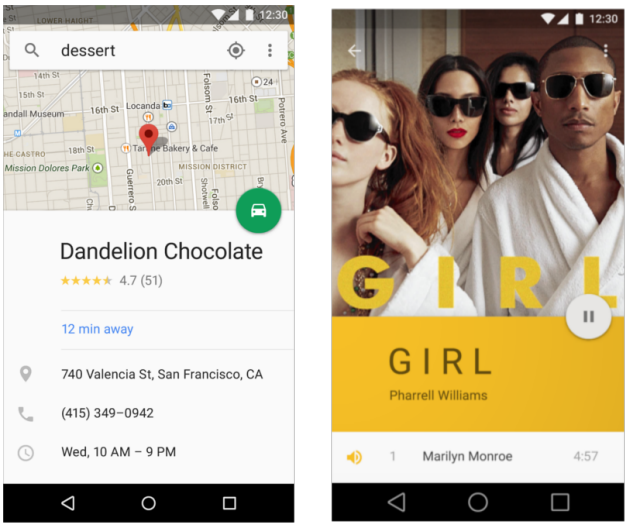
Obrázek 4: Příklad užití List
(Převzato z Google, 2016e)

Card nabízí nový způsob zobrazení různých typů dat. Card se obvykle skládá z nadpisu, obrázku usnadňujícího orientaci uživateli, doplňujícího textu a případně dalších akcí včetně FAB (Thornsby, 2016, s. 138).



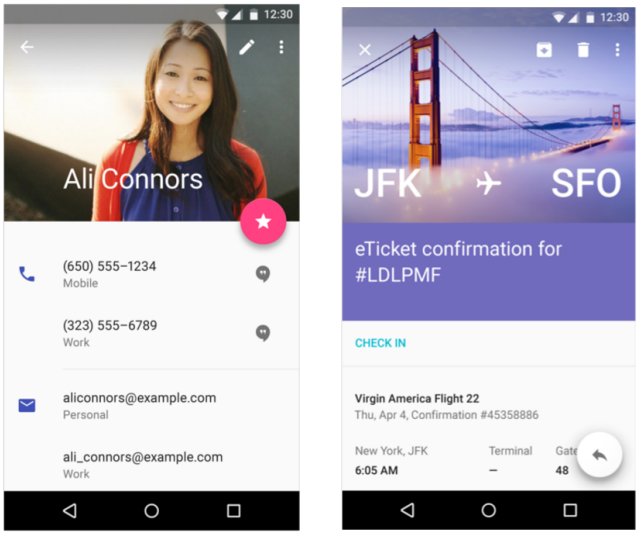
Obrázek 5: Příklad užití Card
(Převzato z Google, 2016c)

Bottom Sheet je komponenta prostřednictvím, které dojde k zobrazení bližších informací dané aktivity. Nachází se v dolní části obrazovky, kde pohybem prstu nahoru lze tuto komponentu aktivovat (Google, 2016a).



Obrázek 6: Příklad užití Bottom Sheet
(Převzato z Google, 2016b)

Floating Action Button (plouvoucí tlačítko) představuje tlačítko ve tvaru kruhu s určitou akcí. Toto tlačítko obvykle bývá umístněno v dolní části obrazovky, pro mobilní zařízení je doporučeno 16 dp[1] od kraje. Mělo by se vztahovat pouze k pozitivním operacím jako je vytvoření, sdílení aj. (Thornsby, 2016, s. 134). Použití plovoucího tlačítka lze vidět na obrázku 7.



Obrázek 7: Příklad užití plovoucího tlačítka
(Převzato z Google, 2016d)

2.3 Cloud computing

Jedná se o poměrně nový způsob využívání softwarových i hardwarových zdrojů poskytovaných formou služeb prostřednictvím internetu. Cloud computing vznikl především z důvodu snížení nákladů a zefektivnění výpočetního výkonu. Funguje na principu Pay-as-you-go. Zákazník platí pouze za to, co doopravdy využije. Další významnou výhodou je možnost rychlého přizpůsobení potřebám a požadavkům zákazníka (Woodford, 2016).

Kromě uvedených výhod se mohou objevit určité nedostatky. Při výběru vhodného poskytovale cloudového řešení je nutné si počínat obezřetně. Data jsou uchovány u poskytovatelů služeb a zákazník nad nimi nemá přímou kontrolu. V určité míře je tedy zákazník závislý na poskytovateli a na jeho důvěryhodnosti. Cloud computing vyžaduje kvalitní a rychlé internetové připojení.

Jak uvádí Cristian Spoiala (Spoiala, 2015) s cloud computing se lze setkat v současné době v následujících podobách kategorizováných dle prostředků poskytovaným uživatelům (tzv. distribuční model):

[1]Jedná o jednotku velikosti UI elementů použitou při vývoji Android aplikaci, která zaručuje jejich stejné zobrazení na různých fyzických zařízení. 1 dp odpovídá 1 px na zařízení s 160 dpi (udává počet px, které se vejde do délky jednoho palce) (Nurik, 2013).

- **Infrastructure as a Service (IaaS)**– v rámci tohoto řešení je uživateli k dispozici hardware, software, síťové zdroje a úložiště nejčastěji formou virtualizace
- **Platform as a Service (PaaS)** – cílem je poskytnout kompletní prostředky podporující celý životní cyklus aplikace, jako významní poskytovatele toho řešení lze zmínit např. IBM Bluemix nebo SalesForce
- **Software as a Service (SaaS)** – jsou nabízeny koncovým uživatelům přes internet. O veškerou údržbu související se softwarem se stará poskytovatel služby.
- **Backend as a Service (BaaS)** – jedná se o nový způsob, jak poskytnout backendovou část aplikace, především je znám v souvislosti s mobilními aplikacemi (dále jen MBaaS).

Další dělení cloud computing je realizováno podle modelu nasazení, který rozděluje cloud computing do čtyř kategorií, podle vlastníka a velikosti:

- **Veřejný cloud** – je dostupný pro veřejné použití se stejnou či podobnou funkcionalitou všem klientům, bývá poskytován zdarma či za poplatek za použití
- **Privátní cloud** – je využíván pouze jednou organizací, což umožňuje větší škálovatelnost a kontrolu
- **Komunitní cloud** – v rámci komunitního cloudu je více organizací se stejnou politikou společně sdílející infrastrukturu, hlavní výhodou je snížení nákladů oproti privátnímu cloudu
- **Hybridní cloud** – jedná se o propojení výše uvedených typů cloudu (Metha, 2012)

2.4 Jazyk UML

2.4.1 Co je UML?

Jazyk UML neboli Unified Modeling Language je modelovací jazyk využívající grafické notace k návrhu a popisu softwarových systémů. Zejména je znám ve spojitosti s objektově orientovanými systémy (Fowler, 2009, s. 23). Díky své univerzálnosti nachází uplatnění v různých oblastech informačních systémů (zdravotnictví, bankovnictví aj.) (Pilone, Pitman, 2005, s. 2).

2.4.2 Historie jazyka UML

Před rokem 1994 existovalo více soupěřících metodik, které nabízely odlišné způsoby jak systém navrhnout a popsat. Bylo tedy obtížné se v jednotlivých přístupech orientovat. Z tohoto důvodu se objevily snahy, jak tyto jednotlivé metodiky sjednotit. Jednou z prvních standardů pokoušejících se o integraci byla metodika Fusion. Její neúspěch zejména způsobilo nezapojení hlavních autorů do její přípravy (Booch, Jasobson a Rumbaugh). Posléze Booch a Rumbaugh se připojili k firmě Rational Corporation, která se v té době zabývala tvorbou jazyka UML. Jejich společným cílem byla standartizace přístupu. Jejich snažení se podařilo naplnit v roce 1997, kdy byl jazyk UML sdružením OMG přijat jako průmyslový standard (Arlow, Neustadt, 2003, s. 5).

2.4.3 Struktura jazyka UML

Dle Arlow a Neustandt (Arlow, Neustadt, 2003, s. 8) strukturu jazyka UML tvoří následující prvky:

- **Předměty**: představují základní prvky modelu, předměty lze dále dělit na:
  - strukturní abstrakce – jedná se o podstatná jména modelu UML, např. třídy, rozhraní, případ užití
  - chování – jsou známa jako slovesa modelu (zpráva, stav)
  - seskupení – jinými slovy balíčky využívané k seskupování souvisejících elementů
  - poznámky – jsou využívány k okomentování daných elementů
- **Relace**: popisují vztahy mezi jednotlivými předměty, v UML existují čtyři typy relací, a to:
  - asociace – asociace vzniká spojením dvou nezávislých předmětů, lze ji zakreslit jednoduchou plnou čárou, směr jednostranné asociace indikuje šipka
  - závislost – tento vztah mezi elementy vznikne pouze v případě, kdy změna jednoho elementu vyvolá změnu druhého předmětu, závislost je znázorněna přerušovanou čárou s jednoduchou šipkou

  - zobecnění – nastává v případě, kdy předmět je specializací jiného předmětu, tato skutečnost se zobrazuje plnou čárou s prázdnou šipkou směřující k zobecněné třídě
  - realizace – vztah vznikající mezi klasifikátory neboli abstraktním vyjádřením předmětu

- **Diagramy**: jedná o vizualizaci systému

Jak Fowler (Fowler, 2009, s. 32) popisuje ve své knize, existuje 13 typů diagramů. Členění spočívá v rozdělení diagramu do dvou hlavních skupin, a to:

- **Diagram struktur**: tato skupina diagramů popisuje předměty a vzájemné relace mezi nimi, příklad může být diagram tříd nebo komponent
- **Diagram chování**: tato kategorie představuje chování jednotlivých předmětů, aby se dosáhlo cíleného chování systému jako celku

3 Metodika řešení

K dosažení stanoveného cíle je třeba postupovat dle následujících kroků:

1. Literární průzkum
   - Seznámit se s OS Android
   - Zjistit si informace o Material Design
   - Osvojit si základy práce s UML modelováním
2. Analýza současného stavu
   - Porovnání a zhodnocení konkurenčních aplikací
   - Výběr vhodného poskytovatele MBaaS
   - Definování funkčních a nefunkčních požadavků aplikace
3. Návrh řešení
   - Vytvoření diagramu případu užití se scénaři
   - Vytvoření sekvenčního diagramu – zaslání notifikací
   - Tvorba diagramu aktivit – přihlášení uživatele
   - Navržení diagramu tříd
   - Vytvoření grafického návrhu jednotlivých obrazovek
4. Sestavení testovacího scénáře
5. Implementace řešení
   - Získání Facebook App ID
   - Získání API key pro použití Google Maps v aplikaci
   - Samotná implementace návrhu v programovacím jazyku Java s užitím IDE Android Studio
6. Testování aplikace
   - zvolení testovacích zařízení
   - otestování aplikace dle vytvořeného testovacího scénáře z kroku 4

4 Analýza současného stavu

Analýza současného stavu se bude skládat ze tří kroků. První krok spočívá v zhodnocení stávající konkurence. Posléze budou vybráni tři poskytovatelé MbaaS a porovnáni dle stanovených kritérií. Po získání informací o současném trhu s mobilními aplikacemi a volbou vhodného poskytovale lze přistoupit k definování funkčních a nefunkčních požadavků na aplikaci.

4.1 Analýza existujících aplikací

Důležitou součástí každého projektu je průzkum trhu. Cílovou skupinu představují majitelé psů vlastnící mobilní telefon s OS Android. Cílem následující podkapitoly je se seznámit s již existujícími řešeními a určit jejich silné a slabé stránky. Ze zjištěných poznatků se poučit, případně se vyvarovat těchto nedostatků. Budou vybráni tři zástupci z českého prostředí a tři zástupci zahraničních aplikací zdarma dostupných na Google Play. Jednotlivé aplikace budou hodnoceny dle stanovených kritérií:

- Hodnocení na Google Play
- Práce se sociálními sítěmi – přihlášení, sdílení, chat
- Možnost zobrazení informací na mapě
- Informace o psích plemenech
- Online x offline
- Uživatelská přívětivost založená na subjektivním hodnocení
- Volba jazyků
- Minimální verze Android
- Velikost aplikace

4.1.1 Zahraniční aplikace

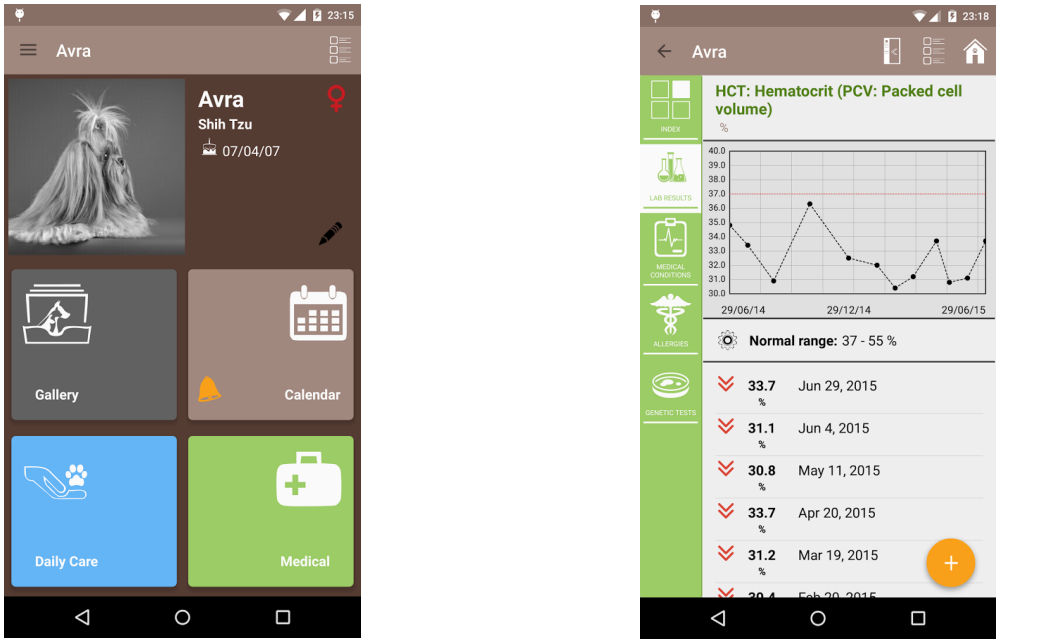
4.1.1.1 11Pets

První zvolenou aplikací je 11Pets. Tato aplikace je určena pro majitele domácích mazlíčků, jak je možné vidět na obrázku 8a. Uživateli je k dispozici řada funkcí od vytváření galerie, spravování upomínek rozdělených do podrobných kategorií až po vytváření historie týkající se zdravotního stavu zvířete (obrázek 8b). Užitečnou funkcionalitou je možnost zálohy dat na cloudu. Aplikace umožnuje přizpůsobení nastavení, k dispozici je 12 jazykových preferencí.

Záporně je hodnocen grafický vzhled aplikace, který se jeví zbytečně složitě a neintuitivně. Dle subjektivního názoru barevná kompozice není vhodná. Splnění jednotlivých kritérií je souhrně demonstrováno v tabulce 2.

