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 SDÍLENÍ INFORMACE O PŘÍTOMNOSTI NA MÍSTĚ

MOBILE APP FOR SHARING INFORMATION ABOUT PRESENCE IN A LOCATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **TOMÁŠ VLK**
AUTHOR

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

BRNO 2017

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Vlk Tomáš**
Obor: Informační technologie
Téma: **Mobilní aplikace pro sdílení informace o přítomnosti na místě**
Mobile App for Sharing Information about Presence in a Location
Kategorie: Uživatelská rozhraní

Pokyny:

1. Prostudujte a popište existující mobiliní aplikace pracující se sdílením polohy a přítomnosti a obdobná řešení.
2. Definujte požadavky na funkčnost aplikace pro sdílení informace o přítomnosti na místě.
3. Prototypujte prvky uživatelského rozhraní a funkčnosti řešené aplikace. Testujte je na uživatelích.
4. Vytvořte řešenou aplikaci, testujte ji na uživatelích a iterativně ji zdokonalujte.
5. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2, značné rozpracování bodu 3, případně i bodu 4.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Herout Adam, prof. Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Cílem této práce je vytvořit uživatelsky přívětivou mobilní aplikaci pro systém Android, díky níž bude snazší sdílet svoji polohu a zároveň zachovat své soukromí. Místo GPS souřadnic aktuální polohy sdílí aplikace pouze informaci o přítomnosti na určitém místě. Pro snazší sdílení nabízí také automatickou detekci přítomnosti na daném místě. Výsledkem práce je aplikace s uživatelským rozhraním respektujícím Material Design, díky níž lze určit přítomnost zařízení na místě s poloměrem větším než 50 metrů, ale zároveň nedochází k aktivnímu dotazování polohy. Podařilo se tak vytvořit aplikaci, která nemusí být vůbec spuštěna (ani žádná její služba na pozadí) a přitom dokáže detekovat přítomnost zařízení na určitém místě a přijímat zprávy od ostatních zařízení. Největší předností tohoto přístupu je velmi nízká spotřeba energie oproti jiným řešením.

Abstract

The aim of this work is to create a user-friendly mobile application for Android that will make it easier to share ones's location while keeping their privacy. Instead of the GPS coordinates of the current location, the application only shares the information about the user's presence at the place. For easier sharing, it also offers automatic detection of the presence at a specific place. The result of this work is a application with a user interface that respects the Material Design, which makes it possible to determine the device's presence at a place with a radius greater than 50 meters, but there is no active polling. The application which has been created does not need to be run at all (or any of its background services) and it can detect the device's presence at a specific place and receive messages from other devices. The biggest advantage of this approach is very low energy consumption in contrast to other solutions.

Klíčová slova

Android, Firebase, Poloha, Geofence, Sdílení

Keywords

Android, Firebase, Location, Geofence, Sharing

Citace

VLK, Tomáš. *Mobilní aplikace pro sdílení informace o přítomnosti na místě*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, Ph.D.

Mobilní aplikace pro sdílení informace o přítomnosti na místě

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana prof. Ing. Adama Herouta, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Vlk
13. května 2017

Poděkování

Chtěl bych poděkovat prof. Ing. Adamu Heroutovi, Ph.D za jeho profesionální přístup a cenné rady při vytváření i propagace aplikace. Dále bych chtěl poděkovat Pavlu Peškovi za pomoc při vytváření ikony a propagační grafiky aplikace, a své rodině a přátelům, kteří mě podpořili a pomáhali mi při testování aplikace.

Obsah

Kapitola 1

Úvod

Lidé chtějí vědět, kde se nachází jejich blízcí a známí, ale ne každý chce svou polohu sdílet. Tato práce vznikla s cílem tento problém vyřešit a najít kompromis mezi těmito extrémy. V práci je popisován vývoj aplikace pro systém Android, která přistupuje k této problematice jinak než ostatní aplikace dostupné na této platformě. Aby uživatelé měli nad svým soukromím kontrolu, nesdílí aplikace GPS souřadnice aktuální polohy, ale pouze informaci o přítomnosti na určitém místě. Dalším stupněm ochrany soukromí je definování osob u každého místa, kterým tato informace bude sdílena. Důležitou součástí aplikace je pak také automatická detekce přítomnosti na místě a sdílení této informace. Uživateli tak velmi zjednodušuje ovládání.

Výsledná aplikace tak ulehčí život lidem, kteří opakovaně píší zprávy svým blízkým o své přítomnosti na určitém místě. Příkladem mohou být děti informující své rodiče o své přítomnosti doma nebo ve škole. Aplikace ale není zaměřena jen na rodinné příslušníky, a tak ji může využít kdokoliv.

Ke správnému fungování aplikace bylo třeba vytvořit systém automatické detekce přítomnosti na místě tak, aby byl spolehlivý a co nejméně energeticky náročný. Další klíčovou věcí je zprostředkování informace o přítomnosti na místě ostatním uživatelům. K tomuto účelu musí být v celém systému zapojený server, ke kterému se klientské aplikace uživatelů budou připojovat. Aplikace disponuje možností zapnutí notifikací při jakékoliv změně u míst. Bylo tedy třeba také vyřešit, aby se jakákoliv událost na serveru zprostředkovala na ostatní klientské aplikace, s důrazem kladeným na nízkou spotřebu energie.

Kapitola 2

Existující aplikace pro sdílení polohy

Sdílením polohy pomocí mobilních aplikací se zabývalo již několik projektů jako například projekt *Locaccino* [16]. Analýza používání této služby ukázala, že uživatelům nevadí sdílení jejich přítomnosti v místech navštěvovaných velkým a různorodým souborem lidí. Časem mají tito lidé tendenci vyvíjet sofistikovanější nastavení ochrany soukromí, což se odráží v nárůstu časových a lokalizačních omezení [15].

