VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MOBILNÍ APLIKACE PRO NOTIFIKACE PŘI DOSAŽENÍ MÍST

MOBILE APP FOR NOTIFYING ABOUT REACHING A LOCATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **KLÁRA UNGROVÁ**
AUTHOR

VEDOUCÍ PRÁCE **prof. Ing. ADAM HEROUT, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání bakalářské práce

Studentka: **Ungrová Klára**
Program: Informační technologie
Název: **Mobilní aplikace pro notifikace při dosažení míst**
Mobile App for Notifying about Reaching a Location
Kategorie: Uživatelská rozhraní

Zadání:

1. Vyhledejte a analyzujte existující aplikace pro sdílení polohy a pro zaslání zprávy při dosažení cíle.
2. Seznamte se s problematikou vývoje mobilních aplikací.
3. Navrhněte aplikaci, která uživateli usnadní odeslání zprávy po dosažení zadaného cíle. Zaměřte se na snadnost použití, celkovou uživatelskou zkušenost a na malou zátěž na straně uživatele.
4. Implementujte navrženou aplikaci.
5. Testujte vytvořenou aplikaci v provozu a ve vhodné skupině testerů. Iterativně vylepšujte uživatelské rozhraní i funkčnost aplikace.
6. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- Ľuboslav Lacko: Vývoj aplikací pro Android ; Brno : Computer Press,2015
- Owens, M.: The Definitive Guide to SQLite. Apress, 2006
- Bill Phillips, Chris Stewart, Kristin Marsicano: Android programming: the Big nerd ranch guide. Third edition. Atlanta, GA: Big Nerd Ranch, 2017
- César Bejarano: Human Centered Design & The 6 Fundamental Principles of Interaction Between Products and Users, Medium.com, 18.6.2019

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3, značné rozpracování bodů 4 a 5.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Herout Adam, prof. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 1. listopadu 2019

Abstrakt

Cílem této práce je vytvoření jednoduché aplikace pro operační systém *Android*, která bude zasílat notifikace při vstupu či výstupu z uživatelem zvolené zóny a celkově ulehčovat proces informování dalších osob. Detekce vstupu/výstupu ze zóny je založena na principu *geofencingu*. Pro trvalé uložení jednotlivých zón je použita *SQLite* databáze. Při překročení hranice zóny obdrží uživatel notifikaci, jejímž stisknutím je možné odeslat zprávu druhé osobě. Tuto zprávu si uživatel může předvolit pro každou zónu zvlášť (jak pro vstup, tak výstup). Zónu je možné kdykoliv aktivovat/deaktivovat. Uživateli je umožněno grafické zobrazení zón na mapě. Aplikace též nabízí odeslání zkušební notifikace, na které je názorně vysvětleno, jaké funkce poskytuje. Služba, průběžně získávající polohu, běží i po odstranění aplikace ze seznamu běžících aplikací. Pro úplné vypnutí aplikace je zapotřebí zmáčknout vypínací tlačítko v aplikaci. Práce se zabývá návrhem, implementací a následným testováním vyvíjené mobilní aplikace. Finální verze aplikace je publikována v obchodě *Google Play*.

Abstract

The aim of this bachelor thesis is to create a simple *Android* application which will be able to send notifications when entering or exiting a zone of user's interest and will overall ease the process of informing other people. Detection of entering/exiting zone is based on the principles of *geofencing*. *SQLite* database is used to save user's zones of interest (and details about them) persistently. When crossing the boundaries of a zone of interest, a user receives a notification – by clicking on it a user has the opportunity to send the message to other person. The message to be sent can be preset for every zone (both for entering and exiting). It is possible to activate/deactivate the monitoring of any zone any time. A user can display zones on the map. There is a possibility to send a trial explanatory notification for user to better understand its functionality. The *foreground service* which takes care of retrieving up-to-date location is running even if the app is swiped away from running applications. To kill the application, the „switch off" button in the app needs to be pressed. The main focus of this work is design, implementation and testing of the mobile application. The final version has been published to *Google Play* store.

Klíčová slova

mobilní aplikace, Android, Android Studio, Kotlin, SQLite, notifikace, mapa, lokace, smartphone, geofencing, GUI

Keywords

mobile app, Android, Android Studio, Kotlin, SQLite, notification, map, location, smartphone, geofencing, GUI

Citace

UNGROVÁ, Klára. *Mobilní aplikace pro notifikace při dosažení míst*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, Ph.D.

Mobilní aplikace pro notifikace při dosažení míst

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana profesora Adama Herouta. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

.....
Klára Ungrová
27. května 2020

Poděkování

V první řadě bych ráda poděkovala panu profesorovi Adamu Heroutovi, který mi pomohl s vyjasněním představ o kýženém výsledku bakalářské práce a realistickými připomínkami během vývoje mi ukazoval správnou cestu. Také bych chtěla poděkovat všem mým blízkým, kteří respektovali zavřené dveře mého pokoje, nejen po dobu psaní bakalářské práce, ale po celou dobu studia.

Obsah

Kapitola 1

Úvod

Na trhu v současnosti existuje již plno aplikací zabývajících se sledováním polohy dalších osob – ať už kvůli marketingovým technikám či z osobních důvodů. Bohužel se většinově jedná o aplikace, které sdílí naši polohu s ostatními uživateli nepřetržitě. Cílem této práce je vyvinutí aplikace, která bude fungovat jako pouhé připomenutí uživateli, že může sdílet místo svého výskytu s ostatními a následně usnadnit proces ohlášení.

Podnětem k vytvoření aplikace byla sada mých vlastních zkušeností, ve kterých jsem byla opakovaně vystavována zbytečným konfliktům, vzniklým pouhým nedorozuměním. Důvodem je čistá lidská zapomnětlivost. Lidé v průběhu dne migrují z místa na místo. A je pravděpodobné, že někteří z nich dostanou za úkol ozvat se, až v pořádku dorazí do cíle. Problém často nastává v situaci, kdy jedinec do cíle dorazí, ale již si nevzpomene poslat čekající osobě potvrzující SMS. Aplikace byla vyvinuta za účelem eliminace zmíněného problému. Představuje ulehčení způsobu ohlašování, ale zároveň nejde o zcela bezmyšlenkovitý a neosobní proces – aplikace totiž stále vyžaduje, aby si uživatel na osobu, kterou chce informovat, vzpomněl.

Používání[1] aplikace[2] je jednoduché – uživatel si předem zvolí lokaci, do které míří a při vstupu do zvolené oblasti nebo výstupu z ní obdrží notifikaci. Notifikaci stačí jednoduše stisknout a uživateli se zobrazí nabídka možných způsobů odeslání zprávy. Text zprávy si uživatel volí pro každou zónu zvlášť (jak pro vstup, tak pro výstup). Pokud tak neučiní, je text zprávy nastaven implicitně na text „Jsem OK“, který je zároveň názvem aplikace.

Následující kapitoly práce popisují tvorbu aplikace s názvem *Jsem OK* pro operační systém *Android*, psané v programovacím jazyce *Kotlin*. Technická zpráva začíná zmíněním motivace k vytvoření aplikace, jejím účelem a porovnáním s podobnými, již existujícími aplikacemi. Představení cílové skupiny, společně s příklady užití, je zakončením druhé kapitoly. V další kapitole jsou popsány technologie, použité pro tvorbu aplikace, například programovací jazyk *Kotlin* či vývojové prostředí *Android Studio*. Metodě *geofencingu* je, z důvodu náročnosti, věnována čtvrtá kapitola. Zmiňuje jak základní principy této metody, tak samotnou implementaci jednotlivých částí. Následuje kapitola, zabývající se popisem návrhu aplikace – od prvotních představ, až k finální podobě. Předposlední kapitola zahrnuje popis důležitých částí aplikace a jejich fungování. Poslední kapitola popisuje proces zveřejnění aplikace, její testování a návrhy na další možný vývoj.

[1] Video, vysvětlující používání aplikace *Jsem OK* – https://youtu.be/Vxb2eV3YLs0

[2] Aplikace *Jsem OK* v obchodě *Google Play* – https://play.google.com/store/apps/details?id=com.jsemok

Kapitola 2

Kontext mobilní aplikace pro notifikace při dosažení míst

Tato kapitola je složena ze čtyř částí. V první je popsán smysl a motivace k vytvoření aplikace *Jsem OK*. V následující části je vysvětleno, na jakém principu aplikace funguje a jak ji správně používat. Ve třetí části je zmíněn dílčí výčet podobných aplikací a rozdíly mezi nimi. V poslední části je poté objasněno, jak zhruba vypadá cílová skupina a je uvedeno pár příkladů, ve kterých by mohlo být vhodné aplikaci použít.

2.1 Motivace k vytvoření aplikace

Jako motivace k vytvoření aplikace posloužily mé vlastní zkušenosti. Nespočetněkrát jsem se ocitla v nepříjemné situaci, kdy jsem jela na výlet s kamarády, a po příjezdu do cíle jsem zapomněla informovat rodiče. A to i přesto, že mi to bylo důrazně kladeno na srdce při každém odjezdu. Člověk se snaží udržet v hlavě spoustu informací, tudíž zapomenutí na jednu z mnoha není nic neobvyklého.

Navíc představa, že bych aplikací mohla pomoct i ostatním lidem, ocitajících se v podobných situacích s tímto problémem, byla jednoznačně motivující.

Aplikace podobného rázu se na trhu již vyskytovaly, viz kapitola 2.3. Já jsem ovšem chtěla klást důraz na jiné aspekty.

2.2 Princip a použití aplikace

Aplikace funguje na jednoduchém principu. Oproti ostatním, již existujícím aplikacím podobného rázu, uživatel není neustále pod kontrolou. Záleží pouze na uživateli, zda se rozhodne o své poloze ostatní informovat. V závislosti na předchozím nastavení aplikace zašle uživateli notifikaci v momentě, kdy je vhodné dát ostatním vědět – neodesílá ovšem automaticky zprávu s informacemi o poloze.

Velký důraz byl kladen na efektivitu aplikace. Kvůli snaze snížit spotřebu baterie, dat či paměti je využíváno metody *geofencingu*. Fungování *geofencingu* je později detailněji popsáno v kapitole 4 a konkrétní implementace jednotlivých částí metody je vysvětlena v podkapitole 6.3.

Na základě předpokladu, že uživatelé budou převážně mladšího věku (viz 2.4), byla vyvinuta aplikace tak, aby k jejímu fungování nebylo nezbytně potřebné internetové připojení. Byl zvažován fakt, že ne všechny malé děti mají přístup k datovému připojení. Pro správné

fungování aplikace je nutné mít zapnuté polohové služby (*GPS*). Dále pak už stačí jen nastavit adresu zóny (zadáním adresy monitorované oblasti a případnou specifikací adresy popotažením špendlíku na vykreslené mapě) a specifikovat její vlastnosti popsané v seznamu, který se nachází na konci této podkapitoly (viz 2.2). Nově vytvořená oblast zájmu je následně přidána na seznam k ostatním zónám – ten je zobrazen při každém znovuotevření aplikace. Seznam řadí na první pozici zóny posledně přidané a znovu aktivované. Podle časového razítka jsou řazeny i zóny deaktivované (nejpozději aktivovaná je umístěna hned za aktivní oblasti). Oblast zájmu je možné jednoduše aktivovat či deaktivovat stiskem tlačítka, nacházejícího se u každé zóny zvlášť. Samozřejmostí je také možnost editace či úplné smazání oblasti zájmu.

Nápad možnosti zaslání ukázkové notifikace přišel ze strany testujícího uživatele a byl následně implementován. Uživatel má také možnost jednoduchého zobrazení jednotlivých zón na mapě – vykresleny jsou oblasti spolu s kruhovým vyznačením velikosti zón. Po kliknutí na špendlík zóny se zobrazí informace zahrnující přezdívku oblasti a její adresu. Hlavní menu taktéž zahrnuje nápovědu popisem nejdůležitějších funkcí aplikace a s příklady použití.

Aplikace běží na pozadí po celou dobu, od doby spuštění, až do manuálního vypnutí vypínačem v aplikaci. Určování polohy v aplikaci lze obnovit znovuotevřením aplikace či zmáčknutím stejného vypínače.

Při zakládání nové zóny je potřeba specifikovat následující parametry:

1. **Přezdívka zóny** – slouží pro pojmenování jednoduchou identifikaci zóny. Tato přezdívka musí být unikátní.

2. **Velikost poloměru zóny** – představuje přibližnou vzdálenost od středu zóny, ve které bude uživatel upozorněn na vstup či výstup. Hodnotu je možné volit v rozmezí 70 – 1000 m.

3. **Vstup/výstup** – uživatel musí určit, zda si přeje být u této konkrétní oblasti upozorněn na vstup do zóny, výstup z ní či obojí. Provádí se zaškrtnutím políčka (minimálně jedno, maximálně obě).

4. **Přednastavení zpráv** – zadaný text bude obsahem zprávy, odesílané z aplikace po kliknutí na příchozí notifikaci. Je možné připravit si zprávu zvlášť pro vstup do zóny, i výstup z ní. Pokud nebudou textová pole vyplněna, bude pro zprávu použit implicitní text, nesoucí název aplikace – „Jsem OK“.

2.3 Podobné existující aplikace

Jednou z důležitých částí před samotným návrhem produktu je průzkum trhu a vyhledání typově podobných aplikací. V ideálním případě je aplikace jedinou svého druhu, pokud existuje již více podobných produktů, měla by být nově vytvářená aplikace minimálně v jistých prvcích inovativní. Hlavními indikátory úspěchu je chybějící druh produktu na trhu a celková žádanost mezi uživateli.

Po provedeném průzkumu aplikací nabízených na Google Play[1] či Apple App Store[2] lze konstatovat, že již existují aplikace zabývající se stejnou či podobnou problematikou.

[1] Obchod *Google Play* – https://play.google.com/store
[2] Obchod *Apple App Store* – https://www.apple.com/ios/app-store/

Všechny z nalezených aplikací měly podobné rysy, kterým se aplikace *Jsem OK* snaží vyhnout nebo je vyřešit jinak.

Zamýšlené změny vůči jiným, existujícím aplikacím:

1. Všechny doposud existující aplikace podobného rázu jsou založeny na principu, kdy má jedna osoba nepřetržitý přístup k informaci o aktuální poloze osoby druhé. Tento rys může v očích některých uživatelů již hraničit s omezováním osobní svobody. Některé aplikace k této funkcionalitě přidávají možnost kontroly stavu baterie či aktivní odposlech a sledování činnosti na telefonním zařízení. Neustálou možnost získání přesné polohy druhé osoby bez jejího vědomí mají všechny existující aplikace, porovnávané v této kapitole. Abych se však neopakovala, zmíním ji pouze zde, v celkovém hodnocení aplikací, a ne u každé zvlášť.

2. Druhý prvek, častá výtka vyskytující se mezi recenzemi, je jazyková orientace aplikace. Málokterá správně fungující aplikace tohoto typu nabízí překlad do českého jazyka. Přitom ten by měl patřit, dle mého názoru, mezi základní vlastnosti – zejména s ohledem na předpokládanou cílovou skupinu uživatelů, u kterých se kvůli nižšímu věku nepředpokládá znalost anglického jazyka, jsem vytvořila aplikaci pouze s podporou českého jazyka.

3. Stížnosti zahrnují vysokou spotřebu baterie či dat.

4. U některých aplikací lze sdílet polohu pouze s dalšími jedinci, využívajícími stejnou aplikaci. Pokud by chtěl uživatel informovat o aktuální poloze osobu, nepoužívající přímo daný produkt, není to v rámci dané aplikace možné. Také na tento nedostatek jsem se při vývoji aplikace zaměřila.

5. U mnoha aplikací je plná funkčnost podmíněna zakoupením placené verze. Požadované peněžní částky se pohybují v rozmezí od 200 – 450,– Kč na měsíc. Primárním účelem mé aplikace není komerční využití a neobsahuje reklamy.

.