Tabulka 2: Shrnutí aplikace 11Pets

Hodnocení na Google Play	4,6 z 590
Práce se sociálními sítěmi	Přihlášení přes Facebook
Informace na mapě	Ne
Online x offline	Online
Vzhled a ovládání	Průměrné
Více jazyků	Ano, 12 jazyků
Minimální verze Android	4.0 a vyšší
Velikost aplikace	4,4 MB



(a) Úvodní obrazovka

(b) Historie zdravotního stavu zvířete

Obrázek 8: Náhled aplikace 11Pets
(Převzato z Google Play, 2016g)

4.1.1.2 Dogalize

Následující zahraniční zástupce je dobrým příkladem komplexní sociální aplikace zaměřené na psy. Kromě funkcí, které bylo možné vypozorovat u aplikace 11Pets, nabízí více možností jak se socializovat s ostatními majiteli psů zejména prostřednictvím chatu a videohovorů. Je možné také přizvat nové uživatele ze svých stávajících kontaktů v mobilním telefonu k používání této aplikace. Zajímavou funkcí je seznamka neboli Dog Dating. Kladně je ohodnocena „zeď příspěvků“, které mohou být ohodnoceny, okomentovány a sdíleny na různých sociálních sítích (obrázek 9a).

Uživatel najde potřebné informace o veterináři, cvičiteli s možností okamžitého kontaktování v podobě zaslání emailu nebo telefonátu.

Jak demonstruje obrázek 9b, ovládání je více intuitivní. Uživatel si je vědom, kde se v aplikaci právě nachází. Dobrým dojmem působí úvodní nápověda při registraci či přihlášení seznamující uživatele o základních funkcích aplikace.

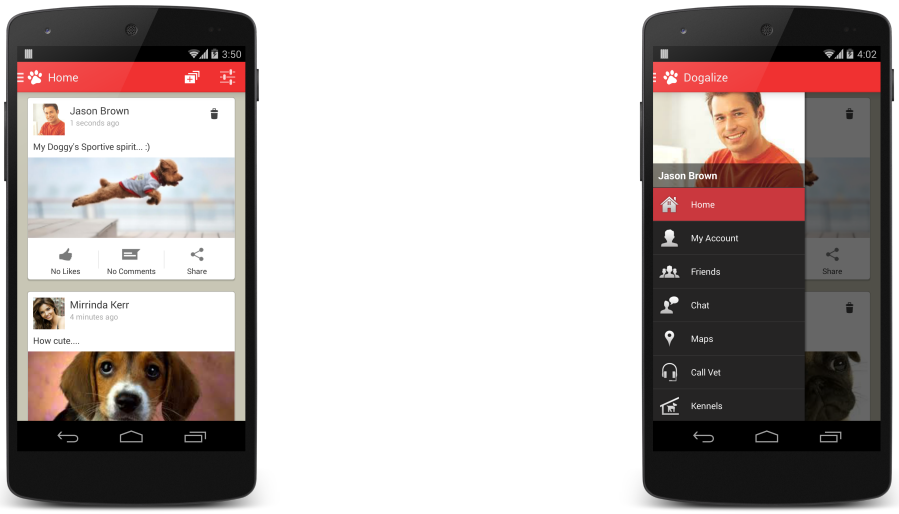
Zajímavým nápadem je možnost sdílení informace o ztrátě psa, kterou mohou vidět i ostatní uživatelé.

Negativně je hodnoceno řešení mapy zobrazující nejbližší veterináře, hotely a další místa. Na testovaném zařízení se tato mapa pouze zobrazila ve webovém prohlížeči, nikoliv přímo v aplikaci. Oproti 11Pets nastavení nenabízí tolik možností k přizpůsobení – chybí volba jazyků. Možnou nevýhodou může být také větší velikost daná komplexností aplikace. Celkově aplikace je hodnocena nadprůměrně.

Tento fakt demonstruje tabulka 3.

Tabulka 3: Shrnutí aplikace Dogalize

Hodnocení na Google Play	4,2 z 2 191
Práce se sociálními sítěmi	Přihlášení přes Facebook, sdílení, chat
Informace na mapě	Ano, pouze ve webovým prohlížeči
Online x offline	Online
Vzhled a ovládání	Nadprůměrné
Více jazyků	Ne
Minimální verze Android	4.2 a vyšší
Velikost aplikace	29 MB



(a) Zeď příspěvků (b) Hlavní navigace aplikace

Obrázek 9: Náhled aplikace Dogalize
(Převzato z Google Play, 2016h)

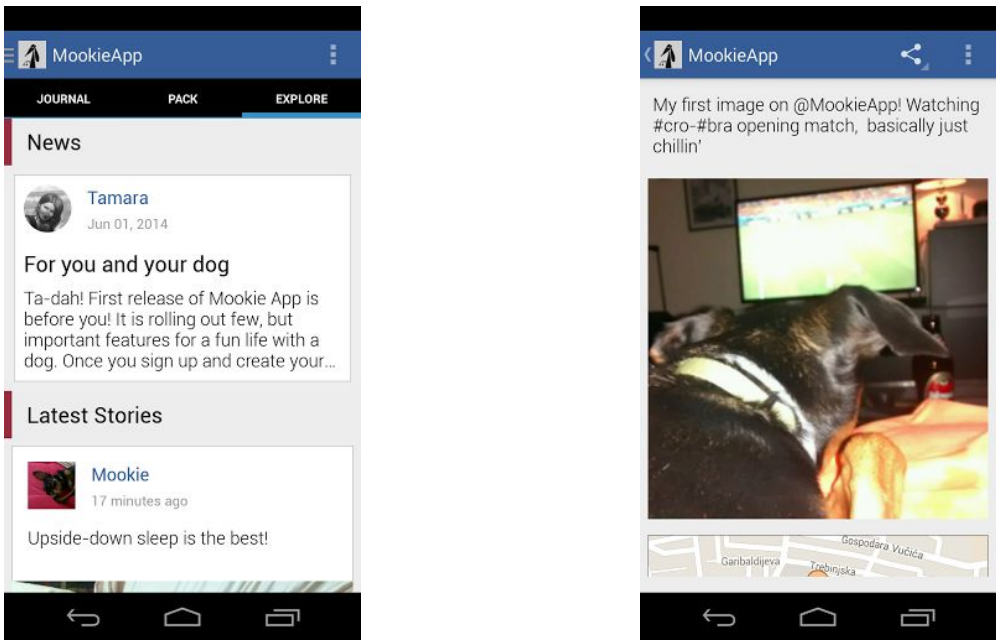
4.1.1.3 MookieApp

MookieApp je sociální aplikací pro majitele psů. Nenabízí tolik funkcí, jak předchozí dvě aplikace (např. vytvoření upomínky). Uživateli je umožněno sdílení příběhů a statusů ostatním uživatelům, které mohou být jimi komentovány a hodnoceny. Opět se v aplikaci nachází zeď příspěvků viditelné na obrázku 10a. Zajímavě je řešeno vyhledávání pomocí hashtagů umístněných v příspěvcích. Kromě sdílení a sledování příspěvků není uživateli umožněno využití dalších funkcí. U sociální aplikace je důležitá komunikace mezi uživateli, bohužel v této aplikaci chybí.

Ovládání a vzhled se pohybují v mezích průměrnosti (obrázek 10b). Nedostatkem je nevhodné umístnění plovoucího tlačítka pro přidání nového příspěvku na každé obrazovce. Vzhledem k omezené funkcionalitě danou zkušební verzí je tato aplikace hodnocena jako horší průměr.

Tabulka 4: Shrnutí aplikace MookieApp

Hodnocení na Google Play	4,6 z 109
Práce se sociálními sítěmi	Sdílení příspěvků na sociálních sítích
Informace na mapě	Ne
Online x offline	Online
Vzhled a ovládání	Průměrné
Více jazyků	Ne
Minimální verze Android	4.1 a vyšší
Velikost aplikace	16 MB



(a) Zeď příspěvků (b) Detail příspěvku

Obrázek 10: Náhled aplikace MookieApp
(Převzato z Google Play, 2016ch)

4.1.2 České aplikace

4.1.2.1 Psí detektiv

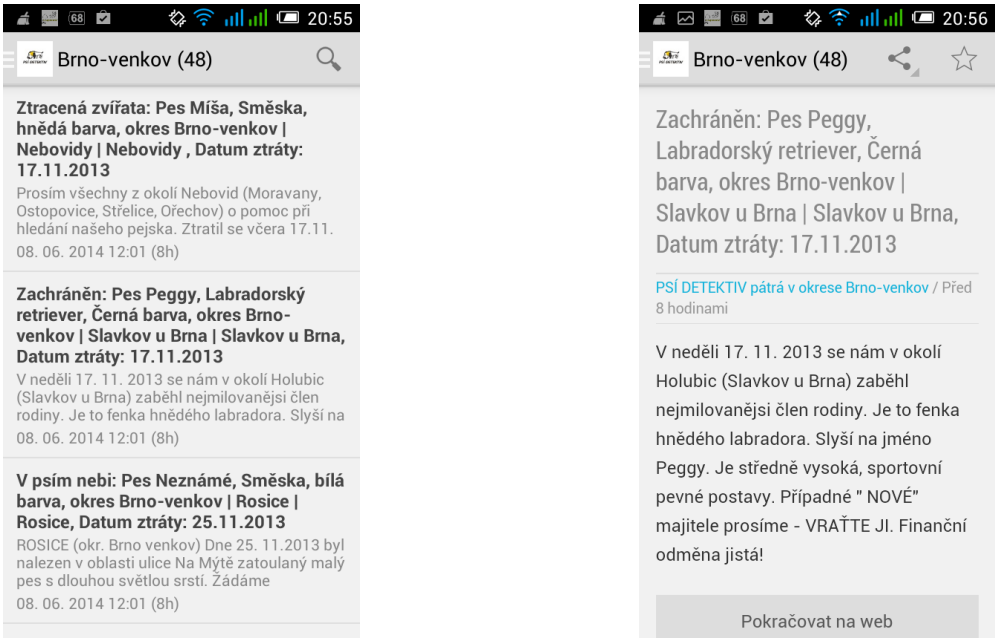
Jedná se o projekt pomáhající hledat ztrácená zvířata podle krajů a následně okresů. Ke spuštění aplikace není třeba se registrovat. Uživateli je umožněno tyto informace dále přeposílat prostřednictvím emailu, Facebooku a Skypu.

Aplikace umožňuje jednotlivé příspěvky přidávat do oblíbených, ale posléze tyto příspěvky nelze jednoduše zobrazit, jak by bylo očekáváno.

Uživatelská přívětivost není na příliš vysoké úrovni. Odrazující vzhled je dán především nevhodnou volbou barev, jak je možné vidět na obrázku 11. Pozitivní je, že uživatel může měnit barevné téma (pouze 2 barevná témata) a rozložení aplikace. Účel této aplikace je dobročinný, bohužel nebyly použity vhodné prostředky (viz tabulka 5).

Tabulka 5: Shrnutí aplikace Psí detektiv

Hodnocení na Google Play	4,1 z 32
Práce se sociálními sítěmi	Sdílení příspěvků
Informace na mapě	Ne
Online x offline	Online
Vzhled a ovládání	Podprůměrné
Více jazyků	Ne
Minimální verze Android	2.3 and vyšší
Velikost aplikace	4.3 MB



(a) Seznam ztracených psů (b) Detail ztraceného psa

Obrázek 11: Náhled aplikace Psí detektiv
(Převzato z Google Play, 2016k)

4.1.2.2 Můj pes

Následující aplikací je demoverze knihy o plemenech psů. Z tohoto důvodu není nutná registrace. Každé plemeno je představeno včetně fotografií, jsou zdůrazněny důležité informace prostřednictvím výstižných ikon jako je např. velikost či povaha plemene (obrázek 12a). Pokud se daný pes nenachází v seznamu, je uživateli umož-

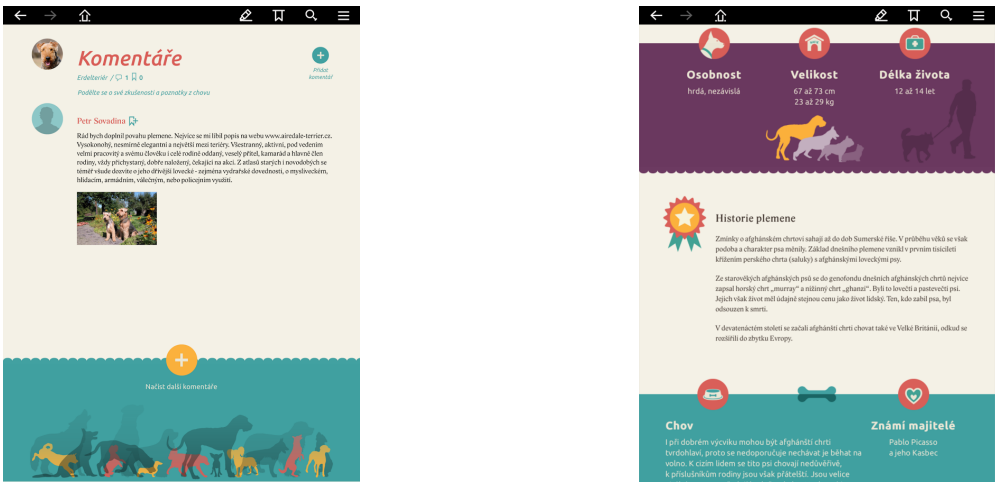
něno vytvořit vlastní medailonek psa. Uživatelé mohou publikovat, sdílet fotografie a informace o svých psech.

Pozitivně je hodnocena práce s textem, je možné zvýraznit určité části textu. Posléze tyto poznámky lze přehledně zobrazit. V nastavení je možné přizpůsobit velikost písma. Ovládání a vzhled jsou odlišné od zbývajících aplikací, nejedná se klasickou Android aplikaci. Kladně lze hodnotit nápovědu vysvětlující ovládání a funkce knihy vzhledem k její specifické manipulaci. Nepatrným nedostatkem může být horší viditelnost navigačních ikon, což je dáno jejich menší velikostí.

Jak lze vidět na obrázku 12b jedinou sociální interakcí mezi uživateli je přidání komentáře týkajího se daného psa. Dalším nedostatkem je fungování pouze za přepokladu internetového připojení. Ačkoliv aplikace nesplňuje aspekt sociální aplikace, je celkově hodnocena jako informační prostředek lepšího průměru (viz tabulka 7).

Tabulka 6: Shrnutí aplikace Můj pes

Hodnocení na Google Play	3,8 z 42
Práce se sociálními sítěmi	Ne
Informace na mapě	Ne
Online x offline	Online
Vzhled a ovládání	Průměrné
Více jazyků	Ne
Minimální verze Android	4.1 and vyšší
Velikost aplikace	7.5 MB



(a) Popis plemene (b) Detail plemene

Obrázek 12: Náhled aplikace Můj pes
(Převzato z Google Play, 2016i)

4.1.2.3 Po stopě

Aplikace Po stopě je jedinou českou sociální aplikací pro majitelé psů, která je poskytována zdarma. Významnou součástí aplikace je mapa, kde je možné co nejrychleji vyhledat informace o důležitých místech týkajících se psa jako veterinář nebo chovatelské potřeby (obrázek 13a). Odlišující funkcionalitou od ostatních aplikací je možnost venčení. Touto akcí se ostatní uživatelé mohou dozvědět o právě venčících majitelech psů. V souvislosti s venčením se v aplikaci nachází uchovávání jejich historie. Bohužel nelze zpětně tyto trasy zobrazit na mapě.