Většina existujících aplikací dostupných v obchodě Google Play nabízejících sdílení polohy funguje na principu sdílení GPS souřadnic v reálném čase. Aplikace tohoto typu často nabízejí možnost omezit přístup k poloze jen určitým lidem nebo jen po určitý čas. Příkladem mohou být služby *Glympse*[1] a *Life360*[2]. Nově pak tuto funkci nabízí i *Google Maps*[3] nebo *Facebook Messenger*[4]. Tyto aplikace nejsou jediné dostupné, ale jsou to nejpopulárnější řešení dané problematiky. Mají miliony stažení. Aplikace *Google Maps* a *Facebook Messenger* mají dokonce více než miliardu stažení v obchodě Google Play. Lze tak porovnat řešení jejich uživatelského prostředí a způsoby zabezpečení těchto populárních aplikací, protože k těmto existujícím řešením má tato aplikace „nejblíže“.

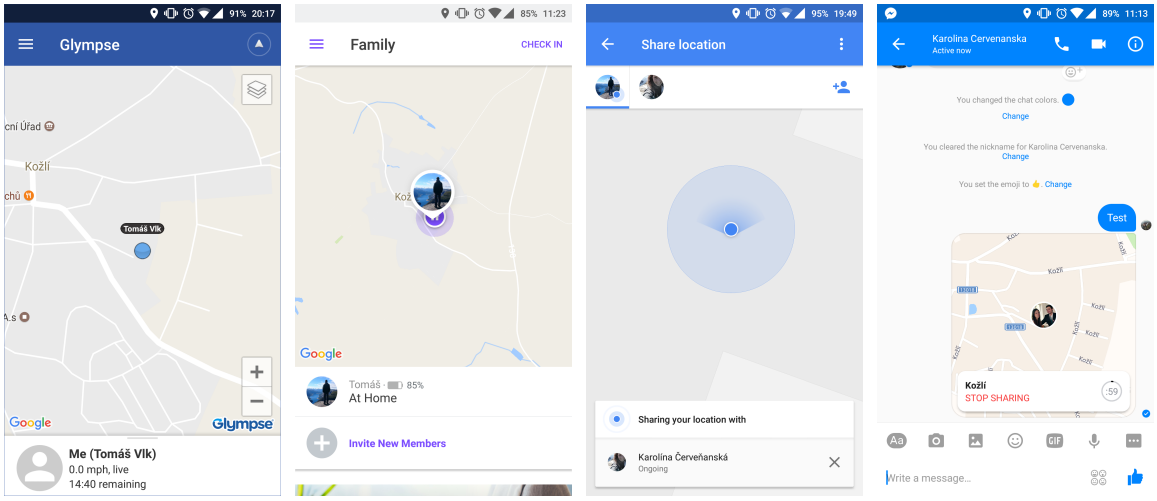
2.1 Glympse

Uživatel v této službě není autentizován. Aplikace proto vůbec nevyžaduje vytvoření účtu nebo přihlášení po prvním spuštění. V aplikaci je možnost sdílet polohu ihned po kliknutí na ikonu v pravém rohu, jenž lze vidět na obrázku 2.1 vlevo. Následně lze definovat osoby, čas sdílení a doplňující zprávu. Po potvrzení je cílovým osobám rozeslán odkaz na webovou stránku obsahující mapu se sdílenou polohou. Cílová osoba není tedy vůbec autentizována.

2.2 Life360

Po prvním spuštění aplikace musí uživatel vytvořit účet u služby *Life360*. Je zvláštní, že aplikace požaduje při vytvoření účtu zadat telefonní číslo, ale toto číslo není službou ověřováno a ani nelze zobrazit u ostatních lidí. Aplikace také vyžaduje zadání emailu, který

[1] Glympse – https://play.google.com/store/apps/details?id=com.glympse.android.glympse
[2] Life360 – https://play.google.com/store/apps/details?id=com.life360.android.safetymapd
[3] Google Maps – https://play.google.com/store/apps/details?id=com.google.android.apps.maps
[4] Facebook Messenger – https://play.google.com/store/apps/details?id=com.facebook.orca



Obrázek 2.1: Uživatelská rozhraní existujících aplikací – zleva *Glympse*, *Life360*, *Google Maps*, *Facebook Messenger*

také není ověřován. Při registraci musí uživatel zadat heslo k účtu. Mobilní číslo a email pak pravděpodobně nahrazují uživatelské jméno. Následně se uživatel může připojit ke kruhu s lidmi, nebo vytvořit kruh s lidmi. Tento kruh má jedinečný identifikátor a jednoho administrátora. Kdokoliv, kdo zná tento identifikátor, má možnost se připojit ke kruhu a vidět tak polohu ostatních. Připojit se ale může jen osoba s účtem služby *Life360*. Poloha všech uživatelů je pak sdílena nepřetržitě a lze ji omezit pouze opuštěním daného kruhu lidí nebo vypnutím sdílení pro daný kruh. Uživatelské rozhraní je velmi podobné službě *Glympse* a je zachyceno na obrázku 2.1.

2.3 Google Maps

Tato aplikace není primárně určena pro sdílení polohy, ale tuto funkci nabízí. Uživatel je autentizován pomocí účtu Google. V aplikaci je možnost sdílet svou polohu v levém menu nebo po kliknutí na svou polohu. Následně lze definovat čas sdílení u každé osoby, kterým bude poloha sdílena. Lze nastavit, aby toto sdílení bylo i bez časového omezení. Pokud je poloha sdílena uživateli Google, tak je tento uživatel autentizován. Jinak je vygenerován odkaz, který již autentizován není. Odkaz směřuje na webové stránky služby *Google Maps* obsahující mapu a zobrazenou sdílenou polohu. Uživatelské rozhraní aplikace *Google Maps* je zachyceno na obrázku 2.1.