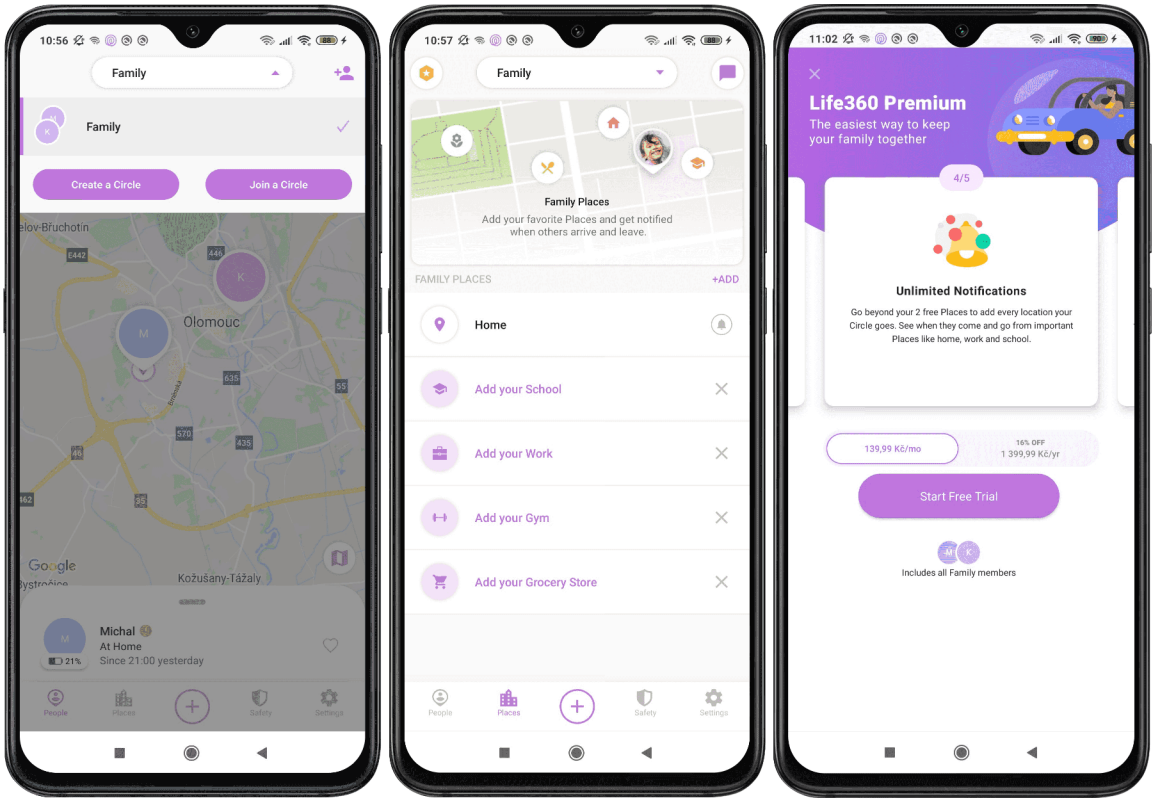
Na zařízeních s operačním systémem *iOS* je k dispozici podobná aplikace s názvem *Find My Friends*[3]

Následující podkapitoly jsou věnovány rozboru a porovnání konkrétních aplikací.

2.3.1 Rodinný lokátor – Life360

Tato aplikace je mezi uživateli nejoblíbenější ve své kategorii. V obchodě *Google Play* má jedno z nejlepších hodnocení – 4,5 hvězdiček z 5, přes 50 milionů stažení a více než 1 milion recenzí. Na mě osobně ovšem po otevření aplikace dopadl spíše pocit zahlcení. Tolik funkcionalit, plno informačních oken, a navíc opakovaně zobrazované reklamy, lákající k zakoupení premium verze (viz názorné snímky obrazovky 2.1).

[3]Aplikace *Find My Friends* v obchodě *Apple App Store* – https://apps.apple.com/us/app/find-my-friends/id466122094



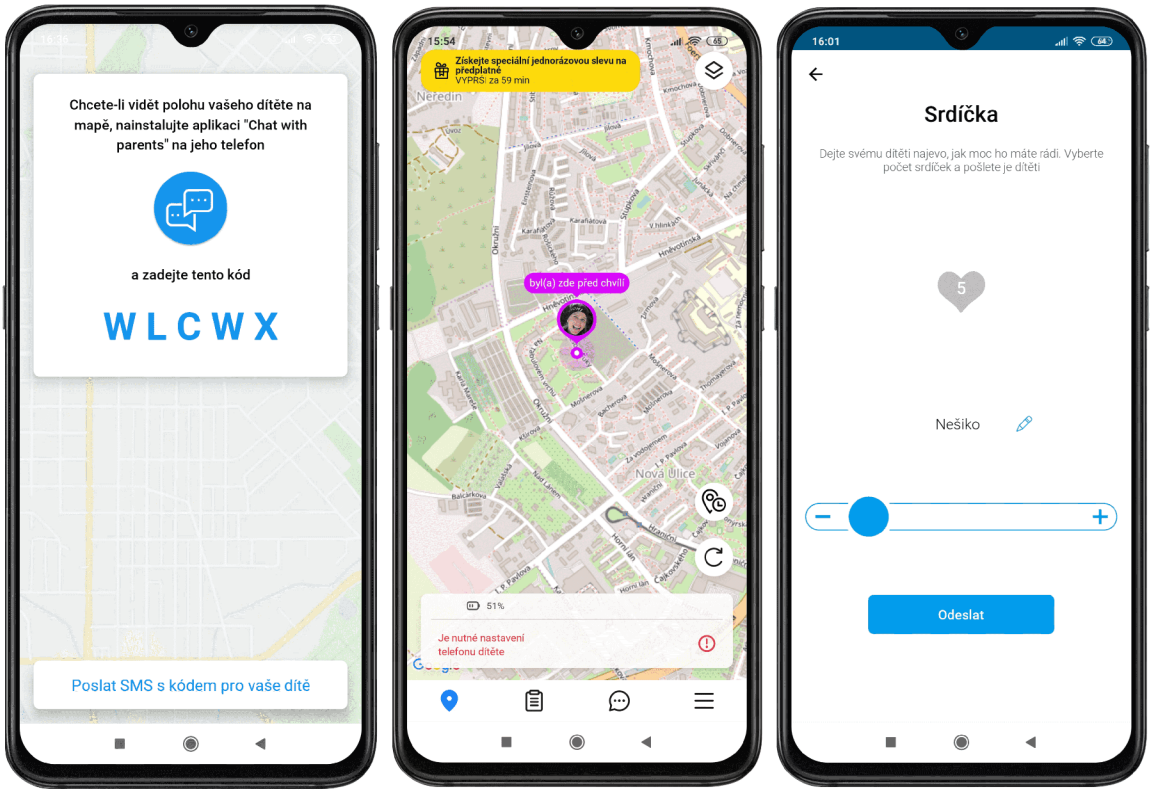
Obrázek 2.1: **Uživatelské rozhraní aplikace *Life360*.** Na prvním snímku obrazovky lze vidět mapu s jedinci momentálně zvolené skupiny, na druhém jsou vyobrazeny možnosti vytvoření nové skupiny a na posledním snímku obrazovky lze vidět podmínky pro využití placeného premium účtu.

Další vlastnosti:

+ Možnost vytvoření sledované zóny přesným umístěním bodu na interaktivní mapě,
+ dostupná jak pro OS *Android*, tak *iOS*,
– aplikace není v češtině,
– dle recenzí časté problémy s lokalizací ostatních uživatelů, náhodně přicházející zprávy, avšak já jsem se s problémy tohoto typu při testování aplikace nesetkala,
– možnost sledování pouze 2 míst – poté je verze placená (139,99,– Kč na měsíc),
- nadstandardní funkce pro starší uživatele používající motorové vozidlo – jedná se o přehled najetých kilometrů, počtu jízd a dokonce je zaznamenána nejvyšší dosažená rychlost,
- funkce sledování stavu baterie ostatních uživatelů – dle mého názoru je tato funkce postradatelná.

2.3.2 Find My Kids – GPS Tracker

Aplikace *Find My Kids*[4] je druhou nejstahovanější svého druhu (4,4 hvězdiček z 5). Díky více než 1 milionu stažení a přes 15 tisíc recenzí v obchodě *Google Play* působí tato aplikace věrohodně. Avšak já jsem k ní zaujala téměř okamžitě negativní postoj. Množství nadstandardních možností, které rodičům umožňují zasahovat do soukromí dětí, je v případě této aplikace extrémní – možnost odeslání hlasité signalizace v případě, že se rodič dítěti nemůže dovolat, možnost „hodnocení“ potomka srdíčky, kdy může rodič vybrat ze škály „nešiko“, až „moje sluníčko“ nebo dokonce odposlouchávat, co se děje v okolí dítěte a kontrolovat, co na telefonu právě dělá (pouze pro předplacenou verzi, která stojí 439,99,– Kč na rok). Snímky obrazovky, vykreslující aplikaci 2.2.



Obrázek 2.2: **Uživatelské rozhraní aplikace *Find My Kids*.** První snímek obrazovky ukazuje, že pro propojení s telefonem dítěte je zapotřebí stáhnout na jeho zařízení jinou aplikaci a sdílet s ním kód; na prostředním ze tří obrázků je vidět základní zobrazení mapy se sledovanými osobami (lze také přepnout na satelitní mapu) a na posledním snímku obrazovky se nachází ukázka způsobu, kterým lze potomkovi udělit takzvané „hodnocení“.

Pokud se rozhodne rodič aplikaci instalovat, požaduje se po něm dítěti nainstalovat aplikaci s názvem „Chat with parents“ – rozdílné názvy, mezi sebou komunikujících aplikací, mohou být matoucí.

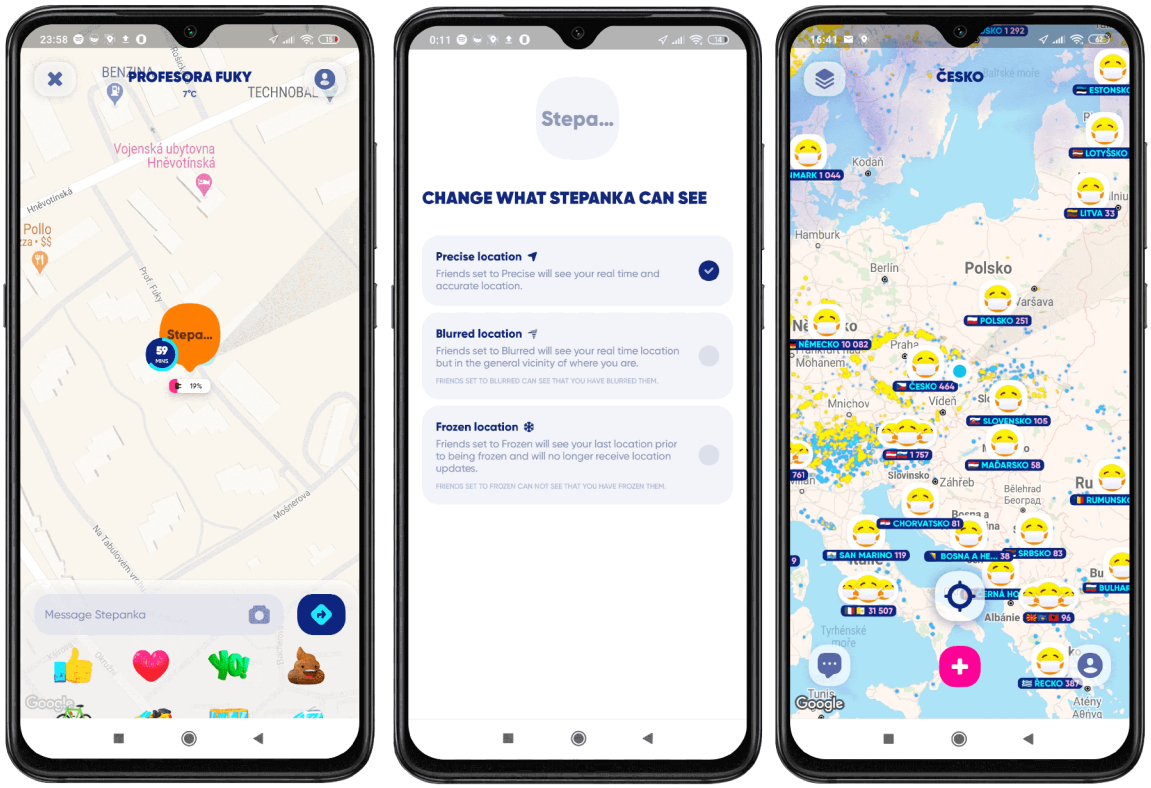
[4] Aplikace Find My Kids – https://findmykids.org/en

Další vlastnosti:

+ Možnost chatu s dítětem přímo v aplikaci,
+ k dispozici verze v českém jazyce,
– odposlech, kontrola činnosti – tyto funkcionality považuji za kontroverzní,
– příliš drahá premium verze,
– rychlé vybíjení baterie,
• je možné kdykoliv vykreslit trasu, na které se dítě v určitý den pohybovalo.

2.3.3 Zenly

Tato aplikace, která má v obchodě *Google Play* 4,2 hvězdiček z 5, více než 10 milionů stažení a nad 71 tisíc recenzí, na mě zapůsobila svým propracovaným a přátelským vzhledem nejlépe ze všech. Je patrné, že je pravidelně aktualizována, jelikož součástí aplikace je nově i mapa, která informuje o počtu lidí nakažených a vyléčených *koronavirem COVID-19* v rámci jednotlivých států. Přátelsky také působí hravá barevnost a nadměrně zaoblené rohy grafických prvků. Funkcionalit je mnoho, ale musím uznat, že jsou logicky utříděny a tudíž nepůsobí chaoticky (viz snímky obrazovky 2.3).



Obrázek 2.3: **Uživatelské rozhraní aplikace *Zenly*.** Na prvním snímku obrazovky je vykreslena mapa se sledovanými jedinci (plus jejich stav baterie), na druhém jsou představeny možnosti, které jsou uživateli nabízeny pro omezení sledování aktuální polohy ostatními uživateli, a na posledním snímku obrazovky je vidět nová aktualizace vyobrazující případy *koronaviru COVID-19* pro jednotlivé státy.

Další vlastnosti:

+ Zapojení gamifikace – možnost získat samolepky do chatu přidáním přátel,
– přílišné zasahování do soukromí – aplikace sama od sebe detekuje, zda uživatel spí nebo ne a stav zobrazuje ostatním uživatelům (možnost zaslání upozornění ostatním po probuzení)
– údaje o aktuálním místě výskytu lze poslat pouze jiným uživatelům stejné aplikace,
• možnost zjištění stavu baterie, chat.

2.4 Cílová skupina a příklady užití

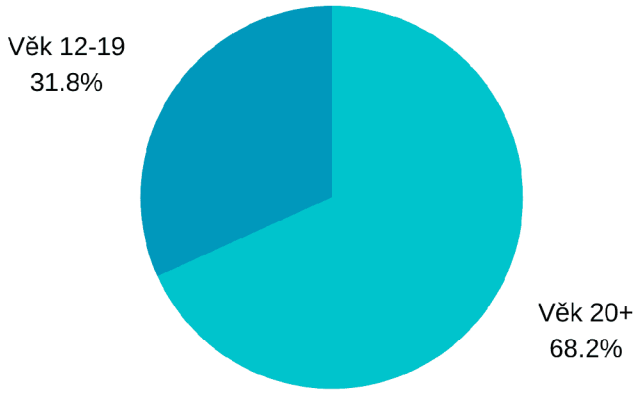
Jakožto typická persóna je ve většině případů očekávána mladší osoba. Tento fakt logicky vyplývá již z problému, který má tato aplikace řešit. Nejčastěji právě uživatelé mladšího věku dostávají za úkol ozvat se příslušné osobě – po doražení domů, do kroužku či na jiné určené místo. Ovšem případy, kdy tato zvyklost přetrvává i do vyššího věku, nejsou ani zdaleka výjimkou. Na základě vlastní zkušenosti, doplněné o názory lidí z mého okolí, bylo možné tuto teorii potvrdit. Vizuální vyobrazení získaných dat (konkrétně ohledně používání aplikace staršími jedinci) lze vidět na obrázku 2.5 v následující podkapitole. Musí se ovšem zvážit fakt, že se dotazník nedostane k většině jedinců mladšího věku, a přesto mohou ve výsledku tvořit převážnou část uživatelů. Právě z tohoto důvodu byla, i navzdory získaným datům, vyvinuta snaha o co nejjednodušší design a logiku aplikace, pochopitelnou i pro ty nejmenší – jednoduché ovládání, český jazyk, hravé barvy a žádné zbytečné či matoucí funkcionality.