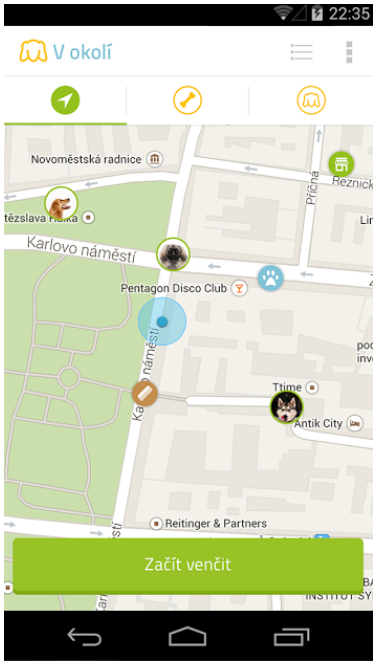
Další odlišností od ostatních zmíněných aplikací je psí škola obsahující videa a krátký popis demonstrující obrázek 13b. Na základě úspěšně splněného výcviku uživatel sbírá odměny, které může posléze proměnit za slevy v e-shopu. Během výcviku uživatel může natáčet vlastní videa a poté je sdílet na sociálních sítích.

Nevýhodu je možné vypozorovat v akci hledání nových přátel, které lze získat pouze během venčení. Aplikace není rozšířena mezi mnoho uživatelů a reálně může nastat problém při nalezení nového psího kamaráda.

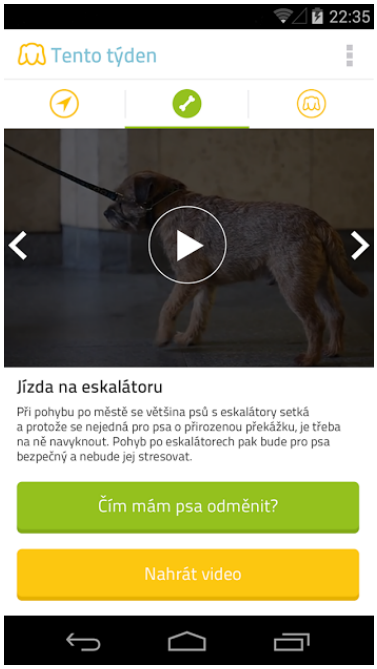
Barevná kompozice aplikace je dobře zvolená. Negativně je hodnoceno vytvoření dalšího psa. Není hned z navigace zřejmé, kde je možné dalšího psa přidat. Je nutné kliknout na defaultní panel mobilního zařízení, až poté se v nabídce nastavení zobrazí možnost přidat dalšího psa. Na rozdíl od zmíněných aplikací nabízí jiné funkce, ale nedosahuje takových kvalit jako např. Dogalize v podkapitole 4.1.1.2.

Tabulka 7: Shrnutí aplikace Po stopě

Hodnocení na Google Play	3,8 z 185
Práce se sociálními sítěmi	Přihlášení přes Facebook a sdílení
Informace na mapě	Ano s možností filtrování
Online x offline	Online
Vzhled a ovládání	Průměrné
Více jazyků	Ne
Minimální verze Android	2.3 and vyšší
Velikost aplikace	9 MB



(a) Mapa umožňující filtrování



(b) Psí škola

Obrázek 13: Náhled aplikace Po stopě
(Převzato z Google Play, 2016j)

4.1.3 Zhodnocení analýzy

Po provedené analýze současných Android aplikací podobně zaměřených je možné přistoupit k jejich zhodnocení. Vzhledem k malému počtu českých zdarma dostupných aplikací bylo rozhodování poměrně snadné. Jako nejlepší zástupce byla zvolena aplikace Po stopě, která propojuje komunitu chovatelů psů a zároveň využívá principy geolokace (viz tabulka 8b).

V zahraničím prostředí je konkurence podstatně vyšší. Poměrně náročné bylo vybrat vhodné zástupce. S přihlédnutím ke zvoleným kritériím aplikace Dogalize je splňuje nejlépe. Uživatel má k dispozici řadu užitečných funkcí jako je vytvoření upomínky, oznámení o ztraceném zvířeti. Ocenit lze zejména možné způsoby socializace mezi majiteli psů (viz tabulka 8a).

Tabulka 8: Vyhodnocení aplikací

(a) Vyhodnocení zahraničních aplikací

1.	Dogalize
2.	11Pets
3.	MookieApp

(b) Vyhodnocení českých aplikací

1.	Po stopě
2.	Můj pes
3.	Psí detektiv

4.2 Analýza cloudových řešení

Cloud computing byl krátce představen v podkapitole 2.3. Cílem této analýzy je vybrat nejvhodnějšího poskytovatele MBaas. Dle provedeného průzkumu Májským mezi tři nejvýznamnější poskytovatelé MBaas patří Parse, Firebase a Kinvey (Májský, 2016, s. 27). Vzhledem k ukončení činnosti společnosti Parse a nejisté situaci ohledně jejího dalšího vývoje, nebude tato společnost zahrnuta do analýzy (Lacker, 2016). Z tohoto důvodu toto místo bude nahrazeno firmou Backendless. Analýza bude spočívat v hodnocení jednotlivých poskytovatelů podle následujících kritérií:

- Datové úložiště
- Správa uživatelů – registrace, přihlášení (včetně sociálních sítí), změna hesla, obnova zapomenutého hesla
- Podpora více platforem
- Synchronizace a offline fungování
- Zasílání notifikací
- Geolokace
- Práce s multimédii
- Cena
- Analytické nástroje
- Hosting
- Debugování
- Vlastní aplikační logika na serveru
- Kvalita dokumentace
- Nástroje zviditelnění aplikace

4.2.1 Firebase

Firebase nabízí uživateli tři základní částí pracující samostatně, ale společně tvoří ucelený celek zajištující bezproblémový chod aplikace.

První důležitou částí jsou nástroje týkající se vývoje jako např. databáze, cloud messaging, hosting pro vlastní webové stránky, ale také testování aplikace.

Data ve Firebase jsou uložena v NoSQL databázi[1] jako stromová struktura v podobě JSON. Velkou předností této realtime databáze je udržování synchronního stavu pro všechny klienty. Pokud se klient odpojí od internetového připojení,

[1] Jedná se o odlišný způsob ukládání dat oproti relačním databázím vhodný především pro velké množství dat. Nejčastěji data bývají uložena jako key-value. Dalším ze způsobů je ukládání v podobě grafu (W3Resource, 2016)

Firebase je schopný zajistit fungování offline prostřednictvím ukládání dat lokálně. Při dalším připojením klienta dojde k synchronizaci s aktuální verzí.

Firebase poskytuje autentifikaci uživatelů. Jedním ze způsobu je registrace nového uživatele nebo přihlášení prostřednictvím poskytovatelů sociálních sítí. Firebase na základě údajů získaných od uživatele jej ověří, v případě úspěšného přihlášení může klient dále tohoto uživatele spravovat. V případě zapomenutého hesla Firebase umožňuje jeho obnovu prostřednictvím zaslaného emailu uživateli.

Cloud messaging umožňuje klientovi zasílat notifikace uživatelům dle svých preferencí. Může se jednat např. o zprávu určenou jednotlivci, skupině uživatelů (sdělení o akci VIP uživatelům) či podle tématu (např. uživatelé určitého diskuzního fóra).

Práce s mapou je realizována prostřednictvím knihovny Geofire. Výhodou této knihovny je vytváření dotazu vracející objekty v dané oblasti, čehož by se dalo využít při zobrazení nejbližšího veterináře a jiných míst od lokace daného uživatele.

Firebase nabízí bezpečnou prácí s multimédií s využitím Google Cloud Storage. Pokud dojde k přerušení nahrávání či stahování, je obnoveno v tom místě, kde k přerušení došlo. Klientovi je umožněno definovat, jestli obsah je veřejný nebo soukromý.

Nezpochybnitelnou výhodou je možnost testování aplikace na různých zařízení umožněné nástrojem Test Lab. Z tohoto důvodu není potřeba vlastnit velké množství fyzických zařízení k testování, jelikož jsou zpřístupněné přes Google Data Center. Mimo jiné jsou k dispozici tzv. Robo test prověřující správné fungování aplikace bez nutnosti psání vlastních testů. S testováním souvisí také další sada vývojařských nástrojů, a to Crash Reporting. Tento nástroj má za cíl spravovat chyby, které se vyskytly během běhu aplikace.

Za zmínku také stojí funkce zvaná Remote Config, která poskytuje klientovi možnost změnit vzhled či funkcionalitu aplikace, aniž by bylo nutné vyžadovovat stažení nové aktualizace uživatelem.

Druhou částí je tzv. Analytics umožňující vytváření statistik dle různých kritérií. Tento nástroj lze využít v situacích, kdy je potřeba nasimulovat chování uživatele a na základě těchto výsledků provést odpovídající změny. Zajímavou a užitečnou funkcionalitou je možnost simulace změny aplikace pouze pro určitou skupinu uživatelů. Následné změny lze např. ověřit A/B testováním.[2]

Poslední částí jsou nástroje sloužící k zviditelnění aplikace. Nástroj App Indexing umožňuje spouštět nainstalovanou aplikaci přímo z webového prohlížeče, zatímco uživatel vyhledává podobný obsah ve vyhledavači Google. Tímto způsobem lze uživatele upozornit k užívání této aplikace. Firebase Invites nabízí dvě možnosti, jak přizvat další uživatelé k používání aplikace, a to pomocí SMS nebo emailové pozvánky. Souhrné informace o Firebase lze vidět v tabulce 9.

[2]Jedná se o metodu porovnávající dvě různé varianty aplikací náhodnými uživateli. Na základě výsledků je možné se rozhodnout, zda-li změny je vhodné provést či nikoliv (Optimizely, 2016).

Tabulka 9: Shrnutí poskytovatele Firebase

Typ datového úložiště	NoSQL – stromová struktura JSON
Správa uživatelů	Ano
Podpora více platforem	Ano, Android a iOS
Synchronizace a offline	Ano, realtime databáze
Zasílání notifikací	Ano
Geolokace	Ano, Geofire
Práce s multimédii	Ano, Google Cloud Storage
Cena	Zdarma (určitá omezení datového uložiště aj.)
Analytické nástroje	Vynikající
Hosting	Ano
Vlastní aplikační logika	Ne
Debugování	Ano, Robo test
Kvalita dokumentace	Vynikající
Nástroje zviditelnění aplikace	Ano (App Indexing, Invites, AdWords)

4.2.2 Kinvey

Druhým zástupcem je poskytovatel Kinvey, především zaměřující se na podniky. Kinvey používá k ukládání dat NoSQL databázi konkrétně MongoDB. Základní jednotkou je entita představující množinu klíč-hodnota. Entity stejného typu jsou dále uspořádany do kolekcí.

Správa uživatelů spočívá v akcích jako registrace, přihlášení včetně sociálních sítí, obnova hesla a odhlášení. V rámci registrace existují dva typy, a to explicitní a implicitní. V případě explicitní registrace se jedná o typický scénář – získání informací od uživatele prostřednictvím formuláře. Kinvey se stárá o duplicitu registrovaných uživatelů, není tedy možné zaregistrovat uživatele se stejným jménem. K dispozici je také ověření uživatele při registrací potvrzovacím emailem, defaultně není tato možnost dovolena. Implicitní registrace je vhodná v situacích, ve kterých není potřeba udržovat uživatelův účet. Vytvoří se automaticky vygenerovaný účet. Přihlášení je realizováno dvojím způsobem. Uživatel se může přihlásit svými získanými přihlašovacími údaji. Kromě tohoto způsobu Kinvey podporuje autentizaci prostřednictvím sociálních účtů konkrétně Facebook, Twitter, Google+ a LinkedIn.

Synchronizace je realizována cachováním, které spočívá v ukládání dat lokálně pokud je zařízení v režimu online. K dispozici je několik strategií definujících použití

cache. Jedná se např. zakázání používání cachování, přednostní získávání dat online nebo naopak z cache. S cachováním úzce souvisí offline funcionalita. Tento princip spočívá v manipulaci lokální kopie dat a po získání internetového připojení dojde k automatické synchronizaci.

Obdobně jako u Firebase je možné informovat uživatele o určité události prostřednictvím notifikací, které mimo jiné mohou být zasílány z vlastní aplikační logiky backendu. Kinvey nabízí tři možnosti, jak přizpůsobit chování backendové části pomocí skriptu v Javascriptu. Jedná se o vytvoření Collection Hooks, které jsou spouštěné v závislosti na změnu stavu entity. Dalším případem je skript, který lze volat z klienta tzv. Endpoints. Poslední variantou Scheduled Code je skript, který se spouští pravidelně v určitých intervalech.

Jako v případě Firebase je klientovi umožněna práce s lokalizačními údaji. Pro tuto funkci je nutné uložit souřadnice místa do atributu geoloc. Následně provádět dotazy nad tímto atributem za využití tří operátorů. První operátor umožňuje nalézt entity v daném okruhu, naopak druhý vyhledá entity nacházející se v daném obdélníku definovanými dvěma body. Třetí operátor vyhledá entity v oblasti určené více než dvěma body.

Analogicky jako u prvního poskytovale je realizována práce s multimédii. Je využito služeb třetí strany konkrétně Google Cloud Storage. Ukládaný soubor může mít velikost až 5 TB. U každého souboru lze nastavit viditelnost a dobu vypršení platnosti odkazu pro stažení jinými uživateli. V případě veřejného souboru může být stažen kterýmkoli uživatelem a jeho doba platnosti není omezena. Díky Media Streaming není nutné načíst celý soubor do paměti, čímž se snižují nároky na paměť a baterii zařízení. Splnění jednotlivých kritérií demonstruje tabulka 10.

Tabulka 10: Shrnutí poskytovatele Kinvey

Typ datového úložiště	NoSQL – entity ve formátu JSON
Správa uživatelů	Ano
Podpora více platforem	Ano, Android, iOS a Xamarin
Synchronizace a offline	Ano
Zasílání notifikací	Ano
Geolokace	Ano
Práce s multimédii	Ano, Google Cloud Storage
Cena	Zdarma (určitá omezení datového uložiště aj.)
Analytické nástroje	Ano, ale pouze pro placenou verzi
Hosting	Ano
Vlastní aplikační logika	Ano
Debugování	Ano
Kvalita dokumentace	Dobrá
Nástroje zviditelnění aplikace	Ne

4.2.3 Backendless

Ukládání dat v Backendless je založeno na objektech a vazbách mezi nimi, jedná se tedy o SQL databázi. K prohlížení dat a provedení CRUD operací je možné využít grafického prohlížeče.

Backendless umožňuje kompletní správu uživatelů zahrnující proces registrace nového uživatele (včetně validace vstupů), jeho následná editace, možnost obnovy zapomenutého hesla. Důležitou funkcí pro vyvíjenou aplikaci je možnost integrace se sociálními sítěmi (Facebook, Twitter, Google+).

Synchronizace není implicitně dostupná jako u Firebase, podobného efektu je možné dosáhnout pomocí Publish/Subscribe Messaging. Backendless prozatím nedisponuje možností offline fungování.

Pokud uživatel v danou chvíli aktivně aplikaci nevyužívá, je potřeba využít notifikací k oznámení určité události. K tomuto účelu je navržen nástroj zvaný Push notification, který může zobrazit notifikace v různých podobách. Tato funkce by byla vhodná pro řešení, jak oznámit uživateli různé typy žádosti v rámci vyvíjené aplikace.

Geolokace je realizována prostřednictvím geopoints, kterým lze nastavit souřadnice. Ty body mohou být dále vyhledávány, filtrovány a tříděny do skupin. K debugování dotazů geografických informací je k dispozici Geo browser zobrazující výsledky na mapě.