2.4 Facebook Messenger

Jak z názvu aplikace vyplývá, je tato aplikace určena k zasílání zpráv. Nabízí ale odeslání speciální zprávy s aktuální polohou uživatele. Uživatel je autentizován účtem služby Facebook. V aplikaci je možnost v konverzaci s jiným uživatelem nebo skupinou uživatelů sdílet svou polohu. Časové omezení sdílení je pevně nastaveno na 60 minut a nelze jej nastavit jinak. Následně je uživatelům doručena zpráva o sdílení polohy obsahující mapu a aktuální polohu uživatele. Tato speciální zpráva je vidět na obrázku 2.1 vpravo. Cílové osoby tedy jsou autentizovány účtem Facebook.

2.5 Vlastnosti existujících aplikací

Výhodou aplikací *Glympse* a *Google Maps* je možnost jednoduše poslat odkaz, přes který je sdílená poloha zobrazena přímo na webových stránkách. Cílová osoba tedy vůbec nemusí mít aplikaci nainstalovanou ani vlastnit žádný účet a i tak se dozví vše podstatné. Cílí tak na uživatele, kteří potřebují jednorázově, rychle a jednoduše sdílet svou polohu jakékoliv osobě. Uživatel ale nemá kontrolu nad tím, kdo ho může vidět. Kdo zná odkaz, může vidět aktuální polohu uživatele. Na druhou stranu uživatel, který takovýmto způsobem sdílí polohu, určitě není na místě, na kterém nechce být viděn a nebojí se tak o svoje soukromí. Aplikace *Facebook Messenger* poskytuje podobně rychle a jednoduše sdílet polohu ale jen uživatelům Facebooku, což zvyšuje zabezpečení. Aplikace *Life360* jako jediná nenabízí možnost časově omezit sdílení polohy a funguje nepřetržitě. Cílí tak například na rodiče a jejich děti. Její velkou výhodu je, že rodiče můžou kdykoliv nalézt své děti, pokud se například ztratí.

Ne vždy je ale tento přístup ideální. Sdílet někomu GPS souřadnice své aktuální polohy je velký zásah do soukromí. Omezení sdílení polohy po určitý čas může tento problém částečně řešit, ale vyžaduje aktivní zapojení uživatele. Uživatel musí aplikaci otevřít a nastavit čas, po který se poloha bude sdílet a případně osoby, kterým tato poloha bude sdílena. Ze zmíněných aplikací se nejvíce odlišuje *Facebook Messenger*, kde je poloha sdílena jako zpráva a nelze nijak nastavit, tudíž je to pro uživatele otázka dvou kliknutí. Na druhou stranu, pokud by uživatel potřeboval sdílet polohu déle než 60 minut, musel by periodicky toto sdílení obnovovat.

Z výše zmíněných aplikací ale nenabízí žádná sdílení pouze informace o přítomnosti na místě. Všechny fungují na principu sdílení GPS souřadnic. Sdílet svou přítomnost na místě může uživatel i jiným způsobem, například pomocí textové zprávy. Při zkoumání i ostatních podobných aplikací jsem ale nenalezl žádnou, která by nabízela **automatické** sdílení informace o přítomnosti na místě.

Kapitola 3

Návrh služby Inspotis

V předchozí kapitole byly představeny aplikace, které sdílejí GPS souřadnice. V určitých případech tak mohou nechtěně zasahovat do soukromí uživatele. Cílem aplikace Inspotis je proto nabídnout uživatelům možnost sdílet pouze informaci o jejich přítomnosti na místě a dát jim tak velkou kontrolu nad svým soukromím. Ke sdílení informace o přítomnosti na místě lze přistoupit dvěma způsoby – manuálně nebo automaticky.

Při manuálním sdílení musí uživatel sám iniciovat zprostředkování informace o své přítomnosti na místě daným lidem. Uživatel může využít sociálních sítí a napsat „status“ nebo může využít nějaké služby pro zasílání zpráv a dát tak slovně nebo vizuálně vědět, kde zrovna je. Pokud uživatel provádí tuto činnost opakovaně u určitého místa, bude pro něj jednodušší provést celou tuto operaci jedním kliknutím. V tomto případě musí být ale předem definováno, komu má být informace o přítomnosti na místě sdílena. Tato služba se nesnaží nahradit statusy nebo zprávy s jednorázovou informací o přítomnosti na místě a cílí tak na uživatele, kteří chtějí sdílet tuto informaci opakovaně u určitého místa.

Automatické sdílení tento proces pro uživatele značně zjednoduší. Uživatel nemusí memorovat, že má dát někomu informaci o přítomnosti na místě. Automatické detekce místa lze dosáhnout prostředky mobilních zařízení k lokalizaci [2]. Problém je ale, která místa identifikovat a kterým osobám automaticky sdílet tuto informaci. Řešením je, že si uživatel předem nastaví daná místa zájmu a také osoby u těchto míst, kterým bude automaticky zprostředkována informace o přítomnosti uživatele na místě.

Sdílení informace o přítomnosti na místě je bezvýznamné, pokud se nedostane k cílovým osobám. Aplikace popisované v kapitole 2 tuto informaci zprostředkovávají cílovým osobám dvěma způsoby. První je bez autentizace cílové osoby (například odkaz na webové stránky obsahující mapu s aktuální polohou). Druhý způsob je autentizace cílové osoby v aplikaci. Cílem služby Inspotis je dát uživateli větší kontrolu nad svým soukromím. Sdílení informace o přítomnosti bez autentizace by tedy bylo proti tomuto principu, proto budou cílové osoby ve službě Inspotis autentizovány. K autentizaci cílové osoby se dá také přistoupit několika způsoby. Já jsem zvolil autentizaci přímo v aplikaci. To znamená, že cílová osoba musí mít aplikaci nainstalovanou a být v ní přihlášena (autentizována). Výhodou tohoto přístupu je, že komunikovat mezi sebou budou jen aplikace služby Inspotis a jiné služby (například služby na zasílání zpráv, nebo sociální sítě) nebudou mít přístup k informacím o uživatelích.