Jednotlivé příklady užití:

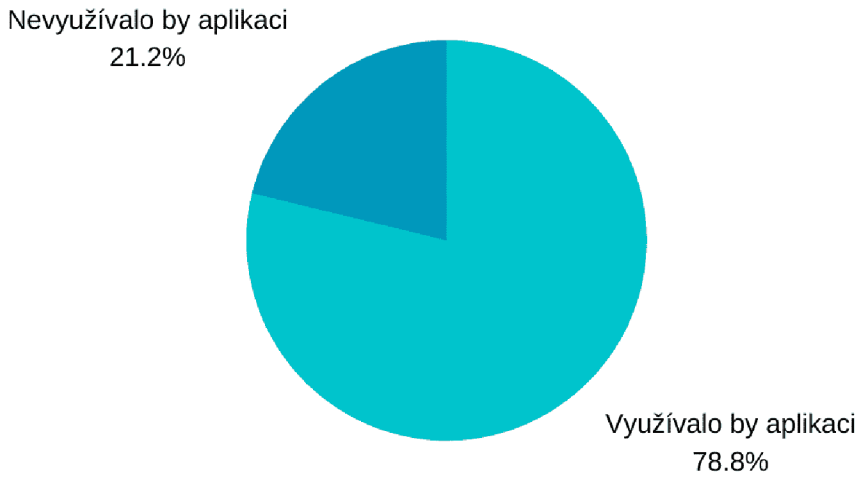
- Malé dítě chodí samo do školy a rodiče chtějí mít jistotu, že v pořádku dorazilo. Avšak zároveň nechtějí omezovat svobodu dítěte a snaží se ho vést k zodpovědnosti. Z tohoto důvodu nebudou kontrolovat polohu dítěte bez jeho vědomí, ale dítě dostane za úkol se ozvat. Potomek dorazí do školy, přijde mu připomenutí, ale stále bude na něm, aby připomenutí zpracoval a zprávu odeslal.
- Tato aplikace nemusí sloužit pouze pro kontrolu potomků. Existují vztahy, ve kterých se partneři informují například o tom, že v pořádku dorazili do práce.
- Aplikace představuje usnadnění v situacích, které se pravidelně opakují, avšak pokaždé v mírně rozlišnou dobu. Nemusí jít vždy o informaci, že je osoba v pořádku. Může se jednat o případ, kdy člověk vyjíždí z domu do práce a chce dát vědět ostatním spolucestujícím, že je vyzvedne za určitý čas od tohoto okamžiku.
- Dalším faktem je, že aplikace *Jsem OK* může být používána pro informování druhých, jak již mimo jiné naznačuje samotný název aplikace. Není to ovšem pravidlem. Mohou se vyskytnout různé situace, ve kterých si chce člověk jednoduše poznamenat příchod do určité oblasti – sám pro sebe. Jednou z takových situací může být příchod uživatele do práce. V takovém případě uživatel s největší pravděpodobností nebude odesílat zprávu sám sobě. Místo toho může zvolit zápis předvolené zprávy do poznámek či jiný podobný způsob, sloužící pro záznam pohybu na pracovišti.

2.4.1 Průzkum zájmu ze strany potenciálních budoucích uživatelů

Aby byly domněnky, na základě kterých je struktura a funkcionalita aplikace budována, podloženy reálnými daty, byl proveden průzkum. Tohoto průzkumu, jenž byl proveden formou online dotazníku, se ve výsledku zúčastnilo celkově 148 lidí různého věku, jak je možné vidět v grafickém zpracování typem koláčového grafu na obrázku 2.4. Z dotazníku je patrné, že téměř osmdesát procent dotazovaných by v jisté míře výslednou aplikaci využívalo (možné vidět na obrázku 2.5). Dotazník dále obsahoval otázky jako „*jakého je jedinec pohlaví*“, „*jak často by aplikaci využíval*“ či „*komu by informační zpráva byla určena*“. Díky získaným odpovědím bylo možné více pochopit potřeby uživatelů a případně aplikaci upravit.



Obrázek 2.4: **Procentuální mapování věkového rozložení všech respondentů.** Z celkového počtu 148 respondentů bylo 47 ve věkovém rozmezí 12 – 19 let, a zbylých 101 dotazovaných bylo starších 20 let.



Obrázek 2.5: **Část výsledku průzkumu provedeného na vzorku široké veřejnosti.** Procentuální podíl odpovědí lidí starších 20 let, na otázku *„Využívali byste aplikaci?“*

Kapitola 3

Použité technologie a prostředí pro vývoj mobilních aplikací s operačním systémem Android

Třetí kapitola zmiňuje technologie a různá prostředí, která byla zapotřebí pro tvorbu vyvíjené *android*ové aplikace [12]. Zmiňuje, jaké jsou výhody zvoleného přístupu. Podkapitoly 3.1 a 3.3 navíc uvádí jiné způsoby, kterými by bylo možné k dané záležitosti přistupovat.

3.1 Programovací jazyk Kotlin

Kotlin[1] je v dnešní době jedním z nejpoužívanějších programovacích jazyků pro tvorbu androidových aplikací. A to i navzdory faktu, že jeho oficiální stabilní verze byla společností *JetBrains*[2] uvedena na trh až roku 2016.

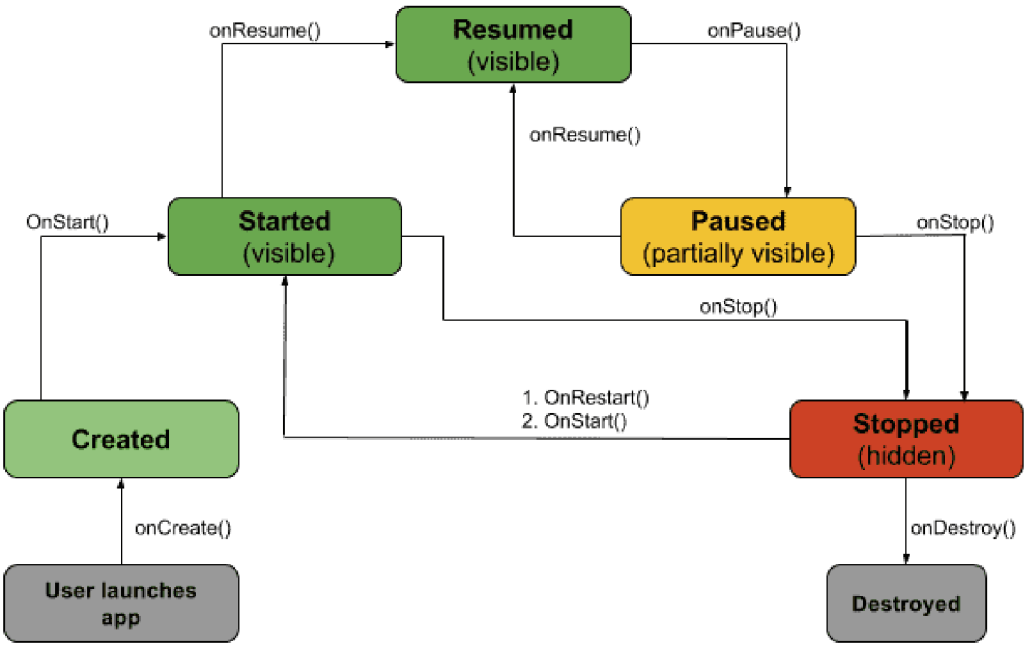
Tento programovací jazyk sice není syntakticky kompatibilní s jazykem *Java*, avšak je navržen pro interoperabilitu s jejími knihovnami (na některých knihovnách jádra je dokonce závislý). Jak se v anglickém jazyce řekne – jazyk Kotlin lze považovat za „*based on Java*“. Cílem jazyka *Kotlin* byla na jednu stranu interoperabilita s jazykem *Java*, na druhou stranu však byla zdůrazněna snaha o zkrácení doby kompilace (v porovnání s časem kompilace v prostředí programovacího jazyka *Java*). Podobnost těchto dvou zmiňovaných jazyků je příčinou faktu, že se programátoři, pracující na vývoji aplikací, neshodnou na jednoznačně nejvhodnějším jazyku k programování aplikací operačního systému Android.

Již na první pohled viditelná odlišnost mezi jazykem *Kotlin* a *Java* je postradatelnost středníků na koncích řádků v případě jazyka *Kotlin*. U deklarace proměnných si lze všimnout jednoho ze základních (avšak ne tak zásadních) rozdílů – datový typ je určován až po názvu proměnné.

Základní vlastnosti jazyka *Kotlin*:

- Vstupním bodem do aplikace je funkce *main*,
- základními stavebními bloky aplikace jsou takzvané *aktivity*, které tvoří jednotlivé obrazovky aplikace – zajišťují tak interakci uživatele s aplikací,

[1] Oficiální webová stránka programovacího jazyka *Kotlin* – https://kotlinlang.org/
[2] Oficiální webová stránka společnosti *JetBrains* – https://www.jetbrains.com/



Obrázek 3.1: **Znázornění životního cyklu každé aktivity v rámci aplikace.** Metoda *onCreate()* je vyvolána při prvním vytvoření aktivity, *onStart()* před představením aktivity uživateli, poté vždy následuje metoda *onResume()* – v momentě kdy aktivita vstupuje do popředí a následně *onPause()* – před vstoupením aktivity do pozadí, metoda *onStop()* je vyvolána ve chvíli, kdy uživatel již nevidí aktivitu. Před navrácením se k aktivitě je volána metoda *onRestart()*, následována poslední metodou *onDestroy()*, která je provedena těsně předtím, než je aktivita zničena.

- každá aplikace má svůj životní cyklus (názorný obrázek 3.1 převzatý z webu [15]), definovaný metodami, které se spouštějí v přesně definovaných situacích v určeném pořadí [12],

\+ je podporováno odvozování typu proměnných,

\+ rozlišuje mezi nulovatelnými a nenulovatelnými datovými typy (za názvem typu nulovatelných objektů musí být otazník),

\+ jednodušší způsob zápisu kódu,

\+ například oproti programovacímu jazyku *Flutter*[3] je *Kotlin* delší dobu na trhu – v očích programátorů tak představuje spolehlivost, podloženou oficiální podporou, a snazší dohledatelnost případných problémů, řešených na internetu.

Jiné populární programovací jazyky pro tvorbu aplikací operačního systému Android:

- **Java** – jak bylo již zmíněno, tak v dnešní době *Java* sdílí s *Kotlinem* první místo na žebříčku jazyků používaných pro tvorbu Android aplikací. Dokud se na scéně neobjevil *Kotlin*, *Java* byla naprostou jedničkou. Proto existuje spousta online zdrojů, kde lze hledat řešení v případě výskytu problémů.

[3] Oficiální webová stránka programovacího jazyka *Flutter* – https://flutter.dev/

- **C#** – někdy označován za nejuniverzálnější jazyk pro tvorbu aplikací, jelikož lze použít *open source* platformu *Xamarin*[4] pro vytváření nativních multiplatformních aplikací – jak pro *iOS*, *Android* tak i *Windows*[5].
- **Flutter** – nový framework, který funguje na obou dvou významných operačních systémech – *iOS* i *Android*. Hybridní *Flutter* byl inspirován *Reactem*. Aplikace se píší v *Dartu*[6] a kompilují se do nativního kódu platformy [8].
- **React Native**[7] – tento programovací jazyk je zajímavý svou architekturou, která dovoluje i nadále využít pro „business logiku“ *JavaScript*, a přesto vytvářet aplikace složené z nativních komponent. *React Native* přichází s určitou dávkou abstrakce pro nativní komponenty obou platform (*iOS, Android*), aby kód mohl být z co největší části sdílený. V jistých situacích je ovšem stále nutné psát v jazyce jedné či druhé platformy.

3.2 Vývojové prostředí Android Studio

Android studio je oficiálním vývojovým prostředím[8] pro tvorbu *androidových* aplikací, založené na *IntelliJ IDEA* (komerční vývojové prostředí), spravované firmou *Google*. Toto vývojové prostředí je zdarma k dispozici pro uživatele všech platforem (*Windows*, *Mac OS X* a *Linux*).

3.3 SQLite databáze

K problému trvalého uchovávání dat lze přistupovat různými způsoby. Existují *online databáze*, jakožto externí úložiště (viz podkapitola 3.3.1), lokální úložiště ve formě SQLite databáze[9] [14] či dokonce uchovávání dat v textovém souboru.

Aplikace *Jsem OK* nepotřebuje uchovávat v databázi velké množství dat, proto stačí interní úložiště v mobilním zařízení. Řešení, které se v této situaci nabízí je použití *SQLite* databáze.

SQLite je relační databázový systém (tzn. založen na tabulkách), obsažený v knihovně, psané v programovacím *jazyce C*. V současnosti je tento databázový systém tím nejvíce využívaným. *SQLite* databáze je vestavěna do mobilních zařízení, většiny počítačů a nespočtu dalších zařízení, která lidé denně používají. Formát složek databáze zaručuje bezproblémové sdílení napříč různými typy platforem – je možné kopírovat data mezi *32-bitovým* a *64-bitovým* systémem či mezi „*big-endian*“ a „*little-endian*“ architekturou. *SQLite* je, po přilinkování k aplikaci, k dispozici pomocí rozhraní (na rozdíl od jiných externích databází, založených na principu *klient-server*).

Důležitou třídou, která je určena pro práci s *SQLite* databází, je třída zvaná `SQLiteOpenHelper`. Tato pomocná třída je zahrnuje a dovoluje upravovat metody, potřebné pro práci s databází při jejím vytvoření, prvním otevření či dalším použití [14].

Každá databáze je ukládána do individuálního souboru *.dbm* (*Database Manager*) – pro přístup k datům je pro urychlení procesu využíváno hašovacích technik.

[4] Oficiální webová stránka pro platformu *Xamarin* – https://visualstudio.microsoft.com/cs/xamarin/
[5] Obchod s *Windows* aplikacemi – https://www.microsoft.com/cs-cz/store/apps/windows
[6] Oficiální webová stránka programovacího jazyka *Dart* – https://dart.dev/
[7] Oficiální webová stránka programovacího jazyka *React Native* – https://reactnative.dev/
[8] Oficiální webová stránka *Android Studia* – https://developer.android.com/studio/intro
[9] Oficiální stránky relačního databázového systému *SQLite* – https://www.sqlite.org/index.html

Postup pro používání *SQLite*:

1. Vytvoření třídy *DatabaseHandler.kt* pro manipulaci s daty v databázi.
2. Nově vytvořená třída musí dědit *SQLiteOpenHelper*, který je používán pro vytvoření a aktualizování databáze dané aplikace.
3. Implementace takzvaných *CRUD* operací – tato zkratka označuje anglická slova *CREATE*, *READ*, *UPDATE* a *DELETE*. Ukázku obecného kódu pro práci s *SQLite* databází je možné vidět na obrázku 3.2.

```
class DatabaseHandler(var context: Context) : SQLiteOpenHelper(context,
DatabaseHandler.DB_NAME, null, DatabaseHandler.DB_VERSION) {

    override fun onCreate(db: SQLiteDatabase?) {
        val CREATE_TABLE = "CREATE TABLE $TABLE_NAME (" +
                ID + " INTEGER PRIMARY KEY," +
                NAME + " TEXT," + DESC + " TEXT," +
                COMPLETED + " TEXT);"
        db.execSQL(CREATE_TABLE)
    }

    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {
        val DROP_TABLE = "DROP TABLE IF EXISTS " + TABLE_NAME
        db.execSQL(DROP_TABLE)
        onCreate(db)
    }

    fun addTask(tasks: Tasks): Boolean {
        val db = this.writableDatabase
        val values = ContentValues()
        values.put(NAME, tasks.name)
        values.put(DESC, tasks.desc)
        values.put(COMPLETED, tasks.completed)
        val _success = db.insert(TABLE_NAME, null, values)
        db.close()
        return (Integer.parseInt("$_success") != -1)
      }
}
```

Obrázek 3.2: **Ukázka manipulace s daty v rámci *SQLite* databáze.** Příklad funkce *onCreate*, která slouží k vytvoření tabulky, funkce *onUpgrade*, sloužící pro přepis již existující tabulky a příklad přidání záznamu do tabulky pomocí funkce *addTask*. U funkcí, sloužících pouze pro získávání dat z tabulky (operace *R* – *read*), je používána databáze, která je uzpůsobena pouze ke čtení – takzvaná *readableDatabase*. U zbytku operací je nutné povolit i zápis – k tomu je využívána *writableDatabase*.

3.3.1 Firebase

Externí datové úložiště společnosti *Firebase*[10] je spravováno firmou *Google*, a je vhodné spíše pro ukládání dat většího objemu. Princip fungování je následující – data jsou do data-

[10]Seznámení s principy *Firebase* – https://www.zdrojak.cz/clanky/firebase-kratke-seznameni/

báze vkládána ve formátu stromově strukturovaném, databáze neobsahuje žádné tabulky. Vkládání dat je tedy, až na absenci polí, podobné formátování *JSON*. Pro získávání dat uložených ve *Firebase databázi* je používána syntaxe *URL* (položky odděleny lomítky). Ukázka kódu struktury (přejato z webu [**?**]) lze vidět na obrázku 3.3, v jehož popisu je vysvětleno i získávání dat z této externí databáze.

```
{
    name: "Zdenda",
    address: {
        city: "Bruntál",
        street: "Ruská"
    }
}
```

Obrázek 3.3: **Ukázka strukturovaných dat, vkládaných do *Firebase databáze*.** Například na řetězec „Zdenda“ se lze následně odkázat *URL* odkazem *https://nazev.firebaseio.com/name*.