Backendless podporuje práci s multimédii na více platformách, které je možné spravovat přímo ve webové konzoli. Ke každému souboru je možné nastavit oprávnění či roli. Veškerá média jsou ihned k dispozici ke stažení, spuštění a provedení dalších akcí, aniž by se o ně musel klient starat.

Kromě výše uvedených funkcí Backendless nabízí možnost vytvoření vlastní business logiky na serverové části prostřednictvím intuitivního webového rozhraní tzv. Server Code Assistant. Uživatel může vytvářet event handlers pouhým kliknutím, nastavit schedulers, které se budou opakovaně vykonávat v určitý stanovený čas.

Velkou výhodu představuje lokální debugování. Mimo jiné Backendless usnadňuje práci při logování, jelikož jednotlivé logy mohou být uchovávány pro další použití.

V rámci nástroje Analytics je možné vytvářet grafy a tabulky či filtrovat informace dle zadaného kritéria. Tyto funkce umožnují sledovat úspěšnost používané aplikace. Splnění jednotlivých kritérií objasňuje tabulka 11.

Tabulka 11: Shrnutí poskytovatele Backendless

Typ datového úložiště	Objektově-relační
Správa uživatelů	Ano
Podpora více platform	Ano (Android, iOS, Windows Phone)
Synchronizace a offline	Částečně
Zasílání notifikací	Ano
Geolokace	Ano
Práce s multimédii	Ano
Cena	Zdarma (určitá omezení v množství dotazů aj.)
Analytické nástroje	Dostačující
Hosting	Ano
Vlastní aplikační logika	Ano, Server Code Assistant
Debugování	Ano
Kvalita dokumentace	Dobrá
Nástroje zviditelnění aplikace	Ne

4.2.4 Vyhodnocení nejvhodnějšího MBaaS

Během analýzy byly brány v potaz kritéria definovaná v podkapitole 4.2. Jednotliví poskytovatelé jsou si v mnoha ohledech podobní. Faktory správa uživatelů, synchronizace a možnost offline fungování měly největší vliv na výsledné hodnocení poskytovatelů (tabulka 12).

Tabulka 12: Shrnutí všech poskytovatelů

	Firebase	Kinvey	Backendless
Typ datového úložiště	NoSQL	NoSQL	SQL
Správa uživatelů	x	x	x
Podpora více platforem (Android, iOS)	x	x	x
Synchronizace a offline	x	x	-
Zasílání notifikací	x	x	x
Geolokace	x	x	x
Práce s multimédii	x	x	x
Cena	Zdarma	Zdarma	Zdarma
Analytické nástroje	Výborné	Dostačující	Dobré
Hosting	x	x	x
Vlastní aplikační logika	-	x	x
Debugování	x	x	x
Kvalita dokumentace	Výborná	Dobrá	Dobrá
Nástroje zviditelnění aplikace	x	-	-

Zajímavé cloudové řešení nabízí Backendless, ale bohužel některé podstatné části jsou stále v řešení jako např. offline funkcionalita. Výhodou může být intuitivnější používání relační databáze, ale které nemusí být vhodné pro velké projekty. Jako jediný poskytovatel nabízí platformu Windows Phone. Jedná se o menší společnost v porovnání s Firebase a Kinvey, proto také jeho komunita není tak velká a aktivní.

Oproti tomu Kinvey se soustředí na enterprise řešení. Mnoho funkcí je omezeno v rámci bezplatného používání. Jako výhodu toho cloudového řešení lze zdůraznit cachování a správu uživatelů.

Nejlépe kritéria splnil poskytovatel Firebase. Nabízí i další nástroje jako jsou zviditelnění aplikace a testování na řadě zařízení. Bylo také přihlédnuto k faktu,

že Firebase je vlastněn Google a jeho komunita je aktivní. Z tohoto důvodu lze považovat jeho budoucí vývoj za stabilní. Mimo jiné Firebase je využíván řadou známých společností jako Atlassian, Skyscanner. Nevýhodou může být absence vytvoření vlastní aplikační logiky prostřednictvím skriptu jako je tomu možné u dalších dvou poskytovatelů.

4.3 Definování funkčních a nefunkčních požadavků

Pro jasnější představu funkcionalit aplikace je vhodné si stanovit funkční požadavky. Nefunkční požadavky budou stanoveny jako možnosti pro zlepšení aplikace v budoucnu.

1. **Funkční požadavky:**
    - Možnost přihlášení přes sociální síť Facebook[3]
    - Vytvoření profilu uživatele a jeho psů včetně možnosti obnovy hesla s využitím Firebase Authentication
    - Umožnění snadné komunikace mezi pejskaři – implementace chatu
    - Možnost přátelství mezi psy
    - Nalezení informací o dalších psích plemenech přímo v aplikaci
    - Sdílení obsahu
    - Možnost vytvoření upomínek a deníku
    - Notifikace ke společnému venčení
    - Notifikace k získání pozornosti uživatele „Štěknout“
    - Nalezení nejbližšího veterináře, chovatelských potřeb, parku aj. pomocí Google Places API
    - Vlastní nastavení - změna profilové fotky, nastavení ohledně notifikací
    - Přidávání psích plemen do oblíbených
2. **Nefunkční požadavky:**
    - Zveřejnění na Google Play
    - Možnost dostupnosti aplikace na více platformách
    - Rozšíření databáze psů a míst
    - Ukládání tras venčení s jejich následnou možností vizualizace na mapě

[3]Facebook byl zvolen z důvodu, že je nejpoužívanější sociální sítí v České republice s 87,27 % (Statsmonkey, 2015). Tento údaj lze také vidět na obrázku 30 v příloze B, kde je demonstrováno používání jednotlivých sociálních sítí na celém světě.

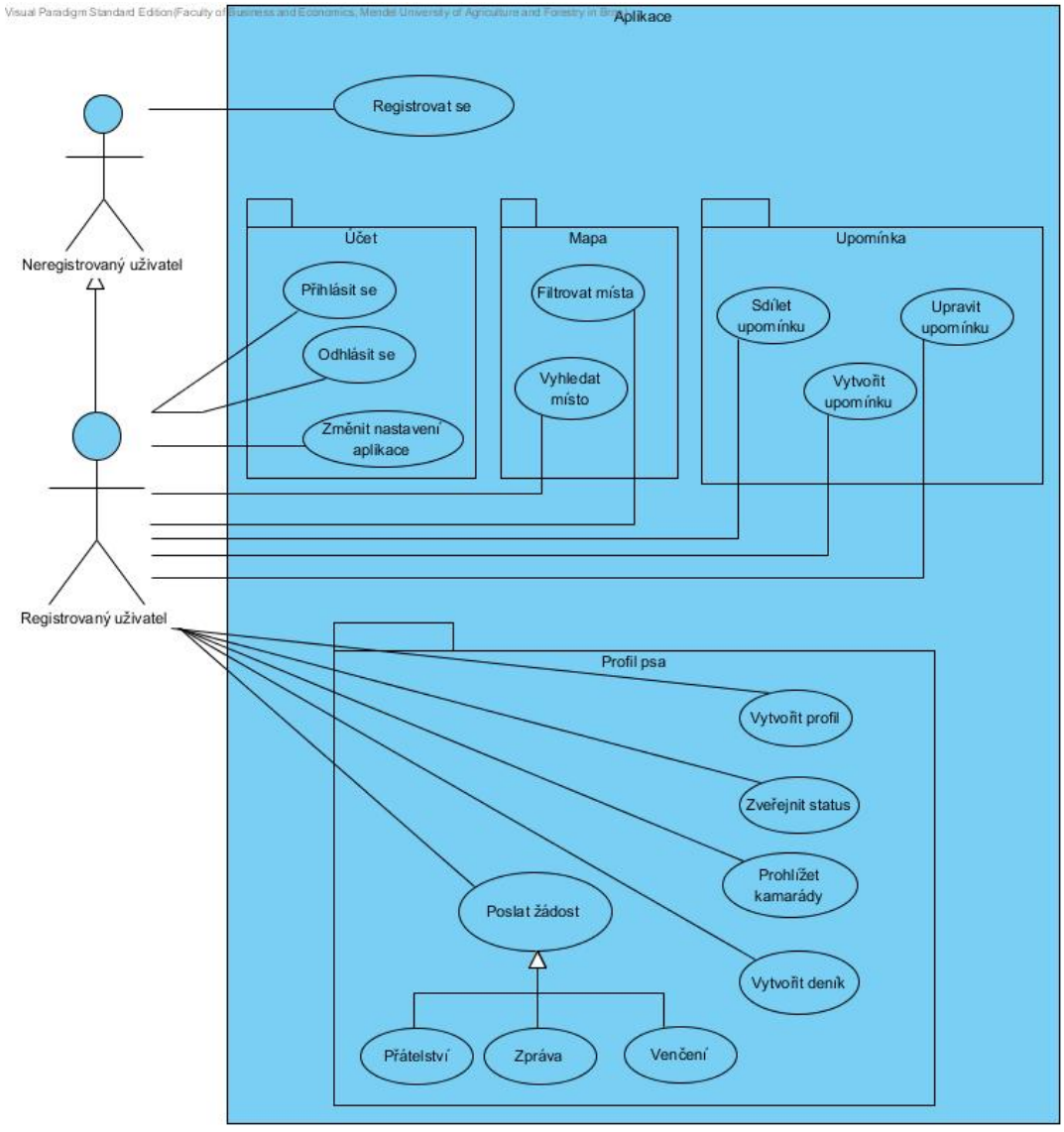
- Přihlášení přes více sociálních sítí
- Možnost volby více jazyků
- Vytvoření diskuzního fóra, skupinového chatu
- Využití inteligentních psích obojků pro monitorování pohybu psů
- Vytvoření zdi příspěvků
- Informace o ztracených psech a otrávených návnadách
- Možnost nákupů prostřednictvím aplikace
- Poskytnutí informací o veletrzích a dalších akcích
- Vytváření statistik ohledně progresu zvířete
- Fungování aplikace v offline režimu za využití Firebase Realtime Database
- Snížení velikosti aplikace
- Umožnění přístupu více uživatelům na jednom zařízení
- Sdílení vytvořeného psa v rámci jedné rodiny
- Obrazová nápověda, případně video
- Možnost výcvikových videií
- Zrychlení stažení dat zakoupením placené verze Firebase

5 Návrh aplikace

Provedená analýza umožňuje přistoupit ke konkrétnímu návrhu. Návrh spočívá ve vytvoření UML diagramů a následnému grafickému návrhu aplikace.

5.1 Diagram případů užití

Jak lze vidět na obrázku 14, hlavní aktérem je registrovaný uživatel, který je specializací neregistrovaného uživatele. Jednotlivé případy užití jsou seskupeny podle společných rysů do balíčků. Scénaře vybraných případů užití jsou uvedeny v tabulkách níže. Zbývající scénaře jsou uvedeny v příloze D.



Obrázek 14: Diagram případů užití

Tabulka 13 popisuje přihlášení uživatele do aplikace. Je nutné, aby uživatel již měl přihlašovací údaje nebo zvolil přihlášení svým facebookovým účtem.

Tabulka 13: Scénář případu užití Přihlásit se

Případ užití: Přihlásit se
ID: UC02
Účastníci: registrovaný uživatel
Vstupní podmínky: 1. Uživatel má zapnutou aplikaci 2. Uživatel není přihlášen
Tok událostí: 1. Uživatel zvolí způsob přihlášení 2. Pokud uživatel zvolí přihlášení přes Facebook: 2.1 Systém přihlásí uživatele přes jeho facebookový účet 2.2 Případ užití končí 3. Jinak systém vyžaduje vložení přihlašovacích údajů 4. Uživatel vyplní přihlašovací údaje 4.1 Uživatel stiskne tlačítko Přihlásit 5. Systém průběžně ověřuje vstupy 6. Pokud jsou přihlašovací údaje správné 6.1 Systém přihlásí uživatele do aplikace
Výstupní podmínky: 1. Uživatel je přihlášen 2. Systém přesměruje uživatele na seznam jeho psů
Alternativní tok: 1. Systém zobrazí chybové hlášení 2. Případ užití začíná od 1. kroku

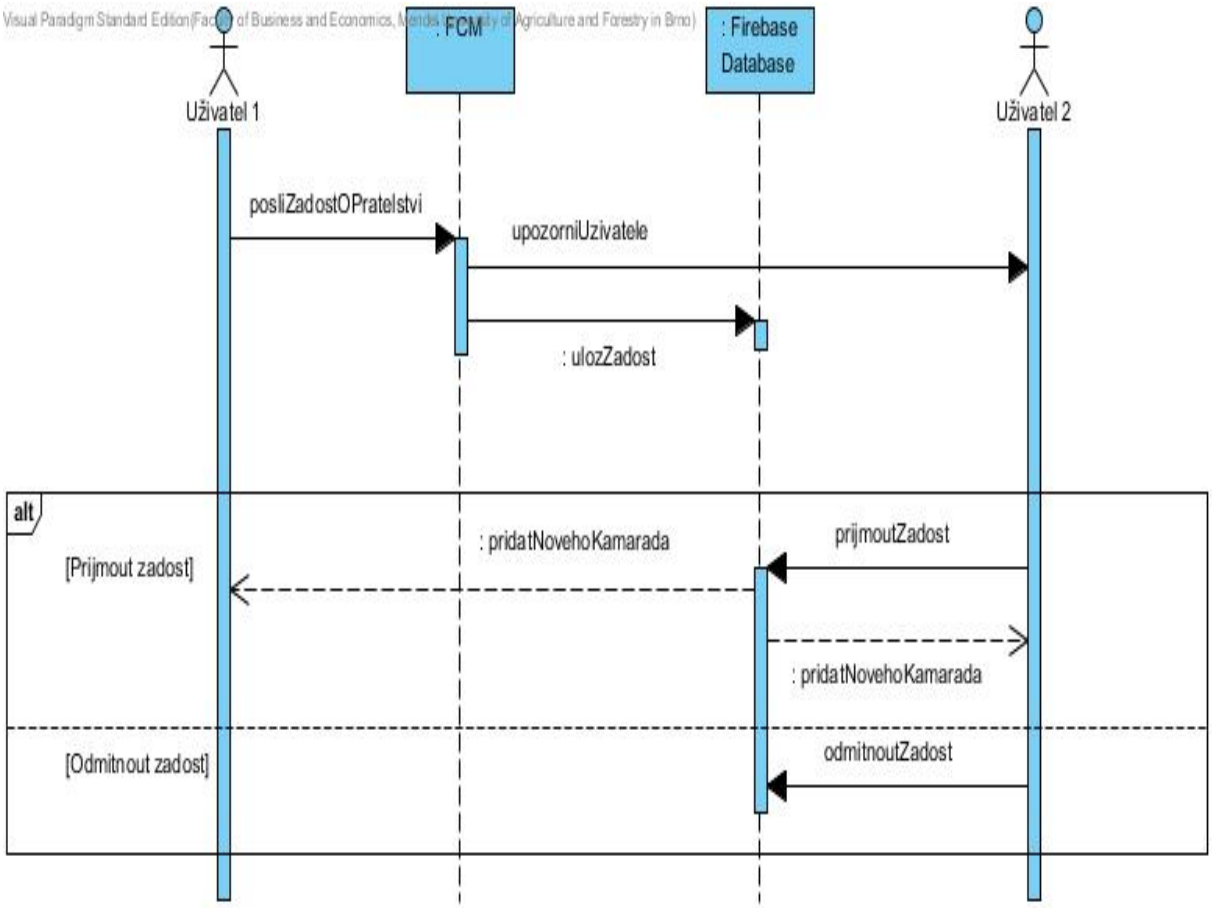
Tabulka 14 vysvětluje funkcionalitu případu užití Poslat žádost o přátelství, který je specializací Poslat žádost. Prostřednictvím tohoto případu užití může uživatel požádat psa jiného uživatele o přidání do seznamu přátel. Příjemce má možnost žádost o přátelství odmítnout.