Aplikace služby Inspotis budou spolu komunikovat prostřednictvím sítě Internet. Mobilní zařízení ale nemají statickou IP adresu. Z toho vyplývá, že zařízení mezi sebou nemohou komunikovat *peer-to-peer* bez centrálního prvku. V celé architektuře služby Inspotis tak musí být zapojený server, ke kterému budou klientské aplikace připojeny a vůči kterému se budou autentizovat.

3.1 Cílová skupina

Různí uživatelé mohou mít různé požadavky na funkčnost aplikace. Jak již bylo zmíněno, cílem aplikace je nabídnout uživatelům možnost sdílet pouze informaci o přítomnosti na místě. Zároveň aplikace cílí na uživatele, kteří chtějí tuto informaci sdílet opakovaně na určitém místě, ať už manuálně nebo automaticky. Cílové osoby jsou při přístupu k informaci o přítomnosti na místě autentizovány klientskou aplikací – jsou tedy také uživatelé aplikace. Uživatel služby Inspotis tak může polohu sdílet, přijímat nebo obojí. Obecně pro uživatele této služby platí, že ji musí využívat s jinými lidmi. Je to tedy služba pro skupinu lidí, kteří chtějí mezi sebou sdílet polohu. Nedává smysl, aby aplikaci využíval jenom samotný uživatel. Potenciálními uživateli služby Inspotis mohou být například tyto skupiny lidí:

- **skupina přátel** scházející se často na jejich oblíbeném místě
- **spolubydlící** se mohou podívat, jestli je na bytě volno
- **rodič** se může podívat, jestli jsou **děti** ve škole/doma
- **spolupracovníci** si mohou zjistit, kdo je v práci

Zde jsem mohl jít dvěma cestami. Mohl jsem aplikaci zacílit jen na jednu cílovou skupinu uživatelů nebo ji navrhnout tak, aby pokud možno vyhovovala všem výše zmíněným cílovým skupinám. Rozhodl jsem se aplikaci zaměřit obecně. Aplikace se tak snaží uspokojit požadavky všech výše zmíněných skupin uživatelů.

Jednotlivé skupiny uživatelů ale mohou mít specifické požadavky na funkčnost aplikace. Uživatelé ze skupiny přátel chtějí vidět, kdo na daném místě je ale i kdo tam není. Zároveň je mezi nimi symetrická vazba. Pokud může uživatel A znát přítomnosti uživatele B na místě, bude uživatel B také chtít znát přítomnost uživatele A na místě.

U uživatelů z řad rodičů a spolupracovníků to může být jinak. Má-li rodič dvě děti, bude mu stačit vědět, které jeho dítě je doma a nepotřebuje zobrazit informaci, které dítě tam není. Opačně ke skupině přátel je mezi rodiči a dětmi vazba asymetrická. Rodič chce vidět přítomnost svého dítěte doma, ale dítě nemusí vědět, je-li rodič doma. Analogicky to tak může být i mezi spolupracovníky, pokud je některý z nich nadřízený jiným.

Rozdíl může být i v používání. Uživatel ze skupiny přátel se podívá, kdo je na daném místě, třeba jen jednou o víkendu. Pokud někdo z jeho známých bude často přítomen na daném místě, nebude potřebovat pokaždé znát tuto informaci. Pokud bude ale on přítomen na místě, měl by mít možnost to explicitně dát vědět ostatním. Na druhou stranu rodič nebo nadřízený možná bude chtít zkontrolovat stav i několikrát denně. Pro rodiče tak bude lepší pokud bude informován o svém dítěti automaticky. Jeho dítě by jinak muselo dávat svému rodiči každý den explicitně vědět, že je na místě.

Aplikace by tak měla nabízet možnost definovat symetrické i asymetrické vazby mezi uživateli. Měla by umět zobrazit, kdo je na místě a zároveň, kdo tam není. Uživatel by měl mít možnost explicitně dát vědět svou přítomnost na místě. Také by měl uživatel mít možnost dostávat automaticky informaci o změně stavu u určitého místa či osoby.

3.2 Autentizace uživatelů

Informace o přítomnosti uživatele na místě se musí dostat jen ke správným osobám a ostatním přístup zamezit. Cílová osoba tedy musí být autentizována a autorizována. Toho je

dosaženo autentizací každé osoby používající službu Inspotis při prvním spuštění aplikace. Uživatel pak definuje u každého místa seznam uživatelů služby Inspotis (autentizovaných osob), kteří mají přístup k jeho přítomnosti na místě. Tito předem definovaní uživatelé mají ke sdílené informaci pak přístup prostřednictvím aplikace, v níž jsou autentizováni. Bylo tedy třeba rozhodnout, jak bude uživatel autentizován. Aplikace pro platformu Android používají několik forem registrace a následné autentizace uživatelů. Mezi nejčastější patří:

1. pomocí uživatelského jména a hesla
2. pomocí emailu
3. pomocí účtu třetích stran (Facebook, Google+, ...)
4. pomocí telefonního čísla

Při výběru bylo nejdůležitější, co přináší nejvíce výhod pro uživatele. Tyto výhody lze definovat takto:

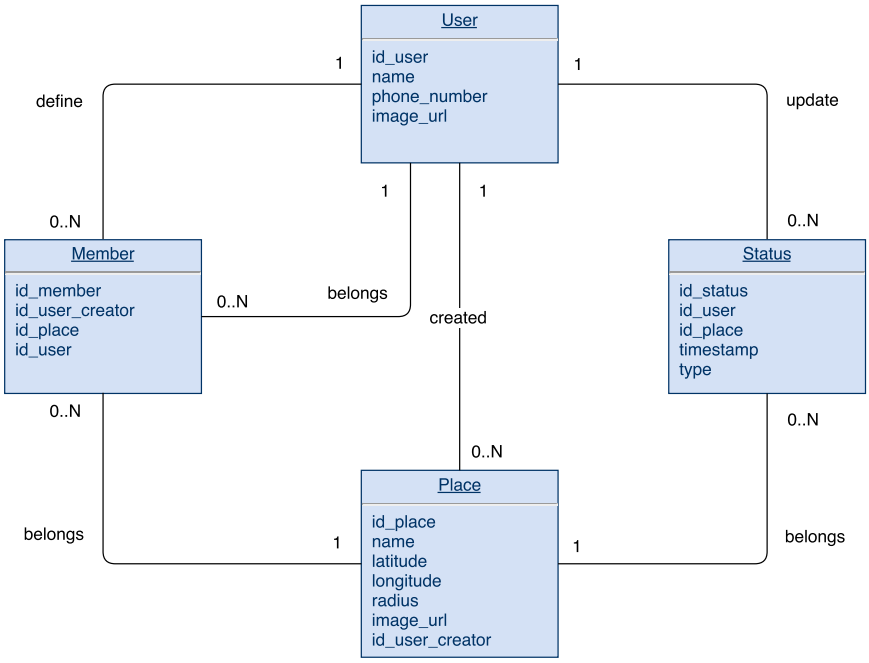
- je jednoduché pro uživatele přihlášení do aplikace
- je jednoduché pro uživatele nalézt ostatní uživatele

První a druhá možnost autentizace je pro uživatele náročná při prvním spuštění aplikace. Uživatel musí zadávat informace, které si pak musí pamatovat pro různé případy. Pamatování více různých hesel je složité, a proto 43–51 % uživatelů opětovně používá stejná hesla u různých služeb, což snižuje bezpečnost [13]. Pokud by se pak uživatel autentizovaný uživatelským jménem nebo emailem chtěl spojit s jiným uživatelem, musí zadat email nebo uživatelské jméno tohoto uživatele, což je další spousta informací, které uživatel odněkud musí získat nebo si pamatovat.

Třetí možnost je ideální pro rychlé přihlášení. Většina lidí má dnes na svém mobilu různé aplikace, do kterých jsou již přihlášeni. Uživatel při autentizaci potvrdí přihlášení pomocí třetí strany a vše je ihned zpracováno většinou pomocí protokolu OAuth 2.0[1] po několika kliknutích. Pokud nemá danou aplikaci třetí strany (nebo do ní není přihlášen), ale má u této služby účet, musí se do dané služby přihlásit. To ale není nová informace, kterou si musí uživatel pamatovat jako v prvních dvou případech. Velkou výhodou autentizace pomocí třetích stran je spojení uživatele s ostatními uživateli. Pokud má uživatel u dané služby nějaký seznam svých přátel, lze mu pak většinou přes API dané služby tento seznam zprostředkovat a uživatel si pak při nalezení jiného uživatele nemusí nic pamatovat. Problém je, že ne všichni uživatelé používají například jen Facebook nebo jen Google+, které tato API poskytují. Proto by uživatel mohl mít na výběr z více služeb, aby si mohl vybrat. Může se pak ale stávat, že uživatel přihlášený pomocí Facebooku chce nalézt někoho, koho má v přátelích na Facebooku, ale tento člověk je přihlášený do aplikace pomocí účtu Google+ a tudíž se nedokáží spojit. Příkladem může být Facebook Graph API, které umožňuje získat pouze přátele uživatele, kteří jsou do aplikace také přihlášeni pomocí Facebooku.

Poslední možnost je takový kompromis mezi všemi těmito možnostmi. Uživatel při přihlašování musí ověřit své telefonní číslo, což není tak jednoduché jako několik kliknutí v případě třetí možnosti. Nalezení uživatele je pak ale mnohem snazší díky telefonnímu seznamu uživatele, v kterém se dají pomocí telefonních čísel vyfiltrovat uživatelé používající tuto službu. Tento způsob je v poslední době častější typ autentizace. Příkladem mohou být velmi populární služby *WhatsApp*[2] nebo *Google Allo*[3]. Uživatel se přihlašuje v aplikaci

[1] OAuth 2.0 – https://oauth.net/
[2] WhatsApp – https://play.google.com/store/apps/details?id=com.whatsapp
[3] Google Allo – https://play.google.com/store/apps/details?id=com.google.android.apps.fireball



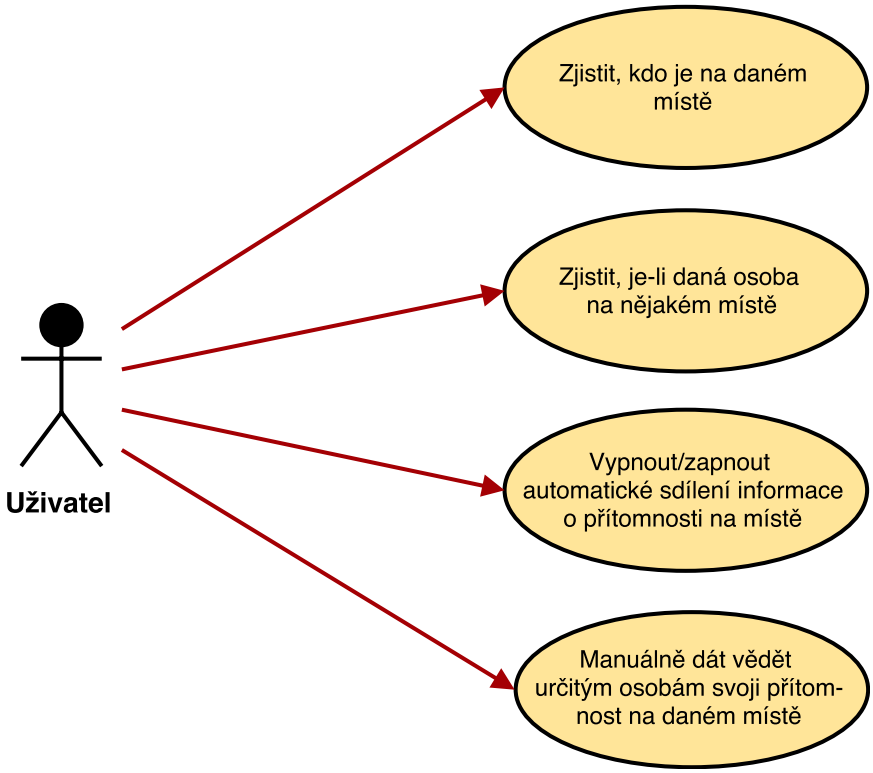
Obrázek 3.1: ER diagram serverové části. Uživatel (entita User) definuje ostatní uživatele (Member) u míst (Place) a zároveň může být přiřazen k místu jiným uživatelem. Uživatel může vytvořit vlastní místo a může nastavit status k jakémukoliv místu.

pouze jednou a vyhledávat ostatní uživatele bude častěji, proto jsem tedy zvolil autentizaci pomocí telefonního čísla.