3.4 Prostředí Google Console[11]

V této aplikaci je potřeba pro fungování části mapy, stejně jako pro získávání aktuální polohy či hledání existujících adres na základě adresy uživatelem právě zadávané, mít k dispozici *API klíč*. K jeho získání je nutné mít založen účet na platformě *Google Cloud* a k němu připojenou platební metodu, jelikož některé služby jsou zpoplatněné. Konkrétně u této aplikace je potřeba získat klíč pro „Places API“ (služba, využívající *HTTP požadavky* a vracející informace o místech buďto ve formátu *JSON* nebo *XML*).

[11] Oficiální webová stránka *Google Play Console* – https://play.google.com/

Kapitola 4

Geofencing

Geofencing je jednou z metod, pomocí kterých lze sledovat aktuální polohu. Tomuto tématu je věnována celá kapitola z toho důvodu, že tvoří značnou část vyvíjené aplikace, zároveň je s největší pravděpodobností tou nejproblematičtější. Zkoumání a testování dostatečně přesné funkčnosti bylo obzvláště časově náročné. V této kapitole je vysvětleno, na jakém principu metoda *geofencingu* funguje, jaké jsou její výhody/nevýhody, jaké změny musely být provedeny k plné funkčnosti metody na mobilních zařízeních *Androidu* verze *Oreo* a výše, a jak tento způsob řešení aplikovat. Zároveň jsou v kapitole zmíněny možnosti, které se nabízejí pro získávání aktuální polohy a rozdíly mezi nimi.

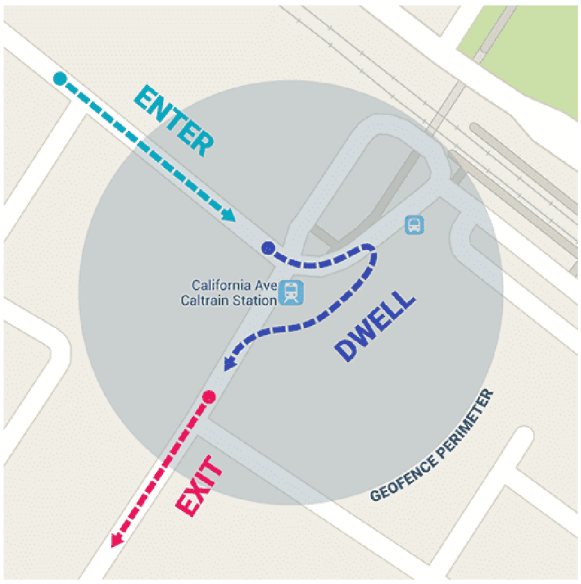
4.1 Základní principy *geofencingu*

Geofencing je metoda, pomocí které lze obdržet upozornění ve chvíli, kdy se uživatel blíží k virtuálně vytvořené oblasti, která je pro něj jistým způsobem zajímavá nebo naopak, když oblast opouští. Třetí možností je obdržení upozornění až po určité době, po kterou se ve vybrané oblasti jedinec zdržuje. Je možné omezit životnost oblasti na určitý čas, po jímž uplynutí přestává být oblast sledována, nebo sledování zóny vypnout manuálně (případ vyvíjené aplikace). Informace o této metodě jsem čerpala na oficiálních stránkách *androidových* vývojářů, viz [3].

Pro vytvoření oblasti, na kterou chce být uživatel upozorněn, je potřeba znát zeměpisnou šířku a délku adresy. Okolo tohoto bodu se následně vytvoří pomyslná kruhová oblast o daném poloměru (lze vytvořit i oblast tvaru nepravidelného, tuto variantu však aplikace *Jsem OK* nepodporuje).

Oblastí si může každý uživatel vytvořit několik, maximálně však 100 na jednu aplikaci. Pro každou oblast je možné nastavit jiné vlastnosti – jedna může reagovat na vstup do oblasti o velikosti 100 m, druhá například na setrvání v oblasti o jiné velikosti po dobu 5 minut.

Když se následně uživatel, spolu s mobilním zařízením, pohybuje, pozice je obnovována na základě informací získaných ze satelitů, Wi-fi, GPS či bluetooth beans (do českého jazkya přeloženo jako *„bluetooth majáky“*).



Obrázek 4.1: **Grafické znázornění jednotlivých událostí, které mohou být detekovány.** Události ENTER (vstup do zóny), DWELL (setrvání v zóně) a EXIT (výstup ze zóny). Ilustrační obrázek viz [3]

Dva typy *geofencingu*:

1. **Aktivní** – tento typ využívá služby *GPS* po celou dobu spuštění aplikace, a proto také v důsledku spotřebovává více baterie; vyžaduje běh aplikace v popředí po celou dobu využívání služby
2. **Pasivní** – nevyžaduje neustále aktivní stav aplikace, běží v pozadí; nevyužívá GPS služby (je spíše využíváno pro sběr dat, nelze používat pro včasné zasílání notifikací)

4.2 Výhody a nevýhody *geofencingu*

+ Implementace nevyžaduje doplňkový hardware,
+ služba je dostupná jak pro operační systém Android, tak i iOS,
+ při správné implementaci lze počítat s přesností v jednotkách metrů,
– předtím, než je možné začít využívat této služby, je nutné povolení sledování polohy uživatelem – což na jedince může působit odstrašujícím způsobem (a ve výsledku ho tak od užívání celé aplikace odradit),
– spotřeba baterie – pokud je potřeba získávat přesné informace o aktuální poloze, je nevyhnutelná častá kontrola polohy (v kombinaci se zapnutým sdílením polohy), která v důsledku znamená nemalou spotřebu baterie.

Jiné možné využití *geofencingu*:

- Marketing – upozornění zákazníka na slevy právě ve chvíli, když prochází kolem daného obchodu; doporučení reklam zákazníkovi na základě sledování jeho pohybu.

- Sledování pohybu zaměstnanců v pracovní době (již existují firmy, které této možnosti využívají).
- Správa výpůjček – umožní zpřesnit vyúčtování díky schopnosti sledování doby pohybu mimo místo zapůjčení, ušetření administrativy.

4.3 Implementace metody *geofencingu*

Tato metoda využívá k lokalizaci zařízení službu, poskytovanou společností *Google*. K této službě je přistupováno prostřednictvím *Google Services API*[1].

4.3.1 Změny metody s příchodem OS *Android* verze 8 a 10

Jak už je to u moderních technologií zvykem, jejich vývoj postupuje kupředu neskutečným tempem. V průběhu vývoje aplikace jsem proto byla nucena zkoumat, která z aplikovaných aktualizací negativně ovlivnila funkčnost aplikace. Pročetla jsem plno oficiálních článků, ale v žádném z nich nebyl uveden postup implementace *geofencingu*, který by byl kompatibilní s desátou verzí OS *Androidu* a byl plně funkční. Část hledání a kombinování různých segmentů možných řešení tak, aby ve výsledku fungoval hlavní princip vyvíjené aplikace (oznámení při vstoupení/vystoupení ze zóny), zabrala nejspíše nejvíce času z celého projektu.

Problém nastal ve chvíli, kdy OS *Android* verze 8.0 (*Oreo*) byl optimalizován za účelem nižší spotřeby baterie zařízení. Tato změna mimo jiné znamenala také nové chování aplikací, které nejsou v daném okamžiku aktivně používány (tj. viditelná aplikace) [16]. Také je méně časté snímání WiFi signálů a nová poloha není zjišťována v případě, že je zařízení delší dobu připojeno ke stejnému statickému přístupovému bodu.

Původně bylo aplikacím umožněno využívat různých služeb i na pozadí (v případě vyvíjené aplikace *Jsem OK* se jedná o aktivní získávání polohy), nyní je aplikace po chvíli neaktivity přeřazena do běhu na pozadí [10]. Pro metodu *geofencingu* to neznamená nic dobrého – zatímco předchozí verze *Androidu* byly schopné požadovat díky *Google API* polohu v libovolně zvoleném intervalu, i při běhu aplikace na pozadí, v důsledku aktualizace je nyní aktuální poloha detekována pouze několikrát za hodinu [2]. Většinou je tento problém řešen pomocí umístění kódu, který je potřeba vykonávat i na pozadí, do takzvaného *foreground service* (služba běžící na popředí po celou dobu její existence) [16].

Aby aplikace směla získávat aktuální polohu i po dobu běhu na pozadí, s aktualizací OS *Android* verze 10 je zapotřebí od uživatele získat povolení k poloze zařízení po celou dobu [1] (viz snímek obrazovky 4.2).

Části řešení potřebné pro funkční získávání aktivní polohy na pozadí:

- **Foreground service** – jedná se o službu, která běží na popředí[2] po celou dobu běhu aplikace [7]. U aplikací pro OS *Android* to znamená, že by služba na popředí měla mít vyšší prioritu než „obyčejná“ služba. Po celou dobu jejího spuštění musí být uživateli poskytována informace o jejím běhu formou „pevné“ notifikace[3]. Tato služba nově nesmí být volána z běhu apli-

[1] *Google Services API* – https://developers.google.com/android/guides/setup

[2] aplikace běžící na popředí = aplikace otevřená na aktivní obrazovce, aplikace, jejíž součástí je právě *foreground service* nebo případ, kdy je aplikace napojena na jinou aplikaci, a ta běží na popředí

[3] Pevná notifikace = notifikace, kterou nelze odstranit

Obrázek 4.2: **Snímek obrazovky, s různými možnostmi udělení oprávnění běhu aplikace na pozadí.** Od desáté verze OS *Android* uživatel musí povolit přístup k poloze po celou dobu. Tento dotazovací dialog je zobrazen pouze pokud aplikace běží na zařízení s OS *Android* verze 10 a výše.

kace na pozadí – pokud by se tak stalo, výsledkem by byla výjimka typu `IllegalStateException`. V manifestu je zapotřebí deklarace atributu oprávnění `android:permission="android.permission.FOREGROUND_SERVICE"`. Podmínkou pro uvedení této služby do chodu je také fakt, že musí být spuštěna z aktivity a do pěti sekund poté zavolána funkce `startForeground(notificationId: Int, notification: Notification)` v rámci již spuštěné služby, čímž se sama zaregistruje jako služba na popředí [2]. Aplikace, běžící na verzi *Androidu* 8.0 a výše, by měly ke spuštění služby používat místo `Context.startService()` metodu `ContextCompat.startForegroundService()`. Upozornění je ze stavového řádku odstraněno po dokončení zadané práce nebo po zavření aplikace. Bez implementace této části by metoda *geofencingu* na zařízeních s OS *Android* verzí 8.0 nefungovala, jelikož by aplikace běžící na pozadí nebyla schopna aktualizovat polohu. Implementace služby na popředí je tedy nutností.

- **Broadcast receiver** – k operování s aplikací při běhu na pozadí je zapotřebí implementace dynamického *BroadcastBroadcast Receiver*u. *Broadcast Receiver* je objekt na vysílání a přijímání. Poslouchá na pozadí a reaguje na události, které se odehrávají na zařízení [12].

- **Job Intent Service** – touto službou je potřeba nahradit *Intent Service*, který se pro obsloužení vzniklých událostí využíval v OS *Android* verze 8 a dříve [2]. Co se týče novějších verzí OS *Android*, může být totiž tato služba již nespolehlivá. Při deklarování `Job Intent Service` v manifestu musí být deklarován atribut oprávnění `android:permission="android.permission.BIND_JOB_SERVICE"`, který zaručuje, že tuto službu může spouštět pouze *JobScheduler*.

Oprávnění nutná pro správné fungování aplikace:

1. **ACCESS_FINE_LOCATION** – jestliže je zapotřebí přistupovat k poloze uživatele, je povinné žádat jeho oprávnění. V takovém případě se nabízí dva typy možných přístupů vedoucích k získání polohy. Pro první typ přístupu, získávající polohu pomocí GPS modulu nebo od poskytovatele sítě, je nutné oprávnění typu `ACCESS_FINE_LOCATION`. Pro druhý typ přístupu, který využívá získávání polohy prostřednictvím poskytovatele sítě, je nutné oprávnění `ACCESS_COARSE_LOCATION`. Druhý přístup získává data s nižší přesností a jeho využití tak dává smysl pouze tehdy, pokud není uděleno oprávnění typu `ACCESS_FINE_LOCATION`.

2. **ACCESS_BACKGROUND_LOCATION** – zařízení s OS *Android* verzí 10 nově vyžadují deklaraci tohoto oprávnění v souboru `AndroidManifext.xml` kvůli běhu aplikace na pozadí (způsob deklarace viz obrázek 4.3). Toto oprávnění spadá do kategorie tzv. nebezpečných oprávnění (anglicky *dangerous permissions*[4]) a tím pádem je nutné se na něj uživatele dotazovat. Toto oprávnění může být uživatelem kdykoliv manuálně odebráno, jeho udělení je proto nutné průběžně ověřovat.

3. **FOREGROUND_SERVICE** – v případě implementace služby běhu na popředí musí být pro správný chod aplikace deklarováno toto oprávnění.

4. **WAKE_LOCK** – toto oprávnění je potřeba získat v případě, že aplikace využívá *Job Intent Service*. Tento mechanismus je využíván pro udržení aplikace v aktivním stavu.

5. **VIBRATE** – vyvíjená aplikace využívá pro upozornění uživatele notifikaci formou vyskakovacího okna, společně s tónem a vibrací. Pro použití vibrací je zapotřebí dopředná deklarace tohoto oprávnění v manifestu. Aplikace by fungovala i bez tohoto oprávnění, není tudíž nezbytné. Zvyšuje se tím však pravděpodobnost, že uživatel zaregistruje příchozí notifikaci.