Tabulka 14: Scénář případu užití Poslat žádost o přátelství

Případ užití: Poslat žádost o přátelství
ID: UC11
Účastníci: registrovaný uživatel
Vstupní podmínky: 1. Uživatel je přihlášený 2. Nachází se na záložce Kamarádi na obrazovce Můj profil
Tok událostí: 1. Uživatel klikne na tlačítko pro přidání nového kamaráda 2. Pokud existují další psi (bez kamarádů psa a jeho samotného): 2.1 Pro každého psa: 2.1.1 Systém zobrazí fotku psa 2.1.2 Systém zobrazí jméno psa 2.1.3 Systém zobrazí plemeno psa 2.1.4 Systém zobrazí další možnosti (např. Požádat o přátelství, poslat zprávu) 3. Uživatel vybere jednoho ze psů kliknutím na tlačítko Požádat o přátelství 4. Systém uloží žádost o přátelství do databáze 4.1 Pokud druhý uživatel má povolené notifikace: 4.1.1 Systém pošle notifikaci o žádost k venčení
Výstupní podmínky: systém oznámí uživateli informaci o úspěšně odeslané žádosti
Alternativní tok: v případě chyby systém zobrazí chybové hlášení

5.2 Sekvenční diagram

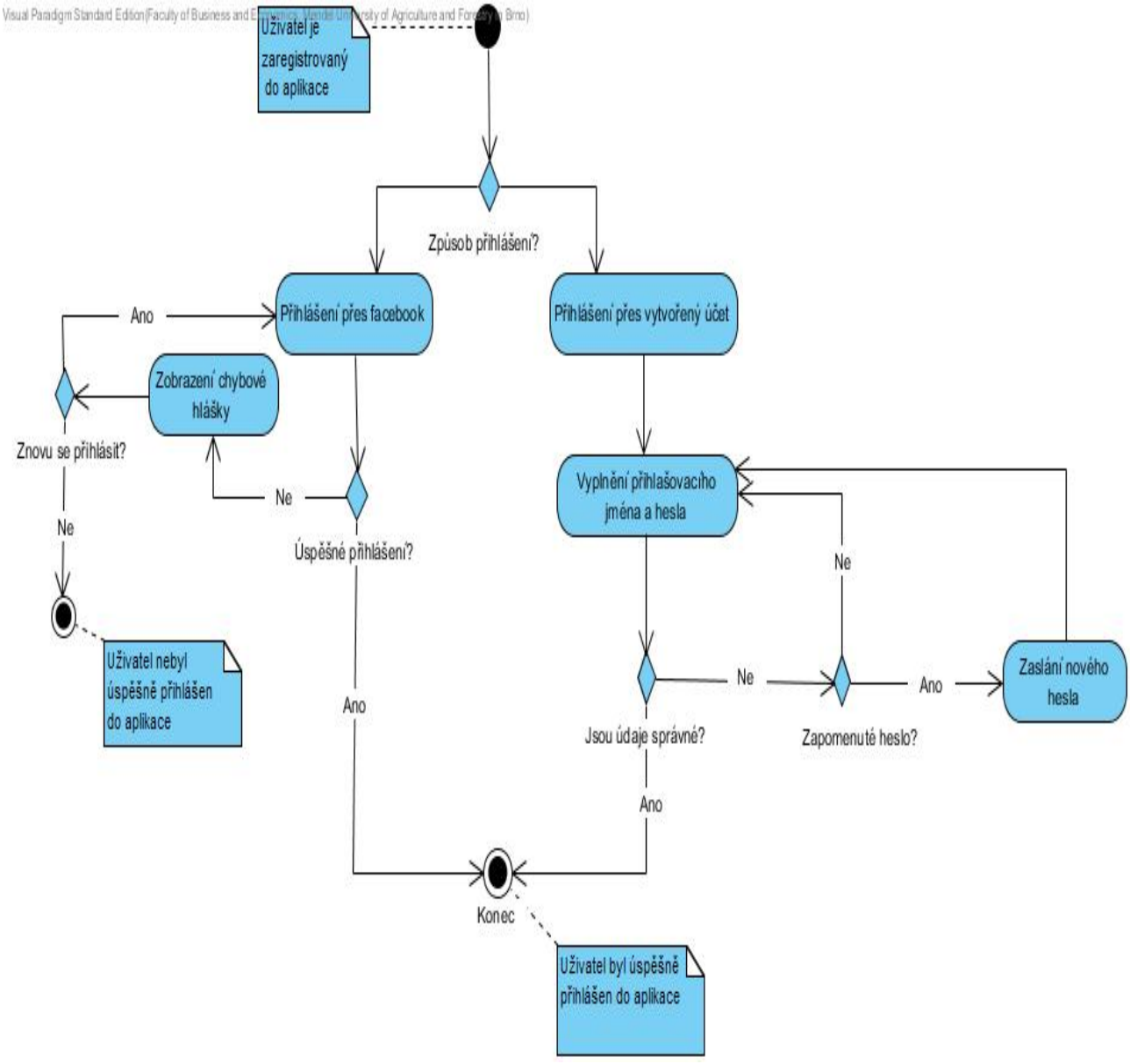
K případu užití Zaslat žádost o venčení byl vytvořen sekvenční diagram. Uživatel zašle žádost o venčení jinému uživateli a tato žádost se uloží v databázi. Prostřednictvím Firebase Cloud Messaging je druhý uživatel informován o dané akci. Následně příjemce může tuto žádost přijmout nebo odmítnout, tuto skutečnost zobrazuje „alt - alternative combined fragment“. V případě pozitivní odpovědi dojde k vzájemnému přidání psů do přátel.



Obrázek 15: Sekveční diagram

5.3 Diagram aktivit

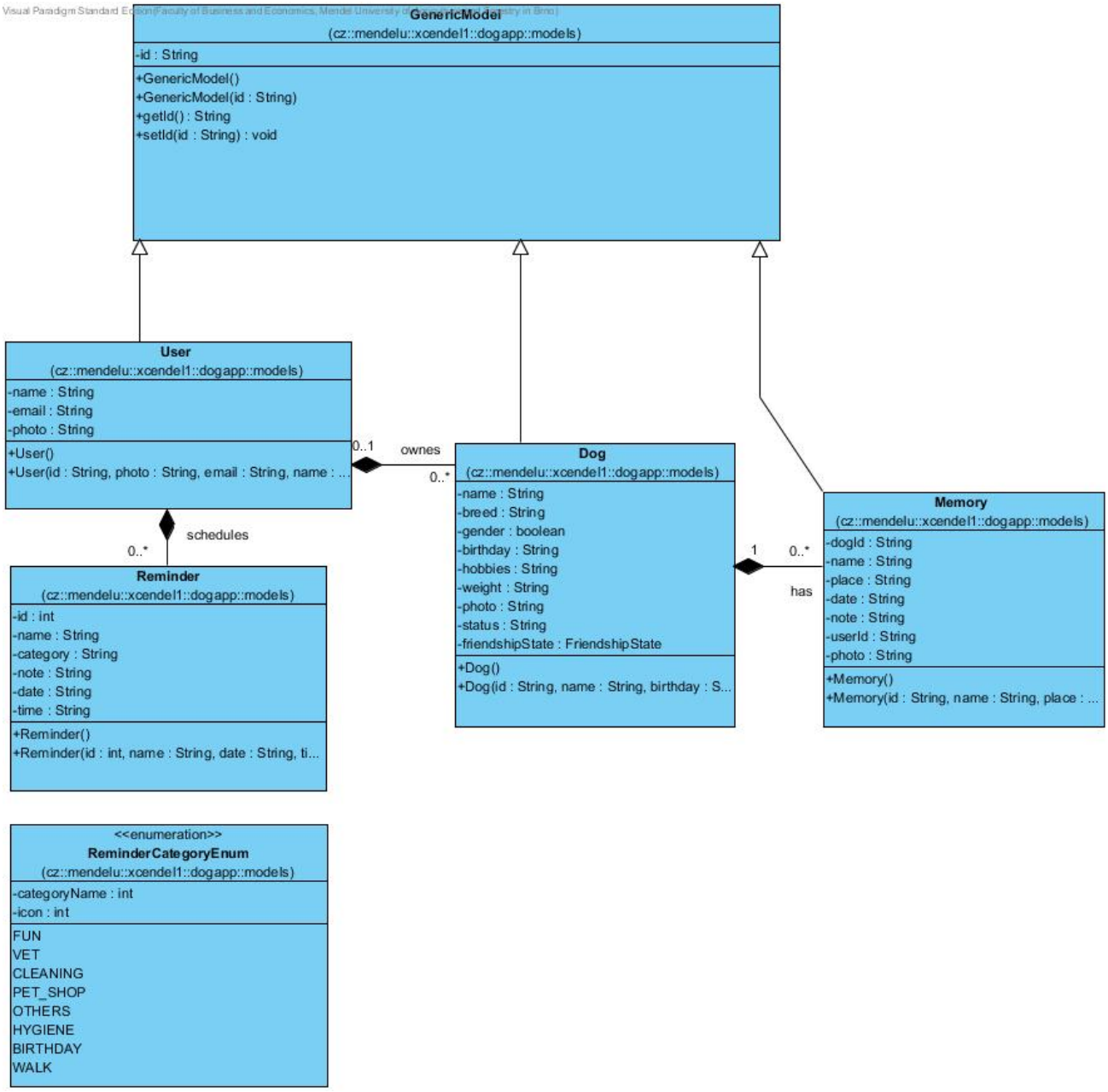
Přihlášení je důležitou částí každé aplikace, z tohoto důvodu byl tento případ užití využit pro modelaci diagramu aktivit (obrázek 16). Jak lze vidět v diagramu uživatel si může zvolit způsob přihlášení. Jedním ze způsobu je přihlášení přes jeho facebookový účet nebo přihlašovacími údaji získanými při registraci.



Obrázek 16: Diagram aktivit – přihlášení

5.4 Diagram tříd

Následující zjednodušený diagram tříd popisuje vztahy mezi základními objekty aplikace. S přihlédnutím k faktu, že je nelogické, aby pes existoval bez uživatele, byla zvolena mezi uživatelem a psem vazba typu kompozice. Vzpomínku lze vytvořit pouze k existujícímu psovi. Upomínka umožňuje několik typů prostřednictvím ReminderEnumCategory.



Obrázek 17: Zjednodušený diagram tříd

5.5 Grafický návrh

V následující části bude představen grafický návrh vybraných obrazovek aplikace. Zbývající návrhy obrazovek lze najít v příloze B. Při jejich návrhu bylo využito programu JustInMind. Obsahuje řadu předdefinovaných komponent, které jsou jedním přetažením ihned k dispozici k použití. Kromě vizualizace aplikace umožňuje nasimulovat reálné chování aplikace prostřednictvím animací a interakcí. Vytvořený prototyp lze dále exportovat a sdílet (JustInMind, 2016).

Vzhledem k podstatě aplikace bylo zvoleno následující barevné schéma. První barva je především použita ve status baru. Primární barvou je druhá barva v pořadí, lze ji najít v toolbaru. Jako accent barva je použita barva #ff9800.

Obrázek 18: Barevné schéma aplikace

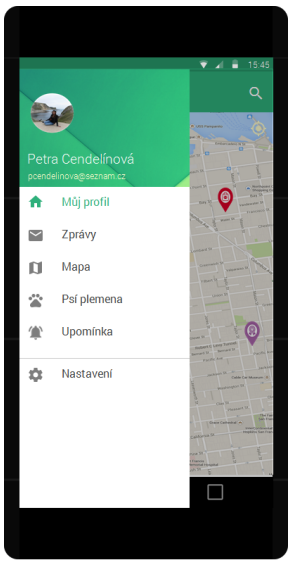
Ikony použité v aplikaci byly převzaty a upraveny z webové stránky Flaticon. Obrázek 19 ukazuje použití ikon tzv. markerů na mapě.

Obrázek 19: Ikony na mapě

Na obrázku 20 je možné vidět úvodní obrazovku s přihlášením. K dispozici jsou dva typy přihlášení, konkrétně pomocí emailu a uživatelova facebookového účtu.

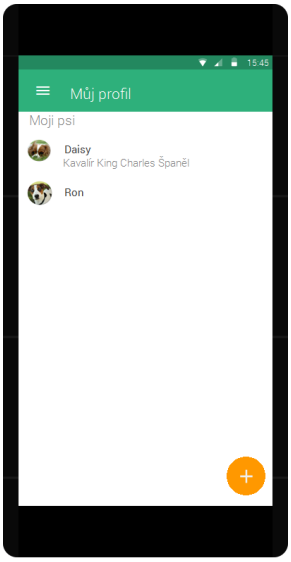
Hlavní navigace je realizována prostřednictvím bočního panelu tzv. Navigation Drawer jak lze vidět na obrázku 21. Uživateli umožňuje se jednoduše přepínat mezi jednotlivými obrazovkami.

Obrázek 20: Snímek úvodní obrazovky

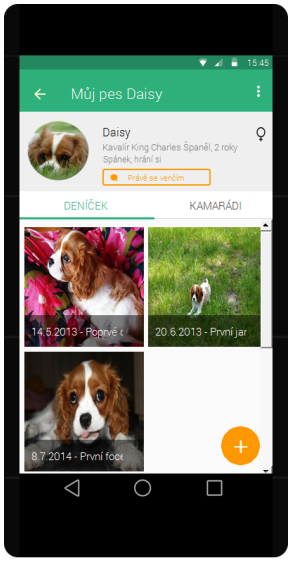


Obrázek 21: Snímek obr. - Hlavní navigace

Důležitou částí aplikace je domovská obrazovka uživatele zobrazená na obrázku 22. Na této obrazovce uživatel může vidět své vytvořené psy. Vybráním konkrétního psa se uživateli zobrazí profil tohoto psa viz obrázek 23.



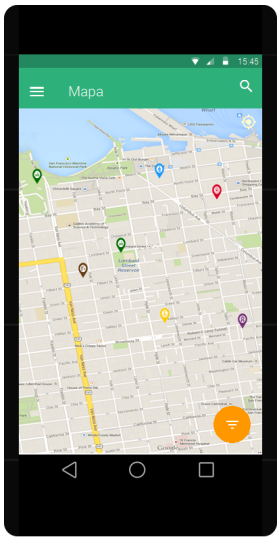
Obrázek 22: Snímek obr. Můj profil



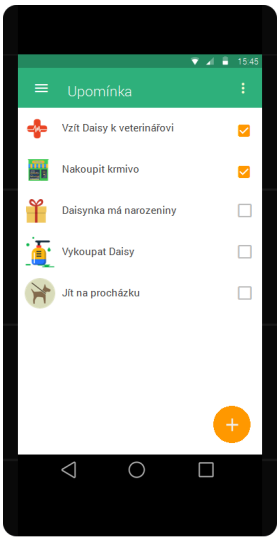
Obrázek 23: Snímek obr. Profil psa

Obrazovka Mapa (obrázek 24) umožňuje uživateli vyhledat si na mapě údaje o nejbližším veterináři, chovatelských potřebách, parku a jiných místech.