3.3 Sdílení a ukládání informací o uživatelích

Předchozí text vysvětlil, jak bude aplikace komunikovat, jaké funkce musí nabízet a jak bude autentizovat uživatele. Je potřeba definovat, s jakými informacemi o uživateli bude služba pracovat. Informace o přítomnosti na místě není jediná informace, kterou lze zpřístupnit ostatním uživatelům. Lze přidat například časová razítka, kdy naposledy byl uživatel na místě, historii přítomnosti uživatele na místě, nečinnost v aplikaci a spoustu dalšího. Na druhou stranu tyto funkce zase mohou odhalovat další informace ze soukromí uživatelů. Proto tato aplikace, která chce zajistit uživatelům kontrolu nad svým soukromím, tyto funkce nenabízí. V základním návrhu tedy aplikace pouze sdílí informace o přítomnosti na místech uživatelů a žádná jiná soukromá data nebo historii. Pokud by uživatelé projevili o danou funkci zájem, může být dodatečně implementována jako volitelná funkce v dalších verzích. Serverová část tak musí být na tyto potenciální změny připravena anebo být alespoň jednoduše upravitelná, aby dokázala na tyto změny reagovat.

Informace o přítomnosti uživatele na místě je ve své podstatě pouze informace identifikující místo, nikoliv uživatele. Je proto lepší, pokud se celá detekce místa a rozhodnutí o tom, jestli je uživatel na místě, provede na mobilním zařízení a na server se bude sdílet poté už jen informace o přítomnosti na místě. To zvyšuje bezpečnost, kdyby se objevila chyba v serverové aplikaci, tak i tak nikdo nebude moci sledovat aktuální polohu uživatelů. Na serveru je pouze informace o stavu jednotlivých míst daného uživatele, nikoliv jeho GPS souřadnice. Na serveru ale musí být uložen seznam míst uživatele a jím definovaní uživatelé



Obrázek 3.2: Nejčastější případy užití klientské aplikace.

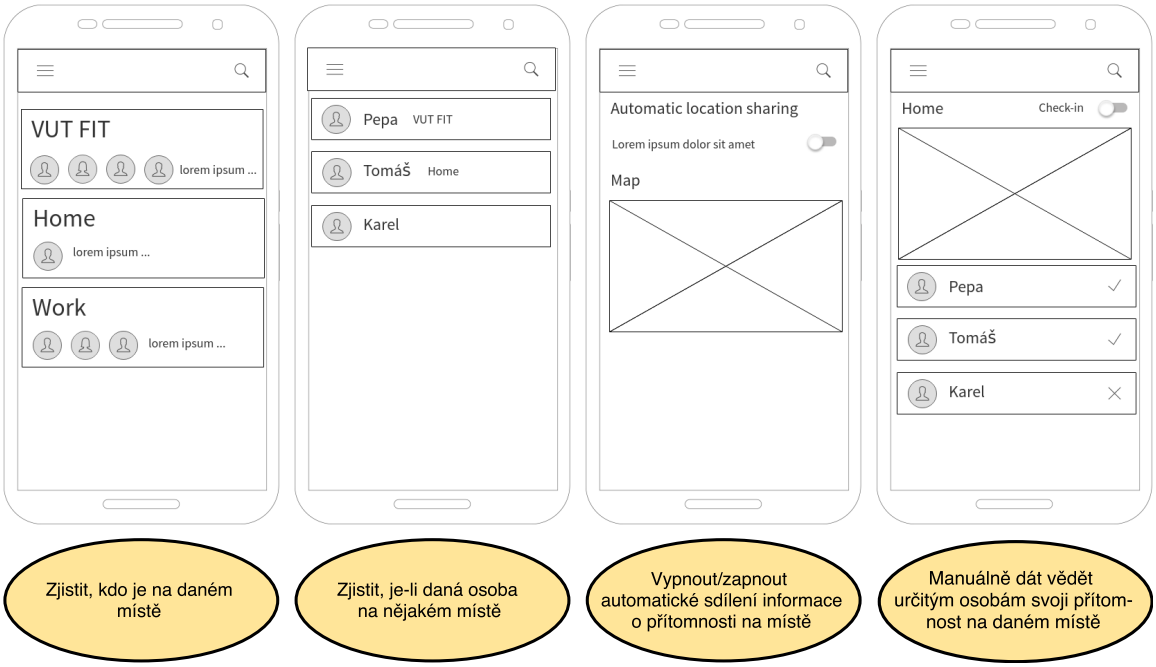
u těchto míst, aby mohli být autorizováni. Tito uživatelé pak mají přístup k informaci o přítomnosti uživatele na místě. ER diagram pro serverovou část je na obrázku 3.1. Databáze vyhovující tomuto diagramu umožňuje i případné uložení časového razítka, kdy byl uživatel na místě, nebo ukládání historie přítomnosti uživatele na místě.

3.4 Uživatelské rozhraní klientské aplikace

Součástí této práce bylo i navržení a vytvoření uživatelského rozhraní klientské aplikace tak, aby bylo co nejvíce intuitivní a zároveň dobře plnilo svou funkci. Uživatelské rozhraní plní své cíle, pokud jsou nejčastější případy užití pro uživatele jednoduché. Na základě informací o cílové skupině v sekci 3.1 byly identifikovány nejčastější případy užití uživatelů zobrazené na obrázku 3.2.