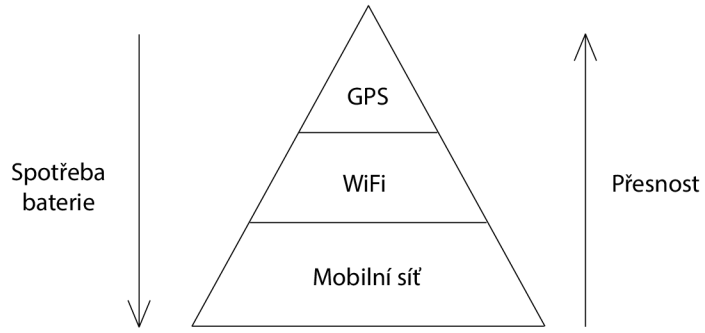
```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Obrázek 4.3: **Obecná forma deklarace oprávnění v soubrou `AndroidManifest.xml`.**

4.4 Možnosti získávání aktuální polohy

Aplikace se snaží kombinovat nejvyšší možnou přesnost lokalizace s co nejmenší spotřebu baterie zařízení.

[4]Skupina nebezpečných oprávnění – tato skupina pokrývá oblasti, kdy aplikace potřebuje využívat data či osobní údaje uživatele (např. kontakty).



Obrázek 4.4: **Grafické znázornění vztahu mezi přesností a spotřebou baterie pro jednotlivé metody získávání aktuální polohy.**

Různé způsoby získávání polohy:

1. **GPS modul (Global Positioning System)** – v dnešní době lze nalézt v každém mobilním zařízení. Tento systém sestává ze 32 družic, díky čemuž jeho největší výhoda spočívá ve schopnosti určit lokaci s velikou přesností – v rámci několika metrů. Tato přednost je ovšem vykoupena vyšší spotřebou energie. Mezi další nedokonalosti patří také omezená dostupnost, zejména uvnitř budov, v podzemí či pod vodou [11].

2. **WiFi** – tento systém využívá *MAC adres* bezdrátových přístupových bodů – tudíž je logicky přístupnější a přesnější v hustě obydlených oblastech. Polohu získanou pomocí WiFi sítě lze určit s přesností 50 – 500 m [11].

3. **Mobilní síť** – mobilní zařízení dokáže vyhledat aktuálně přijímané základnové stanice prostřednictvím síťového připojení, založeného na *IP*, za účelem získání polohy. Jelikož oblasti základnových stanic jsou značně rozsáhlejší než oblasti *WiFi* hotspotů, je přesnost této metody 500 m a více [11]. Určování polohy tedy není natolik spolehlivé, na druhou stranu je výhodou nižší spotřeba baterie.

4. **Pasivní získávání polohy** – pasivní získávání polohy znamená, že sama aplikace aktivně nezískává polohu, ale pouze čerpá tuto informace z jiných aplikací, zjišťujících polohu. Tato metoda je úspornější pro samotnou aplikaci, jelikož běží o služby méně, ovšem data musí být stejně získávána v rámci jiné aplikace, která rovněž spotřebovává energii (navíc se vyvíjená aplikace stává závislou na té, která poskytuje lokalizační informace). Tento přístup snižuje možnost optimalizace spotřeby energie, jelikož to má ve své režii poskytující aplikace.

Pro konkrétní typ aplikace, s ohledem na její požadovaný účel, je tedy potřeba vybrat optimální způsob získávání polohy. Obecně nejvýhodnější, ve smyslu efektivity, se jeví kombinace všech výše zmíněných možností (ty mohou být navíc doplněny například o informace ze senzorů). Jelikož v této aplikaci záleží na přesnosti v rámci metrů, je použita metoda získávání polohy pomocí *GPS modulu*. Tento přístup odstraňuje omezení spojené s nutností dostupnosti internetového připojení. Uživateli je doporučeno mít zapnuté Wifi vyhledávání, i když v ten okamžik není k žádné WiFi síti připojen – zvyšuje se tak počet možných bodů, ze kterých lze získat informace o aktuální poloze. Ilustrační porovnání metod lze vidět na obrázku 4.4.

Kapitola 5

Návrh uživatelského rozhraní aplikace

Základní pravidla pro tvorbu uživatelského rozhraní operačního systému *Android* jsou dána doporučeními v dokumentu *Android Design Guidlines*[1], vytvořeném firmou *Google*. Dalšími pravidly (vizuálním jazykem), opět vytvořenými společností *Google* jsou *Material Design*[2]. Ty se snaží spíše obeznámit vývojáře, tvořící uživatelská rozhraní, s rozmanitými možnostmi, novinkami v této oblasti a jejich správným užíváním. Tato pravidla unifikují styl aplikací napříč všemi zařízeními, aby bylo pro uživatele jejich používání co nejvíce intuitivní. Nedílnou součástí *Material Designu* jsou také fonty a různé způsoby animací jednotlivých prvků.

Ve vyvíjené aplikaci bylo uživatelské rozhraní definováno za pomocí *XML* souboru a nasazeno v nativním jazyce *Kotlin*.

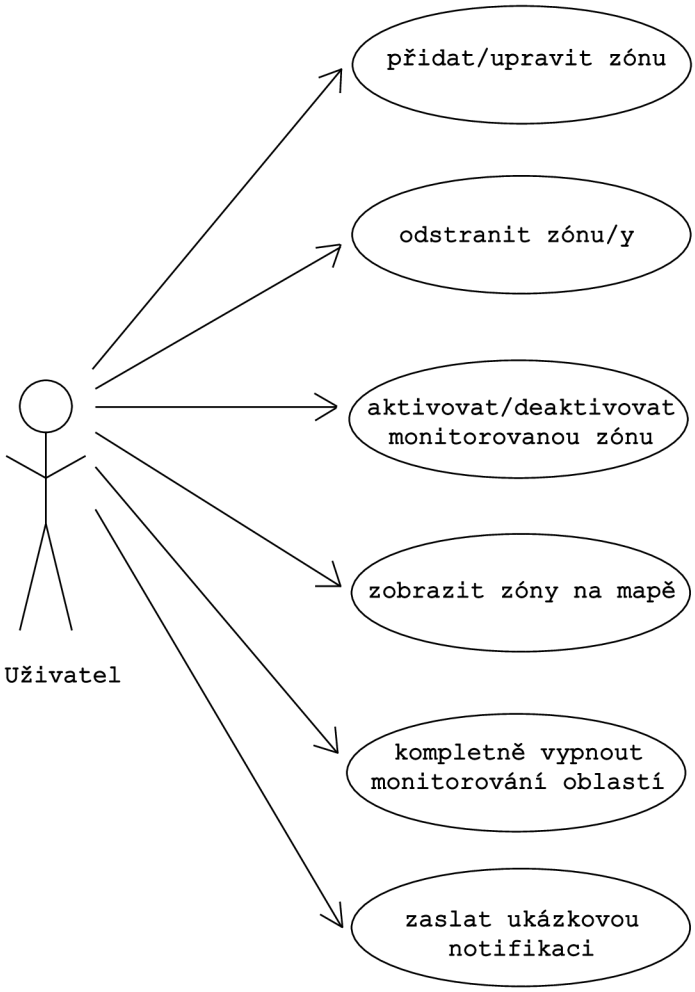
5.1 Návrh použití

Níže vykreslený diagram případů užití (viz obrázek 5.1) znázorňuje možné případy užití finální verze produktu. Podrobněji je vývoj používání aplikace zdokumentován v následujících kapitolách.

5.2 Prvotní návrh

Prvotní návrh aplikace (viz snímky obrazovky 5.2) byl začátečnicky naivní, a po delším promyšlení i zbytečně složitý. Právě komplikovanost se zdála být nevýhodou aplikací již existujících. V závěru tvorby prvního návrhu bylo jasné, že je potřeba od určitých částí upustit – jednou z nich byla například funkce přihlašování a ochrany dat uživatele. Jelikož aplikace nebude propojena s kontakty, ani nebude pracovat s jinými daty, než s adresami zón, usoudila jsem, že přihlašování do aplikace není nutné. Jako domovská stránka měla být původně použita mapa, se zobrazením aktivních zón. Od tohoto záměru jsem také upustila, jelikož oblasti zájmu od sebe mohou být poměrně vzdálené (pravděpodobně by byla na mapě vždy pouze jedna z oblastí zájmu, v jiném případě by mapa musela být značně oddálená a zbavena detailů).

[1] *Android Design Guidlines* – https://developer.android.com/design/
[2] *Material Desgin* – https://material.io/



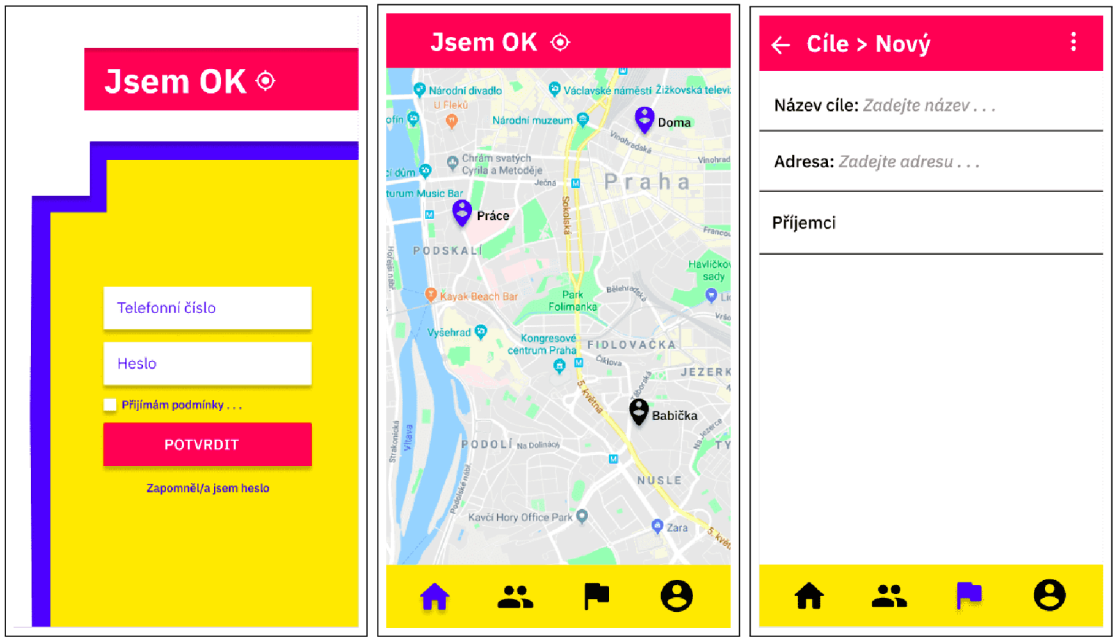
Obrázek 5.1: **Diagram případů užití vyvíjené aplikace *Jsem OK*.**

V další fázi vývoje bylo tedy hlavním cílem uchovat koncept aplikace jednoduchý a implementovat pouze smysluplné funkcionality. Proto jsem se rozhodla, že se aplikace *Jsem OK* uživateli nadále bude jevit jako „pouhý“ seznam zón, které on sám založil, a se kterými bude mít možnost nadále manipulovat.

Co se týče vzhledové složky aplikace, barevná trikolora byla úmyslně zvolena již od začátku tak, aby evokovala pocit bezpečí a jistoty. Kombinace žluté a červené symbolizuje plavčické barvy, modrá je barva uklidňující a symbolizuje moře.

5.3 Iterativní úpravy

Tvorba prvního prototypu aplikace je vždy subjektivní, tudíž jsou velmi důležité následně prováděné iterativní modifikace (založené zejména na zpětné vazbě, získané od testujících uživatelů). Při prototypování si lze ujasnit preference uživatele, uvědomit chyby a nejasnosti v návrhu produktu a vyzkoušet jeho pochopitelnost a funkčnost přímo v rukou uživatele. Následující podpodkapitoly postupně popisují fáze vývoje aplikace.



Obrázek 5.2: **Prototypování vznikající aplikace.** Původně měla mít aplikace *Jsem OK* jiný rozsah a pracovat více s daty uživatele – proto v původním prototypování byla zahrnuta i aktivita přihlašování uživatele, která nakonec nebyla využita. Dále lze na obrázcích vidět mapu s vytvořenými zónami, která je ve výsledku sice realizovaná, avšak nepředstavuje domácí stránku aplikace, a poslední snímek vyobrazuje hrubou představu o zakládání nové zóny. Také si lze všimnout ikon na spodní liště – původně měla být aplikace složena z fragmentů, jelikož byla zamýšlena širší funkcionalita. Prototypování aplikace bylo prováděno pomocí online nástroje *Figma*[4].

5.3.1 První fáze vývoje

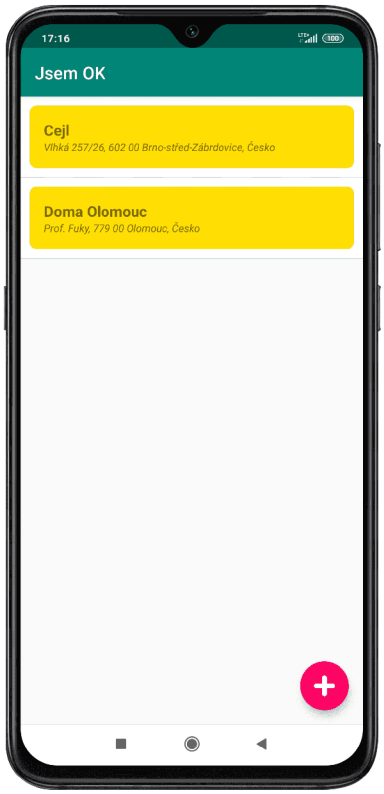
Klíčovým úkolem bylo zprovoznění služby *geofencingu*, aby bylo možné sledovat aktuální polohu (první zkušební verze byla z počátku testována prozatím na OS *Android* verzi 7, jelikož jsem používala telefon, který byl starší a neaktualizovaný). Bylo zvoleno pár pevně daných bodů na mapě, které byly v kódu definovány pomocí ručně zadané zeměpisné šířky a délky. Další potřebnou součástí aplikace byla funkční upozornění, informující o vstoupení či vystoupení ze zón. Nejdříve bylo potřeba otestovat klíčové části aplikace, než bylo možné produkt vyvíjet dále.

S devátou verzí OS *Android* byla aktualizována *Android Support Library*, zajišťující zpětnou kompatibilitu aplikace, na knihovnu *AndroidX*[5]. Ta obsahuje celou nahrazovanou knihovnu *Android Support*, a navíc nejnovější části *Jetpack* balíčku, který je kolekcí *Android* knihoven[6], jež usnadňují vývoj aplikací. Mapování jednotlivých původních balíčků *Support library*, na nové balíčky *AndroidX*, potřebných k implementaci, je popsáno v odkazu [4]. Návody, učící metodu *geofencingu*, nebyly vždy aktualizované. Bylo tudíž potřeba (často experimentálně) ověřovat, zda jsou zmiňované knihovny stále použitelné nebo je potřeba, aby byly nahrazeny aktuálnější verzí.

[4]Oficiální stránky *Figma*, pro tvorbu prototypů uživatelského rozhraní – https://www.figma.com/
[5]Oficiální stránky kolekce knihoven *AndroidX* – https://developer.android.com/jetpack/androidx
[6]Oficiální stránky kolekce knihoven *Jetpack* – https://developer.android.com/jetpack

5.3.2 Druhá fáze vývoje

V další fázi vývoje bylo cílem umožnit uživateli vytvářet zóny za běhu aplikace. Za tímto účelem byla potřebná implementace *Autocomplete Support Fragmentu*, jenž je součástí *Places API* od společnosti *Google*. Následně mohl uživatel vyhledávat místo, ve kterém chce založit zónu, na základě adresy. V této fázi však prozatím nebyla implementována mapa, na které by bylo možné si vytvořenou zónu následně zobrazit. Jakmile aplikace začala umožňovat zadávání dat, bylo zapotřebí vytvořit úložiště, ve kterém by mohly být informace trvale uloženy – začátek práce s *SQLite* databází. Dále byly rozšířeny možnosti zadávání dalších informací, specifikujících tvořenou zónu (její přezdívka, jež byla následně používána jako identifikátor zóny, poloměr velikosti zóny, i určení momentu, ve kterém si přeje uživatel být upozorněn). Se zadanými daty se však v této fázi vývoje aktivně nepracovalo (viz snímek obrazovky 5.3).



Obrázek 5.3: **Druhá fáze vývoje aplikace.** Na snímku obrazovky lze vidět, že byly implementovány pouze základní funkce. Design byl prozatím řešen jen minimálně, lze si všimnout, že implementováno nebylo ani tlačítko pro aktivaci a deaktivaci cíle. Tlačítko pro vytvoření zóny bylo prozatím umístěno v pravém dolním rohu. Taktéž horní lišta prozatím neobsahuje žádná další tlačítka pro doplňkové funkce.

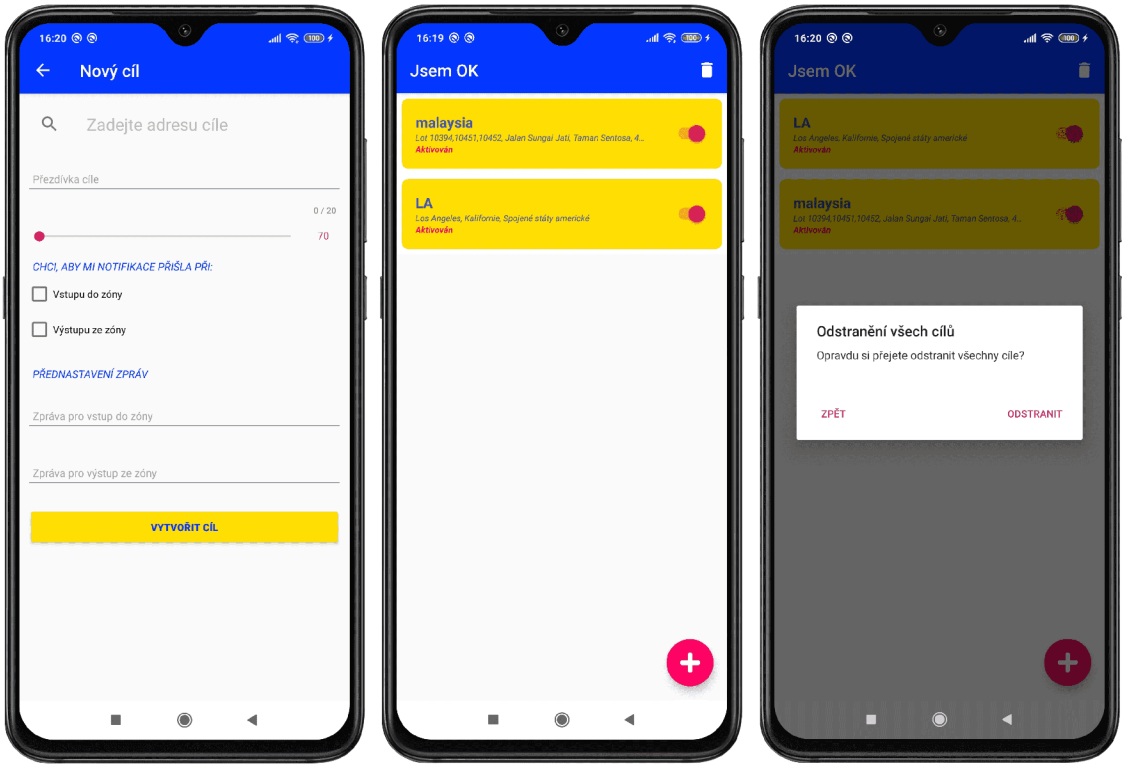
5.3.3 Třetí fáze vývoje

Třetí fáze se již více věnuje komplexnějším operacím. V průběhu jejich implementace vyšlo najevo, že *List View*, struktura, která byla původně zvolena pro zobrazení jednotlivých zón

v seznamu, je zastaralá a nepraktická. Proto byla již v pokročilém stádiu vývoje aplikace přepracována na novější strukturu – *RecyclerView* (popsána níže, v podkapitole 6.1.2).

Zajištění běhu aplikace na pozadí je ve třetí fázi vývoje zajištěno pomocí služby *Foreground Service*, která je spuštěna s otevřením aplikace a ukončena s odstraněním aplikace ze seznamu běžících aplikací.

V této fázi již bylo možné pracovat s daty, získanými od uživatele. Ovšem *Autocomplete Support Fragment*, sloužící k vyhledání adresy, byl v této fázi prozatím součástí jiné obrazovky a celkově nebyl spojen s mapou (ta nebyla vůbec implementována). Bylo možné deaktivovat a opět aktivovat zóny (pomocí takzvaného *toggle buttonu*). Při deaktivaci je u dané zóny změněna hodnota políčka `active` v tabulce zón a samotná oblast je odstraněna z takzvaného *Geofencing klienta*, čímž přestává být monitorována. Při aktivaci je daná zóny do *Geofencing klienta* opět přidána. Aplikace byla rozšířena o funkci smazání všech zón najednou – pomocí ikony koše v pravém rohu horní lišty. Prvek, takzvaný *floating button* (na snímcích obrazovky viz 5.4, zobrazen jako růžové kolečko se symbolem plus uprostřed), který slouží pro přidávání cíle, je prozatím umístěn v pravém dolním rohu obrazovky.



Obrázek 5.4: **Třetí fáze vývoje aplikace.** Na prvním snímku obrazovky je patrné, že v této fázi vývoje byl fragment pro vyhledávání adresy součástí aktivity bez mapy (ta ještě nebyla implementována), druhý snímek vyobrazuje vstupní aktivitu aplikace, s tlačítky pro aktivaci a deaktivaci monitorování zóny, dále v pravém horním rohu na liště ikonu odpadkového koše, sloužící ke smazání všech doposud vytvořených zón najednou – tato akce je doplněna kontrolním dotazem – možné vidět na posledním snímku obrazovky.

5.3.4 Finální podoba aplikace

Finální verze byla zveřejněna v obchodě *Google Play* jakožto *beta verze* – určena pro testování. Díky tomuto kroku mi od testujících přišlo plno názorů a chyby odhalujících postřehů, na které jsem do té doby nepřišla (často způsobeno zobrazením na jiných typech mobilního zařízení).

Co se týče inovací – oproti předchozí fázi vývoje se změnil princip běhu aplikace. Doposud aplikace fungovala tím způsobem, že byl *Foreground Service* (a společně s ním i celkový běh aplikace) ukončen s odstraněním aplikace ze seznamu běžících aplikací. Jelikož má aplikace sloužit také zapomnětlivým jedincům, je nutné počítat i s tím, že by si aplikaci v potřebnou chvíli zapomněli zapnout. Proto bylo fungování vyvíjené aplikace pozměněno – *Foreground Service*, zajišťující běh aplikace na pozadí, je zapnut s otevřením aplikace (tato část zůstala beze změny). Ovšem běh aplikace není vypnut s jejím odstraněním z běžících aplikací, ale poloha je neustále obnovována a monitorována. Pro její kompletní ukončení je potřeba manuálně stisknout vypínací tlačítko v aplikaci (monitorování je možné obnovit stiskem stejného tlačítka nebo znovuotevřením aplikace).

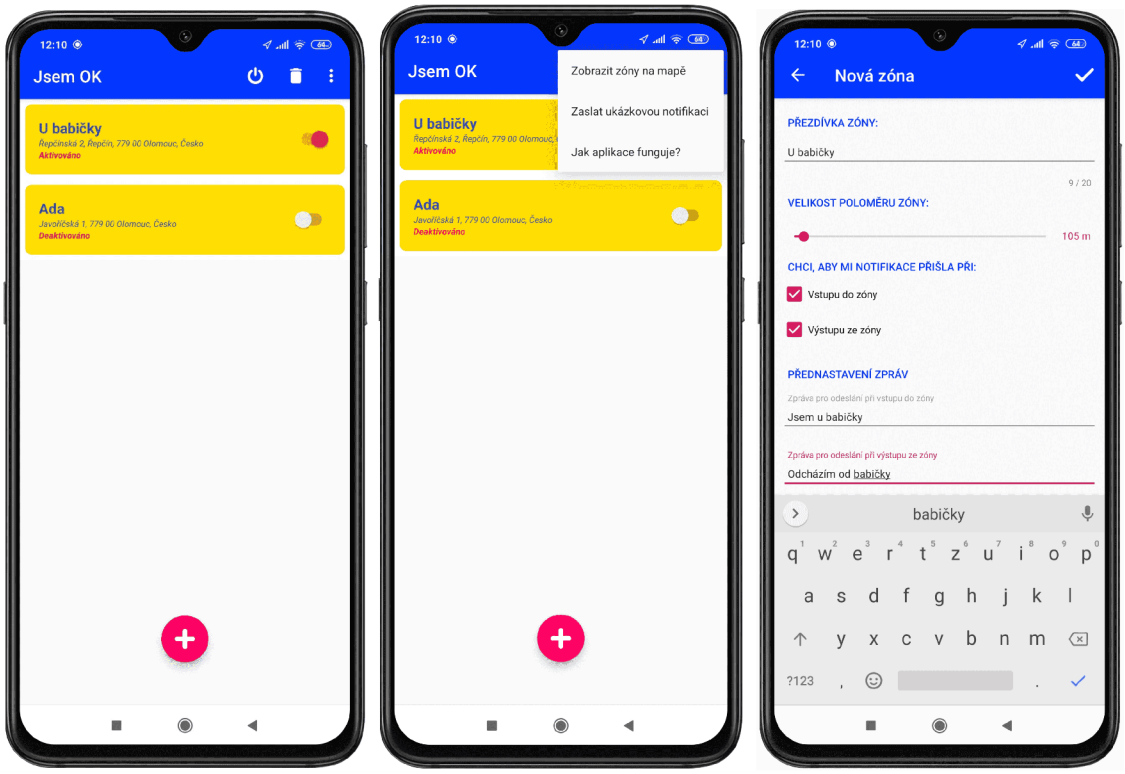
V této fázi vývoje se také poprvé objevuje fragment mapy (viz podkapitola 6.5), který je slouží k vykreslení vyhledané adresy a následnému zobrazení již dříve vytvořených zón.

Původně bylo tlačítko pro vytvoření nové zóny umístěno v pravém dolním rohu obrazovky. Nevšimla jsem si totiž, že když uživatel přidá více zón (a seznam dosáhne až ke spodnímu okraji obrazovky), tlačítko pro vytvoření nové zóny zastiňuje u jistých oblastí tlačítko pro aktivaci/deaktivaci monitorování dané oblasti. Proto bylo tlačítko, pro přidání nové oblasti, přesunuto do středu spodní části obrazovky.

Zpětnou vazbou jsem získala několik zajímavých nápadů a uživatelských vhledů. Jeden z testujících mě přivedl na myšlenku přidání možnosti odeslání testovací notifikace. Pomocí testovací notifikace si může uživatel vyzkoušet chování aplikace bez toho, aniž by musel změnit svou polohu vzhledem k monitorované zóně. Další uživatel mi sdělil, že se mu při vytváření nové zóny nezobrazuje tlačítko pro její uložení (běžně v dolní části obrazovky). To vedlo k úpravě rozložení obrazovky a jejímu nastavení jako „skrolovatelné“. Navíc jsem v částech aplikace, ve kterých je potřeba něco potvrdit, přidala do pravého horního rohu obrazovky ikonu fajfky – pokud si uživatel nevšimne tlačítka pro potvrzení (může být překryto klávesnicí), tuto ikonu se stejnou funkcionalitou uvidí vždy. Poslední zajímavé povšimnutí testerů bylo následné – ve chvíli, kdy uživatel zakládá novou zónu, má možnost předvolit si text automaticky generovaných zpráv. Vlivem nepozornosti při zadávání textu může dojít k záměně dvou pod sebou se nacházejících řádku pro texty zpráv. Aplikace tak dovolí vyplnit textovou zprávu pro vstup do zóny, i když uživatel uvedl, že chce být informován pouze o výstupu. Tuto nejednoznačnost jsem eliminovala následujícím způsobem – pokud uživatel zaškrtne pouze políčko „informuj mě pouze o vstupu do zóny“, zápis do textového pole pro výstup ze zóny není povolen. Pokud si uživatel bude přát dostávat zprávy jak o vstupu do zóny, tak výstupu z ní, možné neúmyslné záměně textových polí nelze programově zabránit.

V neposlední řadě byly v této fázi vývoje vytvořeny ikony aplikace.

Snímky obrazovky (viz 5.5) jsou již výsledkem, upraveným na základě zpětné vazby.



Obrázek 5.5: **Čtvrtá fáze vývoje aplikace.** Na prvním snímku obrazovky lze pozorovat více změn. Zaprvé – tlačítko, sloužící pro přidávání nové zóny je posunuto z pravého dolního rohu doprostřed (kvůli překrývání se s tlačítky deaktivačními). Dále je možné vidět ikony v pravém rohu na horní liště – zleva tlačítko pro kompletní vypnutí aplikace, koš pro odstranění všech cílů najednou a tři tečky, které po jejich rozbalení poskytnou další menu. Na dalším snímku je vidět právě ono rozbalené menu, které skýtá možnost „Zobrazit zóny na mapě“, „Zaslat ukázkovou notifikaci“ či „Jak aplikace funguje“ – tato možnost poskytne informační dialog s pravidly pro používání aplikace. Na posledním snímku obrazovky je vykresleno zadávání informací o vytvářené zóně – oproti minulé fázi vývoje chybí *Autocomplete Support Fragment*, jelikož adresa byla vybrána již ve fragmentu mapy. Jak lze na snímku vidět, tlačítko pro potvrzení vytvoření nové oblasti, je překryto klávesnicí. I když v této fázi je obrazovka již „skrolovatelná“ (a tudíž by bylo možné obrazovku posunout a tlačítko zobrazit), byla do pravého horního rohu obrazovky pro jistotu umístěna potvrzující fajfka, kterou lze použít za stejným účelem.

Kapitola 6

Implementace jednotlivých částí aplikace

V této kapitole je podrobněji rozebrán způsob implementace vyvíjené aplikace *Jsem OK*. Fáze implementace by správně měla přijít na řadu až po důkladně propracovaném prototypování, a jeho iteracemi vylepšené finální verzi. Pokud je „kódování“ aplikace započato před ustálením návrhu, implementační změny jsou poté časově několikanásobně náročnější.

6.1 Architektura aplikace

Jelikož aplikace vznikala iteračně, kdy první zkušenosti s programováním aplikací přicházely až s počátkem vývoje produktu, návrh architektury nebylo možné implementovat v plné podobě. Nasazená finální verze je tudíž kompromisem mezi představou o stoprocentně správně strukturované architektuře a postupně vyvíjenou verzí produktu.

6.1.1 Návrh architektury MVVM (*Model – View – ViewModel*)

Obecně architektonický návrhový vzor je vhodný pro jednodušší pozdější úpravy a celkovou udržitelnost kódu. Pokud uživatel nevyužije návrhového vzoru, bude struktura aplikace postavena na třídách aktivit. To znamená nadměrné množství řádků kódu v jednotlivých zdrojových souborech. Údržba a další rozvoj kódu v budoucnosti se tak stávají zbytečně pracnými a komplikovanými.

Rozhodnutí právě pro návrhový vzor *MVVM* [13] nebylo nahodilé. Svou podstatou je podobný návrhovému vzoru *MVP* (*Model-View-Presenter*), ovšem lze se domnívat, že v budoucnu bude návrhový vzor *MVVM* uplatňován ve větší míře – již nyní je používán jak hybridní platformou *Xamarin*, tak *iOS*. Další výhodou *MVVM* oproti *MVP* je záruka, že *View* existuje (díky využívání třídy *LiveData*) či menší množství kódu – architektonický návrhový vzor *MVVM* na rozdíl od vzoru *MVP* nevyžaduje implementaci velkého množství rozhraní. Grafické vysvětlení návrhového vzoru *MVVM*, viz obrázek 6.1.

Návrhový vzor *MVVM* se skládá ze tří částí:

1. ***Model*** – v jistých článcích také označován názvem *Datamodel*, jelikož abstrahuje data z databáze. *ViewModel* pracuje s *Modelem* za účelem získání a uložení získaných dat. Tato část návrhu (*Model*) by měla správně obsahovat veškerou logiku aplikace.

2. ***View*** – má za úkol informovat *ViewModel* o uživatelem provedených akcích. V aplikaci reálně představuje uživatelské rozhraní (*aktivitu*, *fragment* nebo jiný *view*). Jeho úkolem je být co nejvíce odlehčený, zastávat pouze vizuální stránku aplikace.

3. ***ViewModel*** – představuje abstrakci části *View*. Po získání potřebných dat z *Model*u, a aplikování logiky uživatelského rozhraní, poskytuje relevantní data části *View*. Data jsou poskytována proudy událostí (pro okamžitou informovanost se používá třída *LiveData*, která automaticky upozorňuje na změny), které jsou získávány pomocí *Observable*s (součást knihovny *RxJava*). *ViewModel* nemá žádné reference na *View*, pouze poskytuje informace anonymnímu příjemci. *View* může odebírat informace od nespočetně mnoha *ViewModel*ů, avšak *ViewModel* poskytuje informace vždy konkrétnímu *View*.

Obrázek 6.1: **Grafické vysvětlení vztahů v architektonickém návrhovém vzoru *MVVM (Model-View-ViewModel).***

6.1.2 Recyclerview

V počátku byla aplikace kvůli nedostatku zkušeností postavena na špatném předpokladu. Původně byl seznam zón promítán do aktivity za pomocí formy nazývané *ListView*, jež je předchůdcem *RecyclerView*. Tato komponenta byla zastaralá, a některé problémy tudíž nebyly vůbec řešitelné (a pokud ano, tak zbytečně komplikovaně).

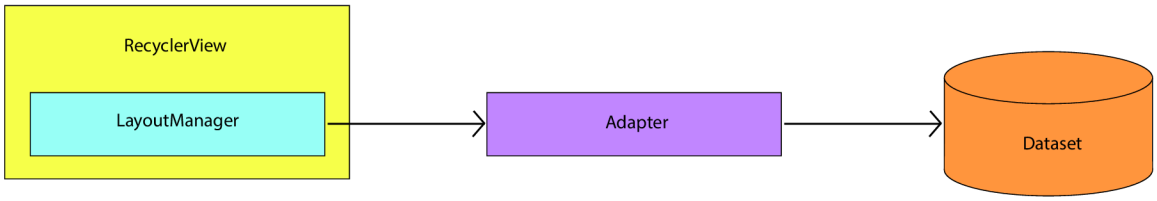
Posléze byl tedy z důvodu praktičnosti kód refaktorován tak, aby implementoval strukturu *RecyclerView*u (viz. názorné video vysvětlující implementaci a následné použití [6]).

Postup pro správné použití *RecyclerView*u:

1. Vytvoření či získání dat, kterými je potřeba strukturu naplnit
2. Definice *Adapter*u – jaké komponenty budou požadovány pro jeho sestavení
3. Vytvoření *RecyclerView*u v jazyce *XML* a *Kotlin*
4. Propojení *RecyclerView*u a definovaného *Adapter*u

Výhody této formy:

- Důležitou roli zde hraje takzvaný *ViewHolder*, který uchovává reference na jednotlivé položky struktury – když je pak třeba vytvořit novou položku, tak je buďto vytvořena nová nebo je znovu použita již existující, jejíž informace momentálně nejsou zobrazovány,
- v rámci této formy, která pojí několik položek dohromady, existují předem nastavené animace.



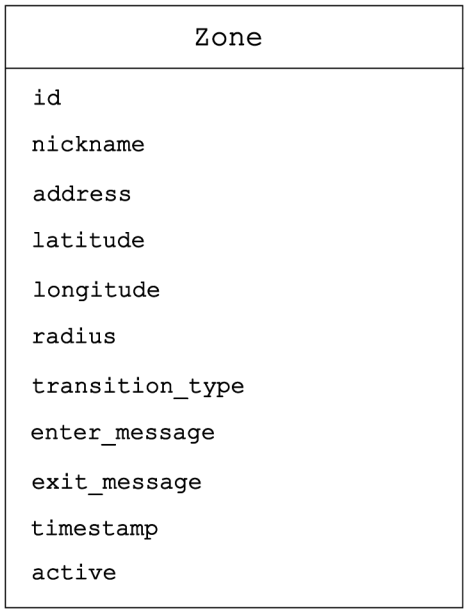
Obrázek 6.2: **Ilustrační schéma použití *RecyclerView*u.** Znázornění vztahů mezi jednotlivými bloky.

6.2 Rozložení databáze

Aplikace má za úkol uchovávat informace pouze o zónách náležících jednomu uživateli (neschraňuje žádná data o uživateli samotném či nepropojuje další jedince používající aplikaci). Tudíž bylo dostačující v rámci *SQLite* databáze vytvořit jedinou tabulku (viz obrázek 6.3), držící specifikace zón.

Vysvětlení k položkám držených tabulkou *Zone*:

- *id* – primární klíč položky v tabulce zón.
- *nickname* – přezdívka zóny, která představuje druhý unikátní identifikátor (možno nazývat druhým kandidátským klíčem). V aplikaci je používána pro adresování jednotlivých zón. Její jedinečnost je zajištěna v části přidávání nové zóny.
- *address* – jednoduše adresa, která je zvolena při vytváření zóny a později již není možné ji změnit (nepředpokládá se, že by se adresa pro jedno místo s časem měnila).
- *latitude* – zeměpisná šířka dané zóny.
- *longitude* – zeměpisná délka dané zóny.
- *radius* – představuje poloměr velikosti zóny, tzn. v jaké vzdálenosti od zvolené adresy bude uživatel upozorněn na její opuštění či vstup.
- *transition_type* – uchovává informaci o tom, zda si uživatel přeje být upozorněn na vstup do zóny či výstup z ní. Může nabývat hodnoty 1 (= vstup do zóny), 2 (= výstup ze zóny) nebo 3 (= vstup i výstup ze zóny). Vždy musí být zvolena minimálně jedna možnost.
- *enter_message* – text zprávy, který bude uživatelovi automaticky umístěn do odesílané zprávy po tom, co klikne na příchozí notifikaci při vstupu do zóny.
- *exit_message* – text zprávy, který bude uživatelovi automaticky umístěn do odesílané zprávy po tom, co klikne na příchozí notifikaci při výstupu ze zóny.
- *timestamp* – časové razítko je prvně zaznamenáno při vytvoření zóny. Poté je obnovováno pokaždé při jeho aktivaci či deaktivaci. Slouží pro řazení zón v seznamu dle jejich posledního použití.
- *active* – tento atribut hodnotami 0 a 1 vyjadřuje, zda je vybraná zóna momentálně aktivovaná či ne.



Obrázek 6.3: **Grafické znázornění jediné tabulky (obsahující informace o zónách) v rámci *SQLite* databáze.** K čemu slouží jednotlivé atributy, je vysvětleno výše, viz 6.2.

6.3 Implementace jednotlivých částí metody *geofence*

Existuje několik způsobů, jak postupovat při implementaci metody *geofencing*u (viz článek [9]). Jednotlivé způsoby se liší dle specifických účelů aplikací, které této metody využívají.

Jednotlivé kroky implementace:

1. Prvním krokem implementace metody *geofence* je získání oprávnění od uživatele a jejich kontrola. Jelikož je ve vyvíjené aplikaci využíváno služeb *GPS*, je potřeba nejdříve zkontrolovat, zdali uživatel udělil povolení k zjišťování polohy pomocí služeb určování polohy *Google*. V případě, že uživatel neměl povolenu službu zjišťování polohy, zobrazí se mu dialog požadující udělení povolení k zapnutí. Až ve chvíli, kdy je povolení uděleno, je možné přistoupit k nastavení ostatních oprávnění.

2. Ve chvíli, kdy uživatel povolí službu zjišťování polohy, a udělí další oprávnění potřebná ke správnému fungování aplikace (běh na pozadí musí být uživatelem povolen vždy, ne jen při používání aplikace), je spuštěno aktivní získávání polohy pomocí služby na popředí (takzvaný *Foreground Service*). Poloha je pravidelně získávána ve vhodně zvoleném intervalu – kompromis mezi spotřebou baterie a včasném informování uživatele (funkce pro zjišťování polohy je uvedena na obrázku 6.5). Lokace je zjišťována za pomoci *Fused Location Provider* klienta[1]. Jedná se o *API* umístění, spadající pod *Google Play* služby, kombinující různé signály (*GPS, WiFi*, atd.) za účelem poskytnutí aktuální polohy. Před obnovou aktuální polohy je potřeba připojit k aplikaci lokalizační služby a vytvořit žádost o polohu (*Location Request*). Za pomoci *Locati-*

[1] *Fused Location Provider API* – https://developers.google.com/location-context/fused-location-provider

onRequest je možné specifikovat parametry, se kterými je poloha získávána – přesnost či interval obnovy lokace.

3. Následně je zapotřebí vytvořit *Pending Intent*, který představuje možnost zacházet s případnou detekcí vstupu do monitorované zóny či výstupu z ní. V této aplikaci je vytvořen *Pending Intent* pro *Broadcast Receiver*, který v momentě detekce vyvolá akci – spustí *Job Intent Service*, ve kterém dochází ke zpracování podnětu a následnému uvědomění uživatele.

4. V dalším kroku se již přistupuje k samotnému vytváření oblasti zájmu, kterou bude posléze možné monitorovat. Zóna se vytváří a následně sleduje pomocí klienta (*GeogencingClient*), díky kterému lze komunikovat s *API geofencingu*. Samotný klient je vytvořen v metodě `onCreate()`, která je volána při prvním otevření aplikace. Před vytvořením další oblasti zájmu je potřeba zkontrolovat, zdali nebyl dosažen maximální možný počet sledovaných zón (100 pro aplikaci na jednom zařízení). Po provedení kontroly je možné zavolat metodu `addGeofence()` – kód této metody, viz obrázek 6.4. Při vytváření oblasti je potřeba specifikovat její `ID`, adresu, která je určena `zeměpisnou šířkou` a `délkou`, `radius`, který se okolo adresy vytvoří a bude následně reagovat na překročení této pomyslné hranice. Dále je zapotřebí určit, zda se má monitorování vytvářené oblasti po jisté době deaktivovat či ne. Jako poslední se musí stanovit, zda má být monitorován `vstup` do oblasti, `výstup` z ní nebo obojí.

5. V tento moment je již oblast zájmu monitorována. Pokud aplikace zaznamená vstup do zóny či výstup z ní, následující operace jsou prováděny v rámci služby *Job Intent*, která je vyvolána díky dřívějšímu nastavení *Pending Intent* (na rozdíl od normálního *Intent*, není proveden v okamžik vytvoření, ale v tomto případě až v momentě zaregistrování průchodu přes hranici zóny). Ve vyvolané službě *Job Intent* lze v rámci předdefinované metody `onHandleWork(intent: Intent)` získat pomocí `GeofencingEvent.fromIntent(intent)` případ, který službu vyvolal. S těmito získanými informacemi se pracuje dále – získání *typu průchodu* (vstup/výstup), *id*, *názvu* zóny. Tato služba taktéž obstarává zasílání notifikací.

6. Monitorování oblasti je následně možné dočasně deaktivovat nebo trvale odstranit. V rámci *GeofencingApi* neexistuje metoda, která by sledování zóny dokázala pouze dočasně deaktivovat. Proto je pro deaktivaci nutné uchovat informace o oblasti v dočasném úložišti (v případě této aplikace se jedná o *SQLite* databázi) a následně odstranit z *GeofencingApi*. Při příští aktivaci je zapotřebí získat informace o oblasti z databáze a pomocí metody `addGeofences()` oblast znovu přidat (způsob přidávání popsán již v bodě číslo 4).

6.4 Notifikace

Notifikace v této aplikaci hrají podstatnou roli – jejím prostřednictvím se uživatel dozvídá o vstupu do zóny či výstupu z ní. Zároveň slouží jako prostředek pro odeslání informační zprávy. Ukázka příchozí notifikace a jejího použití je předvedena na obrázku 6.6

Postup při nastavení notifikací se odvíjí od konkrétního výrobce, jelikož každý má jinak vypadající prostředí systémového nastavení. Vývojář tudíž v této situaci nemůže udělit jednotnou radu, která by znamenala řešení pro všechny uživatele. V některých případech

může mít uživatel zakázaná vyskakovací okna, zvuk či vibrace notifikací – mohlo by se tak stát, že by si příchozího upozornění nevšiml včas.

6.4.1 Změna implementace notifikací pro OS *Android* verze 8.0 (API level 26)

Pokud má vyvíjená aplikace fungovat i na operačním systému *Android* verze 8.0 a vyšší, je potřeba přidat část kódu, týkající se vytvoření *notifikačního kanálu*[2]. Bez této úpravy by se na mobilních telefonech s vyšší verzí OS notifikace nezobrazovaly. Část kódu, která musela být přidána do stávajícího programu, viz obrázek 6.7. Následně je možné pro každý *notifikační kanál* nastavit vizuální a zvukové chování, které je posléze aplikovatelné na všechny notifikace stejného typu. Uživatel tak může ve správě aplikací jednoduše určit, které *notifikační kanály* v rámci dané aplikace by měly být viditelné, a které ztlumené.

Postup pro vytvoření *notifikačního kanálu*:

1. Konstrukce objektu *NotificationChannel* s unikátním *ID notifikačního kanálu*, uživateli zobrazovaným názvem a úrovní důležitosti (*importance level*).
2. Další krok je nepovinný – jde o nastavení popisu, který se bude zobrazovat uživateli při správě jednotlivých *notifikačních kanálů* v systémovém nastavení aplikace.
3. Posledním krokem je registrace *notifikačního kanálu* jeho předáním funkci *createNotificationChannel()*.

6.5 Google mapa

Mapa společnosti *Google* je součástí finální podoby aplikace (viz snímky obrazovky 6.8) – slouží k zobrazení adresy při vytváření nové zóny a jejímu případnému upřesnění potažením špendlíku na mapě. Následně je možné si všechny uživatelem vytvořené oblasti na mapě zobrazit. K jejich fungování je zapotřebí implementovat *Maps SDK* pro *Android*[3]. Toto *API* automaticky zajišťuje přístup k serverům *Google Map*, stahování dat, samostatné zobrazení mapy ve fragmentu aplikace a reakci na gesta, pomocí kterých lze s mapou manipulovat. Dále lze využít přidání špendlíků na mapu a vykreslovat zóny kruhovitého či jiného tvaru (ve vyvíjené aplikaci využito pro zobrazení velikosti poloměru vytvořených zón).

[2] Kanály pro zasílání notifikací – https://developer.android.com/training/notify-user/channels

[3] Oficiální stránka *Maps SDK* pro *Android* – https://developers.google.com/maps/documentation/android-sdk/intro