Aplikace umožňuje uživateli vytvářet upomínky týkající se jeho psů. Ve stanovený čas a datum budou připomenuty uživateli (obrázek 25).

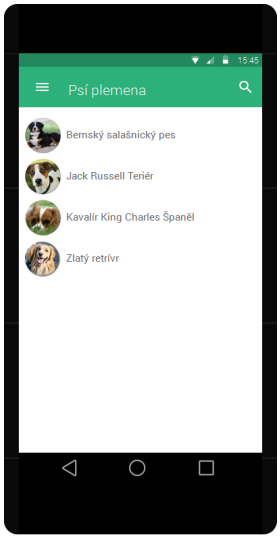


Obrázek 24: Snímek obrazovky Mapa

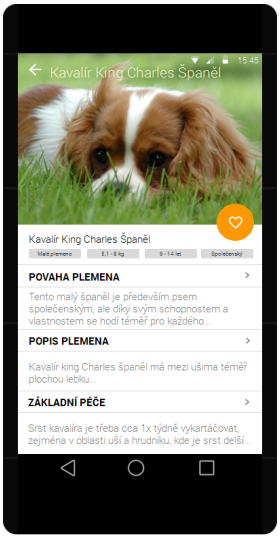


Obrázek 25: Snímek obr. Upomínka

V aplikaci budou dostupné informace o psích plemenech (obrázek 26). Po vybrání konkrétního plemene se uživateli zobrazí detail plemene v podobě zobrazené na obrázku 27. Kliknutím na FAB uživatel může toto plemeno přidat do oblíbených pro snadnější přístup.



Obrázek 26: Snímek obr. Psí plemena



Obrázek 27: Snímek obr. Detail plemene

6 Sestavení testovacího scénaře

Cílem následující kapitoly je sestavení testovacího scénáře, který má prověřit reálné použití aplikace. Při jeho sestavení se vychází z funkčních požadavků aplikace definovaných v kapitole 4.3. Testovácí scénář obsahuje následující úkony:

1. Přihlásit uživatele přes Facebook
2. Registrace uživatele
   - s fotkou a bez fotky
   - vyplnění nevalidních údajů
   - zadání již existujícího emailu
3. Obnova zapomenutého hesla
4. Přihlášení registrovaného uživatele
   - vyplnění špatného formátu emailu
   - přihlášení neexistujícím emailem a heslem
5. Vytvoření nového psa
   - s fotkou a bez fotky
   - zadání nevalidních vstupů – datum narození v budoucnu ve špatném formátu
6. Aktualizace dat o vytvořeném psovi včetně fotky => kontrola v profilu psa
7. Vytvoření psa na jiném zařízení
8. Zaslání žádosti o přátelství
   - zrušit žádost
   - odmítnout žádost
   - přijmout žádost => kontrola přátelství v profilech psů
9. Vytvořit vzpomínku u psa včetně fotky
   - zadání nevalidních dat – datum vzpomínky v budoucnu ve špatném formátu
   - sdílení vzpomínky
10. Aktualizace dat o vytvořené vzpomínce => kontrola v profilu psa a kamarádů
11. Zaslat žádosti o venčení, štěknutí
12. Zaslání zprávy jinému uživateli

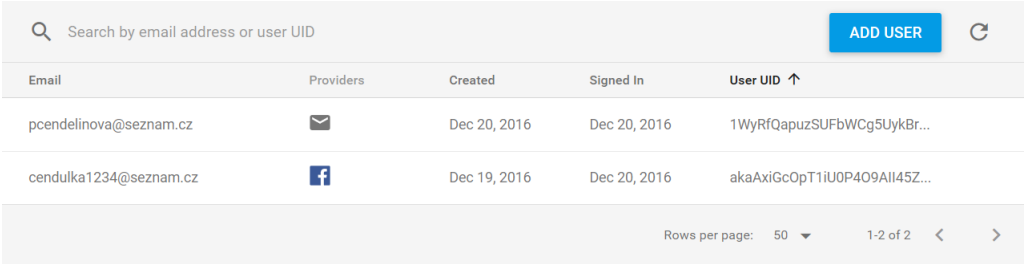
13. Vyhledání míst na mapě
14. Vytvořit, upravit, zrušit upomínku
15. Přidat psí plemeno do oblíbených, odebrat psí plemeno z oblíbených
16. Změnit nastavení aplikace
    - změna profilové fotky uživatele
    - povolení/zakázaní notifikací
    - povolení/zakázaní upomínek
    - zapnutí/vypnutí vibrací
    - změna vyzvánění notifikace
17. Odebrání psa z přátel
18. Vymazání vzpomínky, upomínky a psa
19. Odhlášení uživatele

7 Implementace aplikace

Následující kapitola se zabývá vývojem samotné aplikace. Zejména bude přiblížen postup přihlašování uživatelů, práce s Firebase databází a Google maps. K implementaci aplikace bylo využito vývojového prostředí Android Studio.

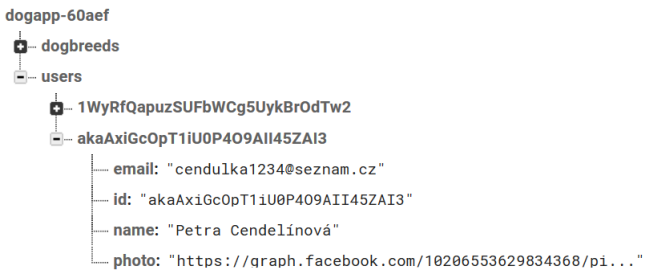
7.1 Přihlášení uživatele

Přihlášení uživatele v aplikaci je možné realizovat dvojím způsobem, a to pomocí registrovaného emailu nebo facebookovým účtem uživatele. V případě přihlášení pomocí emailu je nutná předchozí registrace, při které je nutné získat přihlašovací údaje od uživatele, tj. email a heslo. V případě přihlášení prostřednictvím jiného poskytovale (v tomto případě Facebook) je nutné získat OAuth token. Následně jsou tyto získané údaje poskytnuty Firebase Authentication SDK. Firebase ukládá informace o uživatelích v podobě znázorněné obrázkem 28.



Obrázek 28: Uživatelé ve Firebase
(Převzato z Firebase Console, 2016)

Pro ukládání doplňujících informací (jméno, přijmení, odkaz na stažení fotky) o uživatelích bylo využito Firebase Realtime Database (viz obrázek 29).



Obrázek 29: Ukládání uživatelů ve Firebase Database
(Převzato z Firebase Console, 2016)

Kód 1: Přihlášení prostřednictvím emailu - FirebaseApplication.java

```
public void loginUser(final Context context, String email, String password) {
       // zobrazeni progress dialogu prostrednictvim helper tridy
       DialogHelper.showProgressDialog(context,
           getString(R.string.progress_dialog_title_login));

       // ziskani instance Firebase Auth k prihlaseni registrovaneho uzivatele
           prostrednictvim emailu a hesla
       getFirebaseAuth().signInWithEmailAndPassword(email, password)
              .addOnCompleteListener((Activity) context, new
                  OnCompleteListener<AuthResult>() {
                  @Override
                  public void onComplete(@NonNull Task<AuthResult> task) {
                      if (!task.isSuccessful()) {
                          Log.e(TAG, "signInWithEmail", task.getException());

                          // pokud prihlaseni nebylo uspesne, zobrazi se uzivateli
                              chybova hlaska na zaklade FirebaseAuthExceptions
                          FirebaseExceptionsChecker.checkFirebaseAuth(context,
                              task);

                      } else {
                      // zkontroluje prihlaseni uzivatele a vytvori se Applozic
                          uzivatel potrebny pro chat
                          if (checkUserLogin(context)) {
                              String userId = task.getResult().getUser().getUid();
                              Log.d(TAG, "loginUser: user with id " + userId +
                                  "was logged" );

                              // registrace notifikaci
                              FirebaseMessaging.getInstance().subscribeToTopic(userId);

                              //prechod na hlavni obrazovku aplikace
                              context.startActivity(new Intent(context,
                                  MainActivity.class));
                          }
                      }

                      // skryti progress dialogu
                      DialogHelper.hideProgressDialog();
                  }
              });
}
```

Kód 2: Přihlášení pomocí Facebooku - LoginActivity.java

```
@Override
public void onCreate(Bundle savedInstanceState) {
       // pred pouzitim je nutne inializovat Facebook Sdk
       FacebookSdk.sdkInitialize(getApplicationContext());
       // ziskani instance callback manageru
       callbackManager = CallbackManager.Factory.create();
       // nastaveni opravneni ke cteni informaci o uzivateli
       facebookButton.setReadPermissions("email", "public_profile");
       // registrace callback k ziskani login result
       facebookButton.registerCallback(callbackManager, new
           FacebookCallback<LoginResult>() {
           @Override
           public void onSuccess(LoginResult loginResult) {
               Log.d(TAG, "facebook:onSuccess:" + loginResult);
               // v pripade uspesneho prihlaseni ziska se token
               handleFacebookAccessToken(loginResult.getAccessToken());
           }

           @Override
           public void onCancel() {
               // uzivatel zrusil pozadavek na prihlaseni
               Log.d(TAG, "facebook:onCancel");

           }

           @Override
           public void onError(FacebookException error) {
               Log.d(TAG, "facebook:onError", error);
               // pokud nastala chyba, zobrazi se chybova hlaska
               DialogHelper.showAlertDialog(LoginActivity.this,
                   getString(R.string.occur_error));
           }
       });
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
       // kliknutim na tlacitko, vraci se result, ktery se preposle callback
           manager
       callbackManager.onActivityResult(requestCode, resultCode, data);
}
```

Kód 3: Přihlášení pomocí Facebooku (metoda handleFacebookAccessToken)

```
private void handleFacebookAccessToken(AccessToken token) {
      Log.d(TAG, "handleFacebookAccessToken:" + token);

      // vrati novou instanci AuthCredential na zaklade ziskaneho tokenu
      AuthCredential credential =
          FacebookAuthProvider.getCredential(token.getToken());

      firebaseAuth.signInWithCredential(credential)
             .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                   Log.d(TAG, "signInWithCredential:onComplete:" +
                       task.isSuccessful());
                   if (!task.isSuccessful()) {
                      Log.d(TAG, "signInWithCredential", task.getException());
                      // zobrazeni chybove hlasky
                      DialogHelper.showAlertDialog(LoginActivity.this,
                          getString(R.string.unsuccessful_login));
                   } else {
                      String userId = task.getResult().getUser().getUid();
                      // registrace notifikaci
                      FirebaseMessaging.getInstance().subscribeToTopic(userId);
                      if
                          (!UserUtils.checkIfUserInfoIsSaved(LoginActivity.this,
                          userId)) {
                          // ulozeni informaci o uzivateli do shared
                              preferences pro snadnejsi pristup
                          // uklada se email, jmeno, odkaz ke stazeni fotky
                          UserUtils.storeInfoAboutUser(LoginActivity.this,
                              task.getResult().getUser().getDisplayName(),
                                 task.getResult().getUser(), null);
                      }
                      if (((FirebaseApplication)
                          getApplication()).checkUserLogin(LoginActivity.this))
                          {
                          // presmerovani na hlavni obrazovku
                          startActivity(new Intent(LoginActivity.this,
                              MainActivity.class));
                      }
                   }
                }
             });
   }
```

7.2 Práce s Firebase Realtime Database

Jak bylo možné vidět na obrázku 29, data jsou v databázi ukládana jako jeden JSON objekt. Při práci s tímto objektem je nutné specifikovat uzel, se kterým bude pracováno.

7.2.1 Ukládání dat

Ukládání dat do Firebase databáze bude demonstrováno na příkladu vložení nového psa uživatele. Nejprve bylo nutné vytvořit model Dog s povinným bezparametrickým konstruktorem a veřejnými gettery/settery. Tato třída je potomkem generické třídy GenericModel obsahující jedinou vlastnost, a to ID modelu.

Kód 4: Vytvoření nového psa ve Firebase Realtime Database

```
public String saveNewRecord(GenericModel model, String typeChild) {
       // GenericModel je genericka trida, trida Dog je jejim potomkem
       // vytvoreni id pro noveho psa
       String id = getFirebaseDatabase().child(typeChild).push().getKey();
       model.setId(id);
       // nejprve je nutne ziskat instanci Firebase Database
       // v tomto pripade dogs->userId->dogId->samotny model psa
       getFirebaseDatabase().child(typeChild)
                           .child(getFirebaseUserId())
                           .child(id)
                           .setValue(model);

       return id;
}
// volani teto metody z NewDogActivity.java
((FirebaseApplication) getApplication()).saveNewRecord(createdDog,
    FIREBASE_CHILD_DOGS);
```

7.2.2 Získávání dat

Při získávání dat z databáze je nutné zaregistrovat jeden z možných typů listenerů, který bude poslouchat při změnách v datech na dané pozici. Zmíněný postup bude demonstrován na příkladu získání informací o psích plemenech.

Kód 5: Získávání dat o psích plemenech z Firebase Realtime Database

```
// specifikovani child
valueEventListener =
    databaseReference.child(FIREBASE_CHILD_DOG_BREEDS).addValueEventListener(new
    ValueEventListener() {
              @Override
              public void onDataChange(DataSnapshot dataSnapshot) {
                  for (DataSnapshot postSnapshot : dataSnapshot.getChildren()) {
                      // ziskani objektu DogBreed se vsemi vlastnostmi
                      DogBreed dogBreed = postSnapshot.getValue(DogBreed.class);
```

```
                dogBreedList.add(dogBreed);
            }
            // serazeni podle ceske abecedy
            Collections.sort(dogBreedList, new DogBreedComparator());
            dogBreedAdapter.notifyDataSetChanged();
        }

        @Override
        public void onCancelled(DatabaseError databaseError) {
            Log.d(TAG, "onCancelled", databaseError.toException());
            // chybova hlaska
            DialogHelper.showNoDataRetrieved(getContext());
        }
});
```

7.3 Práce s mapou

Jedním z požadavků na aplikaci byla možnost vyhledávání míst na mapě. K tomuto účelu bylo využito Google Places API. Před samotným použitím mapy bylo nutné vygenerovat klíč k použití Google Maps v2 a jeho vložení do Android manifest.xml. Přidání mapy bylo realizováno následujícími řádky kódu:

```
SupportMapFragment mapFragment = (SupportMapFragment) getChildFragmentManager()
            .findFragmentById(R.id.map);
    mapFragment.getMapAsync(this);
```

Google Places API umožňuje vyhledávání nejbližších míst v lokalitě uživatele dle daného typu. Dle stanovených požadavků na aplikaci v kapitole 4.3 bylo potřeba vyhledat veterináře, chovatelské potřeby a parky aj. Google Places API specifikuje typ pet_store, veterinary_care a park. Ostatní typy míst byly vyhledány podle určitého klíčového slova, např. „psíškolka“ nebo „psíhotel“. Následující místa byla získána za použití knihovny Retrofit.

Využití knihovny Retrofit vyžaduje

- definování modelu – data ve formátu JSON budou deserializována do této třídy
- vytvoření interface specifikující HTTP operace
- vytvoření REST adapter pro danou URL

Zmíněný postup bude vysvětlen na níže uvedeném úryvku kódu.