Hlavní funkcí, kterou aplikace nabízí, je automatická detekce přítomnosti na místě a zobrazení stavů ostatních uživatelů. Uživateli je třeba zobrazovat přítomnost uživatelů na jednotlivých místech a také přítomnost samotného uživatele na místě, pokud byla detekována. Uživatel musí mít kontrolu nad sdílením – jestli se sdílí a co se sdílí. Dále musí uživatelské rozhraní umožňovat uživateli pracovat (vytvářet, upravovat, odstraňovat) s jednotlivými místy a definovat u těchto míst uživatele, kteří budou mít přístup k informaci o přítomnosti uživatele na místě. Podle Kruga [14] má použitelné uživatelské rozhraní tyto atributy:

- Užitečné: Dělá to to, co uživatel potřebuje?
- Intuitivní: Vědí uživatelé, jak to použít?



Obrázek 3.3: Prvotní návrhy rozmístění ovládacích prvků a zobrazení kolekcí a struktur. Každý návrh by měl nabízet uživateli případ užití z obrázku 3.2 zobrazený pod daným návrhem.

- Nezapomenutelné: Musí se uživatel znovu naučit jak to používat, pokaždé když to použije?
- Efektivní: Dokáže to udělat?
- Účinné: Udělá to s rozumným časem a úsilím?
- Žádoucí: Chtějí to uživatelé?
- Nádherné: Je to příjemné, nebo dokonce zábavné?

Uživatelské rozhraní je navrženo tak, aby bylo použitelné (splňovalo předchozí atributy) a aby případy užití na obrázku 3.2 byly pro uživatele nejdostupnější. Prvotní návrhy uživatelského rozhraní a jejich význam je na obrázku 3.3. To, jestli uživatelské rozhraní opravdu splňuje atributy použitelnosti, lze ověřit na reálných uživatelích. Vytváření prototypů uživatelského rozhraní a testování použitelnosti je popisováno v kapitole 6.

Specifikum této aplikace je, že pracuje většinu času na pozadí. Nepředpokládá se, že by uživatel měl stále aplikaci zapnutou a pracoval s ní. V sekci 3.1 bylo zmíněno, že uživatelé budou mít možnost explicitně dát vědět svou přítomnost na místě. Také budou mít možnost dostávat automaticky informaci o změně stavu u určitého místa či osoby. Důležitým prvkem komunikace aplikace s uživatelem jsou proto notifikace. Klientská aplikace tak uživatele upozorní notifikací na explicitní informaci o přítomnosti na místě od jiného uživatele. Pokud uživatel chce dostávat automaticky informaci o přítomnosti jiných uživatelů u daného místa, bude mu tato informace doručena také pomocí notifikací.

Kapitola 4

Metoda pro detekci přítomnosti na místě

Klientská aplikace potřebuje poskytovat detekci co nejmenších míst a zároveň být co nejméně energeticky náročná. Každý mobilní telefon v současné době disponuje GPS modulem, který má přesnost přibližně několik metrů [12]. Zapnutí tohoto modulu je ale velmi energeticky náročné. V Androidu se dá určit poloha i z WiFi nebo mobilní sítě. Poloha z WiFi má přesnost zhruba 50 až 500 metrů, u mobilních sítí to je pak 500 metrů až několik kilometrů [12]. Kombinací všech zdrojů polohy (pokud jsou k dispozici) lze získat nejrychleji a nejpřesněji aktuální polohu. Proces získání polohy může aplikace urychlit a zefektivnit, pokud se společně s těmito zdroji polohy využijí například informace ze senzorů, poloha získaná jinými aplikacemi nebo historie polohy. Systémová komponenta Google Play Services nabízí různá rozhraní pro přístup k poloze, která tyto a další faktory berou v potaz. V následujícím textu jsou názvy programových `rozhraní` odlišeny.

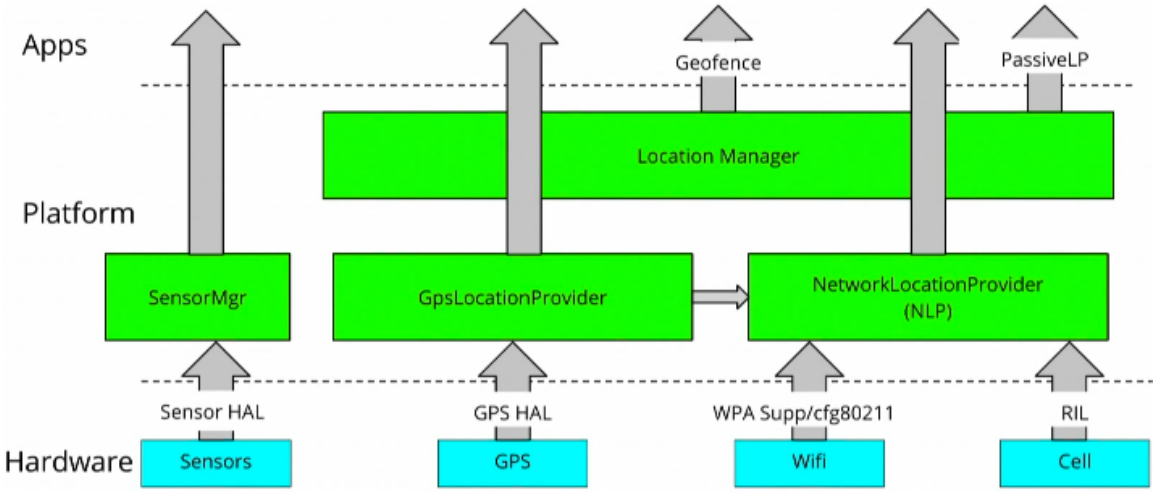
4.1 Prostředky pro lokalizaci zařízení

Nejpřesnější a nejvíce energeticky náročný je systém GPS (Global Positioning System). Signál GPS je dostupný při přímém výhledu na oblohu a uvnitř budovy většinou dostupný není. GPS Modul musí přijmout data nejméně ze 4 satelitů a pomocí těchto dat a aktuálního času vypočítat svou polohu [12]. To trvá nějaký čas a vybíjí baterii zařízení. Aby bylo získání polohy z GPS rychlejší, lze využít i A-GPS (Assisted GPS), které pomocí stažení asistenčních dat ze vzdálených serverů razantně zrychlí rychlost prvního získání polohy z GPS modulu.