```
fun addGeofence(zone: Zone) {
var geofenceToBuild: Geofence ?= null
    when(zone.transitionType){
        1 -> {
            geofenceToBuild = Geofence.Builder().setRequestId(zone.nickname)
                .setCircularRegion(zone.lat, zone.lng, zone.radius.toFloat())
                .setExpirationDuration(NEVER_EXPIRE)
                .setTransitionTypes(Geofence.GEOFENCE_TRANSITION_ENTER)
                .build()}
        2 -> {
            geofenceToBuild = Geofence.Builder().setRequestId(zone.nickname)
                .setCircularRegion(zone.lat, zone.lng, zone.radius.toFloat())
                .setExpirationDuration(NEVER_EXPIRE)
                .setTransitionTypes(Geofence.GEOFENCE_TRANSITION_EXIT)
                .build()}
        3 -> {
            geofenceToBuild = Geofence.Builder().setRequestId(zone.nickname)
                .setCircularRegion(zone.lat, zone.lng, zone.radius.toFloat())
                .setExpirationDuration(NEVER_EXPIRE)
                .setTransitionTypes(Geofence.GEOFENCE_TRANSITION_ENTER or
Geofence.GEOFENCE_TRANSITION_EXIT)
                .build()}
    }
    if(geofenceToBuild != null){
        mGeofencingClient.addGeofences(buildGeofencingRequest(geofenceToBuild),
getGeofencePendingIntent())?.addOnCompleteListener(this)?.run {
            addOnSuccessListener { showToast(getString(R.string.addGeofenceOK)) }
            addOnFailureListener { showToast(getString(R.string.addGeofenceFail)) }
        }
    }
}
```

Obrázek 6.4: **Fragment kódu pro přidávání oblastí zájmu.** Na tomto obrázku lze vidět, jak se ve volané funkci `addGeofence()` vytvoří nová zóna, která je následně monitorována metodou *geofencingu*. Pro její konstrukci je zapotřebí určit `ID` oblasti, `zeměpisnou šířku` a vzdálenost od bodu na mapě (tzv. `radius`), okolo kterého se vytvoří abstraktní hranice. Pomocí metody `.setExpirationDuration()` je stanoveno, po jak dlouhou dobu bude oblast monitorována bez deaktivace a jako poslední je nutno určit, zda má být uživatel upozorněn při `vstupu` do oblasti zájmu či `výstupu` z ní. Posléze má zóna daná specifika a stačí započít její monitorování – provede se přidáním právě vytvořeného *geofence* do *GeofencingClient* za pomocí *geofenciongRequest* a *Pending Intent*.

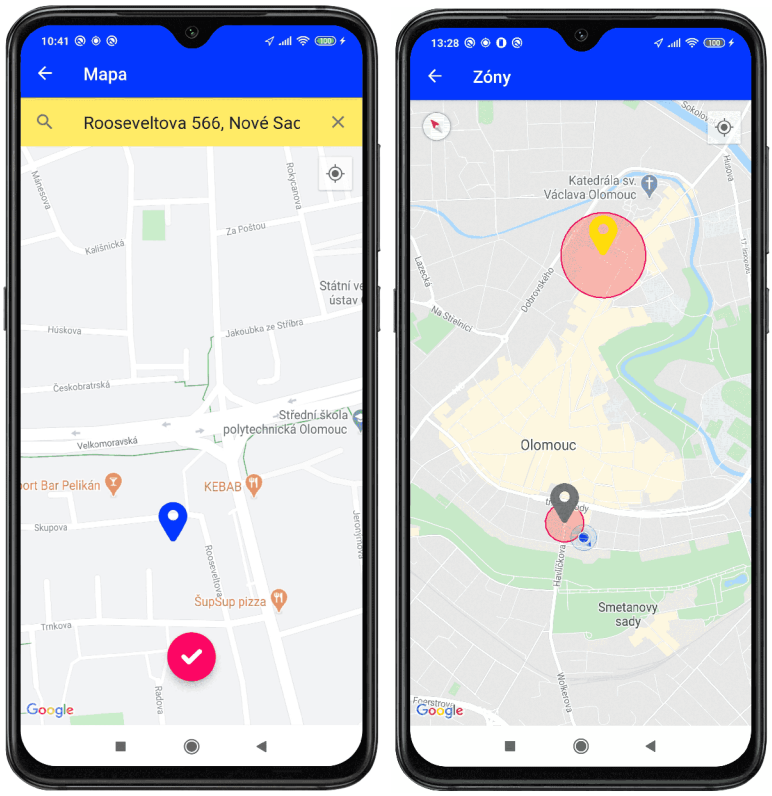
```
fun getLocation(){
  var fusedLocationProviderClient = LocationServices.getFusedLocationProviderClient(this)
  locationRequest = LocationRequest.create().apply {