Kód 6: Získávání dat z Google Places API pomocí knihovny Retrofit

```
/**
 *  1. Vytvoreni modelu Place, PlaceResponse.java
 *  2. Vytvoreni interface
 */
@GET("api/place/nearbysearch/json?sensor=true&key=MY_API_KEY")
Call<PlaceResponse> getNearbyPlaces(@Query("type") String type,
    @Query("location") String location, @Query("radius") int radius);

@GET("api/place/textsearch/json?key=MY\_API\_KEY")
Call<PlaceResponse> getNearbyPlacesByText(@Query("query") String query,
    @Query("location") String location, @Query("radius") int radius);

/**
 * 3. Implementace metody retrieveMapData
 */
public static void retrieveMapData(final MarkerEnum type, boolean
    isSearchedByText, final GoogleMap googleMap, Location currentLocation, final
    Context context) {
       // rozsah vyhledavani
       int PROXIMITY_RADIUS = 50000;
       // vytvoreni REST Adapter pro danou URL
      Interfaces.PlacesInterface service =
          RetrofitGenerator.createService(Interfaces.PlacesInterface.class);

       Call<PlaceResponse> call;
       if (isSearchedByText) {
          // bude vyhledavano podle klicoveho slova
          call = service.getNearbyPlacesByText(type.getType(),
              currentLocation.getLatitude() + "," +
              currentLocation.getLongitude(), PROXIMITY_RADIUS);

       } else {
          // vyhledavani podle typu podporovaneho Google Places API
          call = service.getNearbyPlaces(type.getType(),
              currentLocation.getLatitude() + "," +
              currentLocation.getLongitude(), PROXIMITY_RADIUS);
       }
       // asynchronni volani
       call.enqueue(new Callback<PlaceResponse>() {
          @Override
          public void onResponse(Call<PlaceResponse> call,
              Response<PlaceResponse> response) {
             // deserializace mist
             List<Place> places = response.body().getResults();
             Log.d(TAG, "onResponse: Number of received places " +
                 places.size());
             // vytvoreni markeru na mape
             setMarkers(type, places, googleMap, context);
          }
```

```
            @Override
            public void onFailure(Call<PlaceResponse> call, Throwable t) {
                Log.e(TAG, "onFailure: " + call.toString());
            }
        });
}

/**
 * 4. Volani metody staticke metody v MapFragment
 */
ApiClient.retrieveMapData(MarkerEnum.PET_SHOP, true, map, location,
    getContext());
```

7.4 Vytvoření upomínky s využitím SQLite databáze

Důležitou součástí aplikace je možnost vytvoření nové upomínky, která upozorní uživatele na danou událost ve stanovený čas. Mimo jiné je možné tuto upomínku opakovat každých 5 minut do doby jejího ukončení uživatelem. Uživateli je umožněno vybrat z několika typů události definované ReminderCategoryEnum. Vzhledem charakteru ukládaného objektu bylo zvoleno lokální uložení prostřednictvím SQLite databáze.

Kód 7: Uložení nové upomínky v SQLite databázi

```
private static final String DATABASE_NAME = "ReminderDatabase";
private static final String TABLE_REMINDERS = "ReminderTable";

// jednotlive sloupce
private static final String COLUMN_ID = "id";
private static final String COLUMN_NAME = "title";
private static final String COLUMN_DATE = "date";
private static final String COLUMN_TIME = "time";
private static final String COLUMN_NOTE = "note";
private static final String COLUMN_CATEGORY = "category";

// vytvoreni tabulky v metode onCreate
String CREATE_REMINDERS_TABLE = "CREATE TABLE " + TABLE_REMINDERS +
              "("
              + COLUMN_ID + " INTEGER PRIMARY KEY,"
              + COLUMN_NAME + " TEXT,"
              + COLUMN_DATE + " TEXT,"
              + COLUMN_TIME + " INTEGER,"
              + COLUMN_CATEGORY + " TEXT,"
              + COLUMN_NOTE + " TEXT" + ")";
db.execSQL(CREATE_REMINDERS_TABLE);

// vlozeni nove upominky
public int addReminder(Reminder reminder) {
       SQLiteDatabase db = this.getWritableDatabase();
       ContentValues values = new ContentValues();
```

```
        values.put(COLUMN_NAME, reminder.getName());
        values.put(COLUMN_DATE, reminder.getDate());
        values.put(COLUMN_TIME, reminder.getTime());
        values.put(COLUMN_CATEGORY, reminder.getCategory());
        values.put(COLUMN_NOTE, reminder.getNote());

        long ID = db.insert(TABLE_REMINDERS, null, values);
        db.close();
        return (int) ID;
}
```

Nastavení upomínky v daný čas je realiazováno prostřednictvím AlarmManageru ve třídě ReminderManager.java.

```
// zaslání PendingIntent, ktery se spusti ve stanovenou dobu v~AlarmReceiver
alarmManager.set(AlarmManager.RTC_WAKEUP, when, alarmIntent);
```

Samotné zobrazení upomínky je realizováno ve třídě AlarmReceiver, která je potomkem abstraktní třídy BroadcastReceiver. V metodě *onReceive* je získána konkrétní upomínka z databáze na základě ID. Následně je vytvořena notifikace k dané upomínce. Uživateli je umožněno měnit nastavení notifikací, tj. zvuk, vibrace či zablokování notifikace.

Kód 8: Upozornění na novou upomínku - AlarmReceiver.java

```
@Override
public void onReceive(Context context, Intent intent) {

    Log.d(TAG, "onReceive");
    // ziskani ID upominky
    int reminderId = Integer.parseInt(intent.getStringExtra(REMINDER_ID));
    ReminderDatabase reminderDatabase = new ReminderDatabase(context);
    // ziskani dane upominky podle ID
    Reminder reminder = reminderDatabase.getReminder(reminderId);

    NotificationManager notificationManager =
        (NotificationManager)context.getSystemService(Context.NOTIFICATION_SERVICE);
    // vytvoreni upominky pomoci helper tridy a jeji upozorneni uzivateli
    notificationManager.notify(reminderId,
        NotificationHelper.createReminderNotification(context, reminder));

}
```

8 Testování aplikace

Následující kapitola se zabývá testováním samotné aplikace. Tabulka 15 zahrnuje všechna testovací zařízení, která byla v průběhu implementace a následného testování použita.

Tabulka 15: Testovací zařízení a jejich konfigurace

Název zařízení	Úhlopříčka displeje (palce)	Rozlišení displeje (px)	Verze OS Android
Lenovo Phab	6.98”	720 x 1280 px	5.1
Nexus 6	5.96”	1440 x 2560 px	5.1
Pixel C	9.94”	2560 x 1800 px	5.1
Samsung Galaxy J5	5.0”	720 x 1280 px	6.0.1
Sony Xperia Z3 Compact	4.6”	720 x 1280 px	6.0.1

Na základě testovacího scénáře z kapitoly 6 bylo provedeno testování na výše uvedených zařízení. Provedené testování na mobilních zařízení bylo uspokojující, v případě tabletu Pixel C bude potřeba dalšího přizpůsobení layoutu před zveřejněním aplikace.

9 Závěr a diskuze

Cílem práce bylo vyvinout mobilní aplikaci pro platformu Android, která je zaměřena na majitele psů. Aplikace slouží jako sociální síť, která usnadňuje vzájemnou komunikaci majitelů psů. Před samotnou implementací aplikace bylo nutné se seznámit se základními postupy při vývoji OS Android. Posléze byly zanalyzovány existující mobilní aplikace s obdobným zaměřením. Předmětem tohoto zkoumání bylo české i zahraniční prostředí. Překvapujícím faktem této analýzy byla neexistující komplexní sociální aplikace pro majitelů psů v České republice. Následně byla provedena analýza nejvhodnějšího cloudového řešení dle stanovených kritérií. Jako nejlépe splňující tyto kritéria bylo zvoleno cloudové řešení Firebase od firmy Google. Provedené analýzy umožnily stanovení funkčních a nefunkčních požadavků na aplikaci. Nefunkční požadavky představují možná budoucí zlepšení, kterými by bylo vhodné se dále zabývat. Lze například zmínit monitorování pohybu psů pomocí Bluetooth obojků. Jako zajímavá funkcionalita se jeví možnost informování o ztraceném psovi a výměna dat s útulky.

Při tvorbě aplikace bylo nutné vyřešit několik hlavních problémů. Jednalo se o správné navržení schématu NoSQL databáze zabraňující duplikaci dat. Další problém se vyskytl při výběru způsobu ukládání fotografií v aplikaci. Nabízelo se několik možností. První otázka zněla, zda-li fotografie uchovávat lokálně nebo pomocí Firebase. Firebase umožnuje ukládat fotografie ve formátu Base64 nebo ve Firebase Storage. Ukládání obrázků přímo v databázi se jeví jako vhodnější varianta vzhledem k rychlejšímu získávání dat. Výjimku v ukládání fotografií tvoří profilová fotografie uživatele, kterou je výhodnější uložit pomocí Firebase Storage kvůli získání odkazu pro další práci. Z důvodu efektivnějšího přístupu se tyto obrázky (konkrétně fotografie vzpomínek, psa, psích kamarádů) také ukládají lokálně. V případě ztráty telefonu nebo vymazání aplikace budou staženy z databáze.

Při tvorbě aplikace bylo nutné zajistit správu uživatele, která zahrnuje registraci, přihlášení a obnovu zapomenutého hesla. Pro tyto účely se jevilo vhodné zvolit cloudové řešení. Pro přístup do aplikace je možné se registrovat emailem a heslem nebo se přihlásit facebookým účtem. Po přihlášení uživatel může vytvořit a spravovat profil svého psa.

V aplikaci je kladen důraz na usnadnění komunikace mezi uživateli. První formou komunikace mezi uživateli je zabudovaný chat vytvořený prostřednictvím Applozic SDK. Výhodou tohoto přístupu je hotové fungující řešení s možností vlastního přízpůsobení. Problémová situace může nastat při zveřejnění aplikace, jelikož existují omezení funkcionality určené cenou. V budoucnu bude nutné řešit ekonomickou otázku týkající se výhodnosti zvoleného řešení a to buď implementace vlastního chatu nebo nákup zpoplatněné verze. Druhou možností je přizvat dalšího majitele psa ke společnému venčení prostřednictvím okamžité notifikace. Poslední možností je vyžádání pozornosti uživatele zasláním notifikace „Štěknout“. Implementace zmíněných notifikací byla provedena pomocí Firebase Cloud Messaging.

Aplikace umožňuje získat potřebné informace o nejbližším veterináři, chovatel-

ských potřebách, psích salónech a dalších místech na mapě. Při implementaci se využilo Google Maps v2 a k získání těchto míst Google Places API. Existují zde omezení ve vyhledávání míst na základě aktuální polohy uživatele v daném okruhu. Získaná data nejsou zcela aktuální a kompletní. Při použití Google Places API nastaly určité komplikace s omezeným přístupem bezplatného profilu. Jednalo se o překročený počet žádostí na server za sekundu. V budoucnu bude nutné tento problém vyřešit placeným profilem nebo získáním těchto údajů od jiného poskytovale.

Další užitečnou přidanou funkcionalitou je připomínání důležitých termínů – návštěva veterináře či podání léku psovi.

Další požadavek na aplikaci představoval schopnost poskytnutí informací o dalších existujících plemenech uživateli. V řešení bylo poskytnutí dat od portálu Hafici.cz, bohužel nedošlo k vzájemné dohodě. V současné době je k dispozici 25 plemen k nahlédnutí. Při zveřejnění aplikace bude nutné tento požadavek dále řešit.

Stanovený cíl práce byl naplněn, byla vytvořena Android aplikace pro majitele psů. V současném stavu se nachází aplikace stále v testovacím režimu. Ráda bych tuto aplikaci dále rozvíjela, mám řadu nápadů na její vylepšení. Z tohoto důvodu není mým cílem zveřejnění na Google play v dohledné době. Mimo jiné bych ráda vytvořila verzi pro iOS a webové rozhraní.

10 Literatura

ALLEN, G. *Android 4: průvodce programováním mobilních aplikací.* 1. vyd. Brno: Computer Press, 2013. 656 s. ISBN 978-80-251-3782-6.

ANDROID DEVELOPERS. *Activity* [online]. 2015a [cit. 2015-12-07]. Dostupné na: http://developer.android.com/reference/android/app/Activity.html.

ANDROID DEVELOPERS. *Platfrom Architecture* [online]. 2016a [cit. 2016-10-21]. Dostupné na: https://developer.android.com/guide/platform/index.html.

ANDROID DEVELOPERS. *Platfrom Architecture – Android Stacks* [online]. 2016b [cit. 2016-10-21].
Dostupné na: https://developer.android.com/guide/platform/images/android-stack_2x.png.

ANDROID DEVELOPERS. *Platform Versions* [online]. 2016c [cit. 2016-27-12]. Dostupné na: https://developer.android.com/about/dashboards/index.html.

ANDROID DEVELOPERS. *State diagram for an Android Activity Lifecycle* [online]. 2015b [cit. 2015-12-07].
Dostupné na: http://developer.android.com/activitylifecycle.png.

ARLOW, J., NEUSTADT I. *UML a unifikovaný proces vývoje aplikací: průvodce analýzou a návrhem objektově orientovaného softwaru.* Brno: Computer Press, 2003. 387 s. ISBN 80-7226-947-X.

BACKENDLESS. *Backendless* [online]. ©2013-2016 [cit. 2016-10-14]. Dostupné na: https://backendless.com/.

BOHN, D. *Material world: how Google discovered what software is made of* [online]. 2014 [cit. 2016-09-30].
Dostupné na: http://www.theverge.com/2014/6/27/5849272/material-world-how-google-discovered-what-software-is-made-of.

FIREBASE. *Features* [online]. 2016 [cit. 2016-10-14]. Dostupné na: https://firebase.google.com/features/.

FIREBASE. *Firebase Console* [online]. 2016 [cit. 2016-12-22]. Dostupné na: https://console.firebase.google.com.

FLATICON. *Free vector icons* [online]. 2016 [cit. 2016-09-30]. Dostupné na: http://www.flaticon.com/.

FOWLER, M. *Destilované UML.* 1. vyd. Praha: Grada, 2009. 176 s. Knihovna programátora. ISBN 978-80-247-2062-3.

FULCHER, R. *DesignBytes: Paper and Ink: The Materials that Matter* [online video]. 2014 [cit. 2016-09-30].

Dostupné na: https://www.youtube.com/watch?v=YaG_ljfzeUw.

GOOGLE. *Material Design : Components* [online]. 2016a [cit. 2016-09-30]. Dostupné na: https://material.google.com/components/.

GOOGLE. *Material design : Components – BottomSheets* [online]. 2016b [cit. 2016-09-30]. Dostupné na: https://material.io/guidelines/components/bottom-sheets.html.

GOOGLE. *Material design : Components – Cards* [online]. 2016c [cit. 2016-09-30]. Dostupné na: https://material.io/guidelines/components/cards.html.

GOOGLE. *Material design : Components – Buttons: Floating Action Button* [online]. 2016d [cit. 2016-09-30]. Dostupné na: https://material.io/guidelines/components/buttons-floating-action-button.html.

GOOGLE. *Material design : Components – Lists* [online]. 2016e [cit. 2016-09-30]. Dostupné na: https://material.io/guidelines/components/lists.html.

GOOGLE. *Material Design : Introduction - Goals* [online]. 2016f [cit. 2016-09-30]. Dostupné na: https://material.io/guidelines/#introduction-goals.