     priority = LocationRequest.PRIORITY_HIGH_ACCURACY
     interval = 10000
     fastestInterval = 3000
  }

  LocationServices.getFusedLocationProviderClient(this)
     .requestLocationUpdates(locationRequest, locationCallback, Looper.getMainLooper())
}
```

Obrázek 6.5: **Fragment kódu pro zjišťování aktuální polohy v rámci běhu aplikace na popředí.** Aktuální poloha je opakovaně získávána pomocí *Fused Location Provider* klienta, jehož funkcionalita je již popsána výše, viz bod 2. Na tomto obrázku kódu je mimo jiné předvedena specifikace požadavků na žádost o polohu. Jedním požadavkem je `priorita` – v tomto případě nastavena na `PRIORITY_HIGH_ACCURACY`, což znamená, že je získávána co nejpřesnější poloha – za pomoci *GPS, WiFi, mobilní sítě* a jiných *senzorů* (může představovat zátěž na baterii, ovšem pokud je potřeba získávat přesnou lokaci, je to nevyhnutelné). Dalším požadavkem je `interval`, který určuje v jednotkách milisekund, v jakém časovém intervalu bude získávána aktuální poloha pro aplikaci a `fastestInterval` určuje časový interval získávání polohy z jiných aplikací.

Obrázek 6.6: **Postupné fáze od příchodu notifikace aplikace *Jsem OK*, až po odeslání zprávy vybraným způsobem.** Na prvním snímku obrazovky lze vidět příchozí notifikaci, na které je vysvětleno, zda se jedná o vstup do zóny nebo výstup z ní, plus přezdívka zóny. Pod ní je zobrazena notifikace, která uživatele obeznamuje s během aplikace na pozadí. Na druhém snímku je takzvaný *Share button*, který nabízí způsoby, jakými je možné informační zprávu odeslat, uložit do poznámek a jiné. Na posledním snímku obrazovky je text zprávy předvyplněn přednastaveným textem pro vstup do dané zóny.

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    // Create the NotificationChannel
    val name = getString(R.string.channel_name)
    val descriptionText = getString(R.string.channel_description)
    val importance = NotificationManager.IMPORTANCE_DEFAULT
    val mChannel = NotificationChannel(CHANNEL_ID, name, importance)
    mChannel.description = descriptionText
    // Register the channel with the system; you can't change the importance
    // or other notification behaviors after this
    val notificationManager = getSystemService(NOTIFICATION_SERVICE) as NotificationManager
    notificationManager.createNotificationChannel(mChannel)
}
```

Obrázek 6.7: **Nutná úprava kódu pro *API 26* a výše.** Vytvoření notifikačního kanálu je nutností pro vylepšení OS *Android*. V jiném případě by notifikace nebyly zobrazovány.



Obrázek 6.8: **Využití fragmentů map ve výsledné aplikaci** Na prvním snímku je vidět mapa v rámci přidávání zóny, v jeho horní části se nachází *Autocomplete Support Fragment*, sloužící pro vyhledávání místa na základě adresy. Modrý špendlík, který se na mapě objeví po vyhledání určité adresy, lze následně dlouhým podržením a tahem přemístit, a adresu tak upřesnit. Druhý snímek již vykresluje vytvořené zóny – aktivní je značena barvou žlutou, neaktivní šedou.

Kapitola 7

Nasazení a výsledky testování aplikace

Testování v průběhu vývoje aplikace nesmí být podceňováno. Než může být zveřejněna finální verze projektu, měla by aplikace projít **minimálně** jedním kolem testování, na kterém se bude podílet více uživatelů (takzvaných testerů). Nezáleží ovšem jen na náhodných testujících a jejich pocitech při prvním pohledu na aplikaci. Je také potřeba vytvořit jisté datové statistiky, zkoumat kvalitu napsaného kódu. V rámci této kapitoly bude vysvětleno, jaké formy testování existují, jaké z nich byly použity při testování této aplikace a jaké zajímavé poznatky přinesla zpětná vazba.

7.1 Formy testování aplikace[1]

Testování aplikace musí probíhat na různých zařízeních z důvodu zajištění kompatibility. Musí být zohledněny různé značky výrobců, lišící se operační systémy (dále jejich různé verze a následné aktualizace) nebo odlišné hardwarové konfigurace (velikost obrazu, klávesnice).

Před nasazením aplikace je potřeba ji podrobit testování s ohledem na více aspektů. Prvním z nich je **použitelnost**, která vypovídá o tom, že je aplikace lehce použitelná a splňuje uživatelova očekávání. Dalším aspektem vhodně otestované aplikace by měla být **kompatibilita**, což je již dříve zmíněné fungování produktu na odlišných zařízeních s jinými podmínkami. Dále je testováno **uživatelské rozhraní**, u kterého se sleduje správné a smysluplné rozvržení tlačítek a celková propracovanost logické struktury aplikace. Je také zapotřebí otestovat **služby** aplikace v online a offline režimu. **Výkon** aplikace je stále důležitějším aspektem, jelikož je plno produktů s podobnou myšlenkou, a v ten moment jde o to, který z nich je navržen nejefektivněji (spotřeba baterie, snadné ovládání, atd.). **Provoz** by měl být bezproblémový a plynulý – nemělo by se stát, že dojde ke ztrátě dat (ať už při vypnutí zařízení, v důsledku vybití baterie či aktualizace aplikace). Bezchybně by měla taktéž fungovat **instalace** a **odinstalace** aplikace ze zařízení. V neposlední řadě by měla být vždy stoprocentně zajištěna bezpečnost dat uživatele a zachování důvěrnosti obsahu.

Samotné testování lze provádět třemi možnými způsoby – zaprvé je to testování prováděné za pomocí reálných zařízení (výhoda skutečných operačních systémů, zkoumaných v reálném čase), zadruhé pouze využitím emulátoru (lze testovat na různých verzích plat-

[1] Článek o testování mobilních aplikací – https://pixelfield.cz/blog/co-musite-vedet-o-testovani-mobilnich-aplikaci/

formy, navíc je tato metoda nákladově méně náročná, pokud je aplikovaná ve větším měřítku), nebo do třetice lze použít kombinace obou předchozích možností.

7.2 Zveřejnění na Google Play

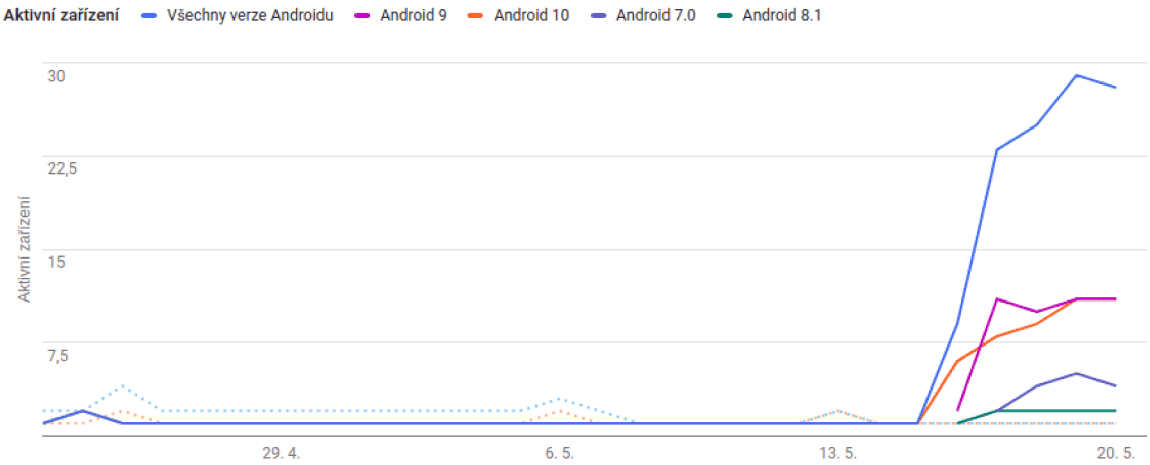
Aby mohla být aplikace testována reálnými uživateli, byla zveřejněna na stránkách *Google Play*. Před zveřejněním oficiální verze aplikace je možné uveřejnit takzvanou *beta verzi* – jedná se o předběžný přístup k aplikaci, který lze udělit testerům pomocí odkazu.

Jakkoliv jednoduchý se může zdát níže uvedený postup pro uvedení aplikace na trh skrz obchod *Google Play*, orientace na jednotlivých platformách (*Google Play Console*, *Google Console*) zabere vskutku více času, než by člověk čekal. Tento návod [5], formou videa, na stránkách *youtube* dokázal proces značně ulehčit.

Jakožto *beta verze* byla zveřejněna poslední fáze vyvíjené aplikace. Její odkaz jsem sdílela na sociálních sítích a ve výsledku si aplikaci stáhlo celkem 28 lidí. Nárůst uživatelů se předpokládá po oficiálním uveřejnění aplikace na trh.

Postup pro zveřejnění aplikace na *Google Play*:

1. Vytvoření účtu na platformě *Google Play* (jednorázový registrační poplatek 25$ USD)
2. Založení profilu aplikace pomocí *Google Play Console* – při vytváření profilu je nutné přiložit logo, snímky obrazovky dané aplikace, jistý popis aplikace
3. Zveřejnění aplikace vytvořené v *Google Play Console* na stránkách *Google Play*



Obrázek 7.1: **Graf počtu stažení *beta verze* aplikace v obchodě *Google Play*.** K datu 22. 5. 2020 aplikaci využívalo 28 uživatelů.

7.3 Získaná zpětná vazba

Na základě zpětné vazby ze strany testerů, bylo provedeno několik změn. Zde budou uvedeny pouze ve formě výčtu, jelikož podrobněji popsány jsou již v podkapitole 5.3.4, pojednávající o inovacích ve finální verzi.

Změny v aplikaci na základě zpětné vazby:

1. Přidání možnosti odeslání testovací notifikace,
2. úprava obrazovky, aby byla nadále „skrolovatelná“,
3. přidání potvrzujícího tlačítka na horní lištu,
4. propojení zaškrtávacích políček a textových polí pro vstup do zóny či výstup z ní při vytváření či úpravě oblastí zájmu,
5. u jistých nadpisů změna z italického písma na normální styl (na uživatele působilo matoucím dojmem),
6. změna barvy pozadí u *Autocomplete Support Fragmentu*, za účelem zvýšení kontrastu při zobrazení v mapě.

7.4 Další možný vývoj

Jak již bylo několikrát v této práci zmíněno (např. viz 2.2), vyvíjená aplikace dbá na uživatelovo soukromí a bezpečí. Za tímto účelem nejsou zprávy odesílány automaticky – pokud bude o tuto doplňující funkcionalitu ze strany uživatelů zájem, je možno ji dodatečně přidat.

Jednoduchost aplikace je stavena na téměř stejnou úroveň důležitosti, jako soukromí a bezpečí uživatele. Případné rozšíření aplikace o počítadlo, které by uživateli poskytovalo informaci o tom, kolikrát danou lokalitu navštívil, by jednoduchost nemělo narušit, a proto by mohlo být případně zvažováno.

Aplikace byla záměrně cílena na český trh – proto je prozatím dostupná pouze v českém jazyce. Do budoucna je však zvažován překlad do jazyka anglického, možná i francouzského.

Samozřejmě si všímám, že je trh dělen na dvě poloviny – první skupina používá OS *Android* a druhá *iOS*. Proto bych v budoucnu ráda aplikaci rozšířila i na platformu *iOS*, za účelem oslovení širšího spektra uživatelů. Také by bylo vhodné vytvořit jednoduchou webovou prezentaci aplikace, která by potenciálním uživatelům názorněji na příkladech vysvětlila, k čemu je produkt vhodný.

Možné úpravy se netýkají pouze technických vylepšení. Jelikož aplikace má pomáhat zejména mladším uživatelům, přemýšlela jsem o možném výběru kresleného panáčka s vhodným informačním textem – ten by byl zasílán místo strohé zprávy.

Kapitola 8

Závěr

Po porovnání podobných aplikací na trhu byl vyhotoven návrh aplikace a prostudovány nejvhodnější metody pro jeho realizaci. Další fáze vývoje představovala implementaci jednotlivých částí aplikace. V průběhu tvorby byl produkt průběžně testován lidmi z mého blízkého okolí, kteří byli ochotni poskytovat zpětnou vazbu. Po strastiplném období, kdy metoda *geofencingu* nefungovala kvůli změnám v aktualizovaném OS *Android*, nakonec vše fungovalo podle očekávání. Ve finální fázi vývoje byla aplikace umístěna v testovacím režimu do obchodu *Google Play* – po získání názorů z širšího okolí a následném doladění různých prvků byla aplikace oficiálně uvedena na trh.

Výsledkem mé práce je mobilní aplikace[1], která uživatele upozorní na vstup do dané oblasti či naopak výstup z ní a poskytne mu snadný způsob informování dalších osob. Produkt dbá na uživatelovo soukromí a celkovou jednoduchost aplikace. Jakožto propagační a vysvětlující materiál byl vytvořen plakát a video – obojí je možné najít na přiloženém CD, video bylo navíc sdíleno na stránkách youtube[2]. Nápady na další možný vývoj produktu jsou uvedeny v předchozí podkapitole, viz 7.4.

Závěrem bych ráda poznamenala, že jsem si samozřejmě uvědomovala, i před psaním této práce, že v dnešním světě technologií je tempo vývoje neskutečně rychlé. Doposud jsem vždy programovala pouze školní projekty. Tudíž jsem nikdy předtím nepracovala na jednom produktu po delší dobu, než jsou zhruba dva měsíce. Teprve tato zkušenost mi ukázala **skutečné** tempo vývoje a odhalila zajímavý pohled na, pro mě donedávna neprobádanou, oblast informačních technologií.

[1] Aplikace *Jsem OK* v obchodě *Google Play* – https://play.google.com/store/apps/details?id=com.jsemok

[2] Propagační a vysvětlující video na stránkách youtube – https://youtu.be/Vxb2eV3YLs0

Literatura

[1] *Android developers - Access location in the background.* [Online; navštíveno 12.02.2020].
URL https://developer.android.com/training/location/background

[2] *Android developers - Background Location Limits.* [Online; navštíveno 10.02.2020].
URL https://developer.android.com/about/versions/oreo/background-location-limits

[3] *Android developers - Create and monitor geofences.* [Online; navštíveno 16.03.2020].
URL https://developer.android.com/training/location/geofencing

[4] *AndroidX: Class Mappings.* [Online; navštíveno 20.01.2020].
URL https://developer.android.com/jetpack/androidx/migrate/class-mappings

[5] *How to publish Android apps on Google play - Step by Step guide.* [Online; navštíveno 28.03.2020].
URL https://www.youtube.com/watch?v=AWawL5HFn64

[6] *Simple RecyclerView in Kotlin (2020 with AndroidX).* [Online; navštíveno 14.03.2020].
URL https://www.youtube.com/watch?v=afl_i6uvvU0

[7] *Služby na popředí.* [Online; navštíveno 16.04.2020].
URL https://docs.microsoft.com/cs-cz/xamarin/android/app-fundamentals/services/foreground-services

[8] *Zdroják - Flutter.io – mobilní aplikace, znovu a lépe.* [Online; navštíveno 21.03.2020].
URL https://www.zdrojak.cz/clanky/flutter-io-mobilni-aplikace-lepe/

[9] *Codelab: Geofences. https://www.raywenderlich.com/7372-geofencing-api-tutorial-for-androidtoc-anchor-001*,
2018.

[10] *Will my geofencing function in the background? https://proximi.io/will-my-geofencing-function-in-the-background/*, květen 2018.

[11] Bareth, U.: Simulating Power Consumption of Location Tracking Algorithms to Improve Energy-Efficiency of Smartphones. In *2012 IEEE 36th Annual Computer Software and Applications Conference*, IEEE, 2012, ISBN 9781467319904, s. 613–622.

[12] Luboslav Lacko: *Vývoj aplikací pro Android.* Computer Press, 2015, ISBN 978-80-251-4347-6.

[13] Muntenescu, F.: *Android Architecture Patterns Part 3: Model-View-ViewModel. https://medium.com/upday-devs/android-architecture-patterns-part-3-model-view-viewmodel-e7eeee76b73b*, november 2016.

[14] Owens, M.: *The Definitive Guide to SQLite*. Apress, 2006, ISBN 978-1-59059-673-9.

[15] Smith, S.: *Introduction to Android Activities with Kotlin. https://www.raywenderlich.com/*, květen 2019.

[16] Srinivasan, V.: *Android O — Work around Background Service Limitation. https://medium.com/@debuggingisfun/android-o-work-around-background-service-limitation-e697b2192bc3*, october 2018.

Příloha A

Obsah CD

Přiložené CD obsahuje zdrojové soubory a jiné části bakalářské práce. Níže vykreslená struktura mapuje umístění nejdůležitějších adresářů a souborů.

```
/
├── README.txt - informace o jednotlivých adresářích a souborech
├── jsemok-documentation/ - zdrojové soubory této technické zprávy ve formátu LaTeX
├── jsemok-app/ - zdrojové soubory mobilní aplikace
├── jsemok-release.apk - instalační soubor mobilní aplikace
├── jsemok-documentation.pdf - tato technická zpráva ve formátu PDF
└── graphic/
    ├── screenshots/ - snímky obrazovky jednotlivých fází vývoje aplikace
    ├── jsemok-poster.pdf - plakát pro prezentaci aplikace
    └── jsemok-video.mp4 - video pro prezentaci aplikace
```