GOOGLE. *GOOGLE PLAY : 11Pets: Pet Care – Android Apps on Google Play* [online]. 2016g [cit. 2016-27-12]. Dostupné na: https://play.google.com/store/apps/details?id=com.m11pets.elevenpets.

GOOGLE. *GOOGLE PLAY : Dogalize - Pet Social Network – Android Apps on Google Play* [online]. 2016h [cit. 2016-27-12]. Dostupné na: https://play.google.com/store/apps/details?id=com.dogalize.

GOOGLE. *GOOGLE PLAY : Mookie - Dog Diary – Android Apps on Google Play* [online]. 2016ch [cit. 2016-27-12]. Dostupné na: https://play.google.com/store/apps/details?id=com.fortmobile.mookieapp.

GOOGLE. *GOOGLE PLAY : Můj pes – Android Apps on Google Play* [online]. 2016i [cit. 2016-27-12]. Dostupné na: https://play.google.com/store/apps/details?id=cz.simopt.get2knowpsi.

GOOGLE. *GOOGLE PLAY : Po stopě – Android Apps on Google Play* [online]. 2016j [cit. 2016-27-12]. Dostupné na: https://play.google.com/store/apps/details?id=eu.mobera.flowmedia.brit&hl=en.

GOOGLE. *GOOGLE PLAY : Psí Detektiv - pátrací systém – Android Apps on Google Play* [online]. 2016k [cit. 2016-27-12]. Dostupné na: https://play.google.com/store/apps/details?id=com.ps.detektiv.

IDC. *Smartphone OS Market Share* [online]. 2015 [cit. 2015-12-07]. Dostupné na: http://www.idc.com/prodserv/smartphone-os-market-share.jsp.

JUSTINMIND. *Overview* [online]. © 2014 - 2016 [cit. 2016-09-30]. Dostupné na: https://www.justinmind.com/overview.

KINVEY. *How Kinvey helps you....* [online]. 2016 [cit. 2016-10-15]. Dostupné na: https://www.kinvey.com/platform/.

LACKER, K. *Moving On* [online]. 2016 [cit. 2016-10-10]. Dostupné na: http://blog.parse.com/announcements/moving-on/.

LACKO, L. *Vývoj aplikací pro Android.* 1. vyd. Brno: Computer Press, 2015. 472 s. ISBN 978-80-251-4347-6.

MANAGEMENT MANIA. *Cloud computing* [online]. 2016 [cit. 2016-10-09]. Dostupné na: https://managementmania.com/cs/cloud-computing.

MÁJSKÝ, M. *Analýza současných řešení Backend-as-a-Service pro vývoj mobilních a webových aplikací.* [diplomová práce] Praha: České vysoké učení technické v Praze, 2016. Fakulta informačních technologií, vedoucí práce Ing. Vladislav Skoumal.

METHA, N. *Cloud Deployment Models* [online]. 2012 [cit. 2016-10-10]. Dostupné na: http://cloudtweaks.com/2012/07/4-primary-cloud-deployment-models/.

NURIK, R. *DesignBytes: Density-independent Pixels* [online]. 2013 [cit. 2016-10-22]. Dostupné na: https://developer.android.com/training/multiscreen/screendensities.html.

OPTIMIZELY. *A/B Testing* [online]. 2016 [cit. 2016-10-14]. Dostupné na: https://www.optimizely.com/ab-testing/.

PILONE, D., PITMAN, N. *UML 2.0 in a Nutshell.* Vyd. 1. Sebastopol, California: O'Reilly Media, 2005. 234 s. ISBN 0-596-00795-7.

SECURITY MAGAZÍN. *Inteligentní psí obojek pro sledování pohybu a monitoring zdravotního stavu* [online]. 2015 [cit. 2016-10-02]. Dostupné na: http://www.securitymagazin.cz/technologie/psi-obojek-pro-sledovani-pohybu-a-monitoring-zdravotniho-stavu-1404043577.html.

SPOIALA, C. *Cloud offering: Comparison between IaaS, PaaS, SaaS, BaaS* [online]. 2015 [cit. 2016-10-09]. Dostupné na: https://assist-software.net/blog/cloud-offering-comparison-between-iaas-paas-saas-baas.

STATSMONKEY. *Mobile Facebook, Twitter, Social Media Usage Statistics in Czech Republic* [online]. 2015 [cit. 2016-10-02]. Dostupné na: https://www.statsmonkey.com/table/21332-czech-republic-mobile-social-media-usage-statistics-2015.php.

THORNSBY, J. *Android UI Design.* Birmingham: Packt Publishing, 2016. 373 s. ISBN 978-1-78588-742-0.
Dostupné na: http://it-ebooks.info/book/1464606113/.

UJBÁNYAI, M. *Programujeme pro Android.* Vyd. 1. Praha: Grada, 2012. 187 s. ISBN 978-80-247-3995-3.

VINCOS BLOG. *World Map of Social Networks* [online]. 2016 [cit. 2016-10-02]. Dostupné na: http://vincos.it/world-map-of-social-networks/.

VITAS, M. *ART vs Dalvik - introducing the new Android runtime in KitKat* [online]. 2013 [cit. 2016-10-21]. Dostupné na: https://infinum.co/the-capsized-eight/articles/art-vs-dalvik-introducing-the-new-android-runtime-in-kit-kat.

WOODFORD, C. *Cloud computing* [online]. 2016 [cit. 2016-10-09]. Dostupné na: http://www.explainthatstuff.com/cloud-computing-introduction.html.

W3RESOURCE. *NoSQL* [online]. [cit. 2016-10-15].
Dostupné na: http://www.w3resource.com/mongodb/nosql.php.

Přílohy

A Zdrojové kódy

K zobrazení seznamu dat byl využit RecyclerView s návrhovým vzorem ViewHolder. Vzhledem k opětovnému použití implementace byla vytvořena generická třída BaseAdapter dědící třídu RecyclerView.Adapter a BaseViewHolder dědící třídu RecyclerView.ViewHolder.

Kód 9: Generické třídy pro zobrazení dat

```
// genericka trida pro view holders
public abstract class BaseViewHolder<T> extends RecyclerView.ViewHolder
    implements View.OnClickListener {

    private T item;
    private Interfaces.ItemClickCallback itemClickCallback;

    protected BaseViewHolder(ViewGroup parent, @LayoutRes int itemLayoutId,
        Interfaces.ItemClickCallback itemClickCallback) {
        // ziskani layoutu pro radek
        super(LayoutInflater.from(parent.getContext()).inflate(itemLayoutId,
            parent, false));

        this.itemClickCallback = itemClickCallback;
        // findViewById knihovnou Butterknife
        ButterKnife.bind(this, itemView);
        // registrace onClickListener
        itemView.setOnClickListener(this);

    }

    public final void performBind(T item, int position) {
        this.item = item;
        // zobrazeni dat na urcite pozici
        onBind(item, position);
    }

    protected abstract void onBind(T item, int position);


    @Override
    public void onClick(View view) {
        // registrace interface zajistujici kliknuti na item
        itemClickCallback.onItemClick(getAdapterPosition());
    }
}

// genericka trida Adapter
public abstract class BaseAdapter<T, VH extends BaseViewHolder<T>> extends
    RecyclerView.Adapter<VH> {
    private List<T> genericList = new ArrayList<>();
    private Interfaces.ItemClickCallback itemClickCallback;
```

```
    public BaseAdapter(List<T> genericList) {
        this.genericList = genericList;
    }
    // getter
    public Interfaces.ItemClickCallback getItemClickCallback() {
        return itemClickCallback;
    }
    // setter
    public void setItemClickCallback(Interfaces.ItemClickCallback
         itemClickCallback) {
        this.itemClickCallback = itemClickCallback;
    }

    public List<T> getGenericList() {
        return ((genericList != null) ? genericList : null);
    }

    // getter
    public T getItem(int pos) {
        return ((genericList != null && pos < genericList.size()) ?
            genericList.get(pos) : null);
    }

    // setter
    public void setGenericList(List<T> genericList) {
        this.genericList = genericList;
    }

    @Override
    public final void onBindViewHolder(VH vh, int position) {
        T item = genericList.get(position);
        // metoda z BaseViewHolder zajistujici zobrazeni dat na urcite pozici
        vh.performBind(item, position);
    }

    @Override
    public int getItemCount() {
        return ((genericList != null ? genericList.size() : 0));
    }

}
```

Následující kód demonstruje konkrétní použití BaseAdapter a BaseViewHolder ve třídě DogDiaryFragment. Cílem je zobrazit existující vzpomínky uživatele.

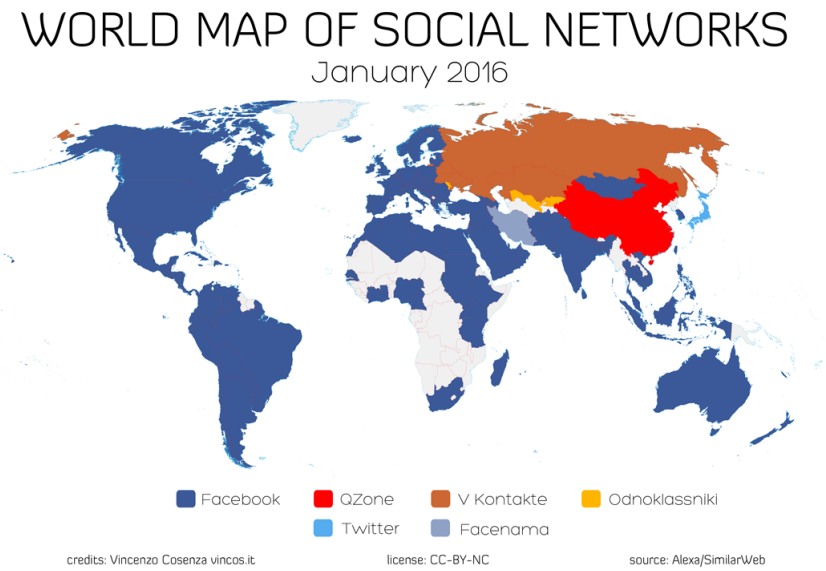
Kód 10: Generické třídy pro zobrazení dat

```
// pouziti genericke tridy u seznamu vzpominek
public class MemoryViewHolder extends BaseViewHolder<Memory> {
    Activity activity;
    // knihovna ButterKnife zajistiji pouzitim techto anotaci findViewById
    @BindView(R.id.memory_photo)
    ImageView memoryPhoto;
    @BindView(R.id.memory_title)
    TextView memoryTitle;

    public MemoryViewHolder(Activity activity, ViewGroup parent,
        Interfaces.ItemClickCallback itemClickCallback) {
        super(parent, R.layout.memory_card_view_item, itemClickCallback);
        this.activity = activity;
    }

    @Override
    protected void onBind(Memory item, int position) {
        // zobrazeni dat pro dany radek na urcite pozici
        if (item.getDate() != null) {
            memoryTitle.setText(item.getDate() + " " + item.getName());
        } else {
            memoryTitle.setText(item.getName());
        }

        // ziskani cesty k fotce vzpominky v external storage
        String imagePath = item.getUserId() + STORAGE_IMAGES_MEMORIES_PATH +
            item.getId();
        if (item.isUpdatedPhoto()) {
            // nastaveni fotky, pokud existuje, bude nastavena z external storage,
                v opacnem pripade dojde ke stazeni z Firebase
            // pokud nebyla fotka nastavena uzivatelem, vlozi se defaultni obrazek
            ImageUtils.setImageView(activity, imagePath, item.getPhoto(),
                memoryPhoto, true, SET_MEMORY_PHOTO);
            // pokud byla fotka vzpominka zmenena, zmeni se i v databazi
            // uzivatel bude mit k dispozici pristup k fotkam i po odinstalaci
                aplikace nebo ztrate dat v mobilnim telefonu
            ((FirebaseApplication)
                activity.getApplication()).saveOrEditSingleValue(false,
                item.getDogId(), FIREBASE_CHILD_MEMORIES, item.getId() + "/" +
                UPDATED_PHOTO);
        } else {
            // parametr false - fotka nebyla updatovana, neni potreba zrusit
                cachovani v knihovne Picasso
            // SET_MEMORY_PHOTO - definuje typ fotky
            ImageUtils.setImageView(activity, imagePath, item.getPhoto(),
                memoryPhoto, false, SET_MEMORY_PHOTO);
        }
```

```
    }
}

// pouziti genericke tridy v MemoryAdapter
public class MemoryAdapter extends BaseAdapter<Memory, MemoryViewHolder> {
    private Activity activity;
    public MemoryAdapter(Activity activity, List<Memory> genericList) {
        super(genericList);
        this.activity = activity;
    }


    @Override
    public MemoryViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        // vytvoreni ViewHolder
        return new MemoryViewHolder(activity, parent, getItemClickCallback());
    }
}

// pouziti MemoryAdapter ve tride DogDiaryFragment.java
memoryAdapter = new MemoryAdapter(getActivity(), memoryList);
memoryAdapter.setItemClickCallback(this);
recyclerView.setAdapter(memoryAdapter);
```

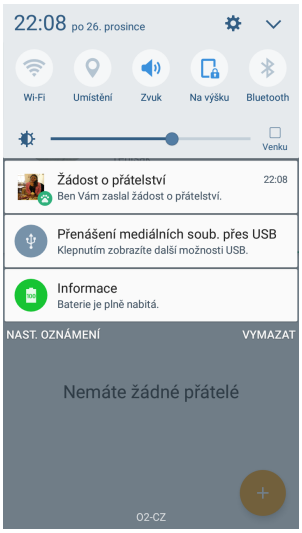
B Analýza

Následující obrázek demonostruje mapu použití sociálních sítí na celém světě v únoru roku 2016. Je zřejmé, že nejpoužívanější sociální sítí v České republice je Facebook.

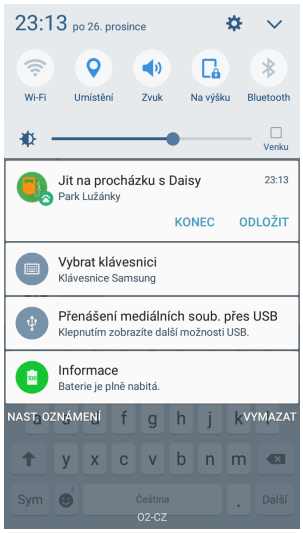


Obrázek 30: Použití jednotlivých typů sociálních sítí na celém světě

C Ukázky obrazovek z aplikace

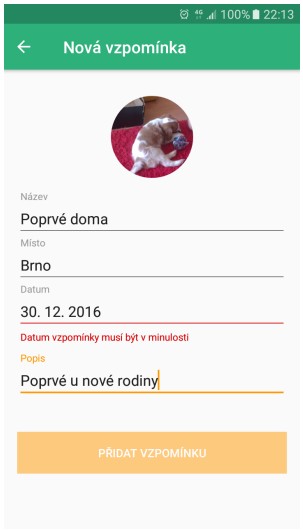


Obrázek 31: Nová žádost o přátelství



Obrázek 32: Upomínka

Obrázek 33: Mapa s místy



Obrázek 34: Validace vstupů

D Elektronické zdroje

V elektronické podobě jsou k této práci přiloženy následující soubory:

- zdrojové soubory aplikace
- APK soubor
- interaktivní prototyp aplikace
- scénaře k zbývajícím případům užití
- ukázky obrazovek z aplikace
- ERD