Dalším zdrojem polohy může být WiFi. Pomocí MAC adresy dostupných AP (přístupových bodů) a případně síly signálu lze zjistit polohu zařízení. Je však důležité mít polohu AP v databázi a aktualizovat informace o jejich poloze. Toho společnosti dosahují například mapováním AP při mapování ulic (například Google Street View) nebo sbíráním anonymních dat ze zařízení. Ve vnitřních prostorách a ve městech tak může být poloha, díky velkému počtu přístupových bodů, určená pomocí WiFi velmi přesná. Zároveň je její získání málo energeticky náročné. Na druhou stranu v neobydlených oblastech je tato metoda nepoužitelná.

Podobně jako lokalizace pomocí WiFi funguje i lokalizace pomocí mobilní sítě. Zařízení je připojené k jedné vysílací věži BTS[1] a může mít signál i z několika dalších. Z informací

[1] BTS – Base transceiver station



Obrázek 4.1: Schéma rozhraní `LoacationManager` pro přístup k základním zdrojům polohy v systému Android. Umožňuje přístup k GPS, WiFi a k poloze z mobilních sítí. Nepřistupuje ale k senzorům zařízení (akcelerometr, gyroskop).

o poloze vysílacích věží a síly signálů lze vypočítat podobným způsobem jako u WiFi polohu zařízení. Takto určená poloha není příliš přesná, ale na druhou stranu funguje většinou jak ve vnitřních prostorách, tak v neobydlených oblastech a je nejméně energeticky náročná ze všech zmíněných metod. Porovnání všech zdrojů polohy a jejich přibližná energetická náročnost je v tabulce 4.1.

Technology	**Accuracy**	**Power**	**TTFF**	**Restricted Availability**
A-GPS	< 12m	1 425 - 5 700mWs	3s-114s	indoors urban canyons
WiFi	50 - 500m	724.7mWs	<3s	rural areas without WiFi
Cell-Id	0.5 - 35km	< 20mWs	<1s	regions without cellular

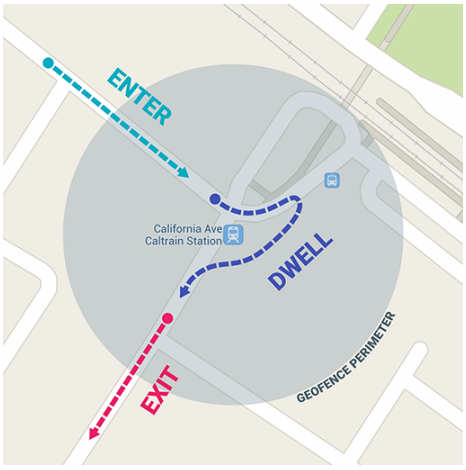
Tabulka 4.1: Porovnání vlastností zdrojů polohy [12].

Systém Android nabízí rozhraní `LocationManager` [6], které zastřešuje všechny zdroje polohy. Schéma tohoto API je pro lepší představu na obrázku 4.1[2].

4.1.1 Lokalizace zařízení pomocí Fused Location Provider API

Na obrázku 4.1 si lze všimnout, že rozhraní `LocationManager` nevyužívá informace ze senzorů. Tyto informace mohou být velmi cenné, protože lze pomocí nich detekovat aktivitu uživatele (například pomocí `ActivityRecognitionApi`). Pokud aplikace potřebuje polohu z GPS modulu, může být tento modul vypnut pro úsporu energie, pokud je zařízení stacionární. Naopak může být tento modul zapnut, pokud uživatel například běží nebo řídí auto. Rozhraní `LocationManager` také nebere v potaz historii polohy. Byla-li poloha určena v blízké historii, lze přibližně odhadnout, že zařízení bude stále na stejném místě nebo blízko tohoto místa. Tyto faktory a další bere v potaz `FusedLocationProviderApi`. Toto API je k dispozici prostřednictvím komponenty Google Play Services a je doporučováno pro přístup k poloze v systému Android [9].

[2] Zdroj schématu – https://www.youtube.com/watch?v=Bte_GHuxUGc



Obrázek 4.2: Geofence a detekce jednotlivých událostí – ENTER, DWELL a EXIT [2].

Aplikace může pomocí `FusedLocationProviderApi` registrovat žádost o periodické aktualizace polohy ve dvou režimech – aktivním nebo pasivním. Při pasivním režimu dostává aplikace aktualizace polohy při požadavcích od ostatních aplikací v systému. Při aktivním režimu jsou aplikaci doručovány aktualizace v explicitně daném intervalu. Ani v jednom z těchto režimů nemusí být aplikace spuštěna. Google Play Services při změně polohy aplikaci spustí a předá jí data o poloze.

Pasivní režim nemá vliv na spotřebu energie, kdežto aktivní má podle nastavení dalších proměnných větší či menší vliv na spotřebu energie. Pokud programátor nastavil u aktivního režimu, že žádá o vysokou přesnost polohy (API se pokusí zapnout GPS modul), mají pak další proměnné minimální vliv a je tedy pak toto API velmi energeticky náročné. Pokud není potřeba vysoká přesnost polohy, lze zažádat o vyváženou přesnost polohy a spotřebu energie. Při tomto režimu se nezapíná GPS modul a určuje se poloha z mobilních sítí a WiFi. Pokud je ale k dispozici poloha z GPS modulu, použije se.

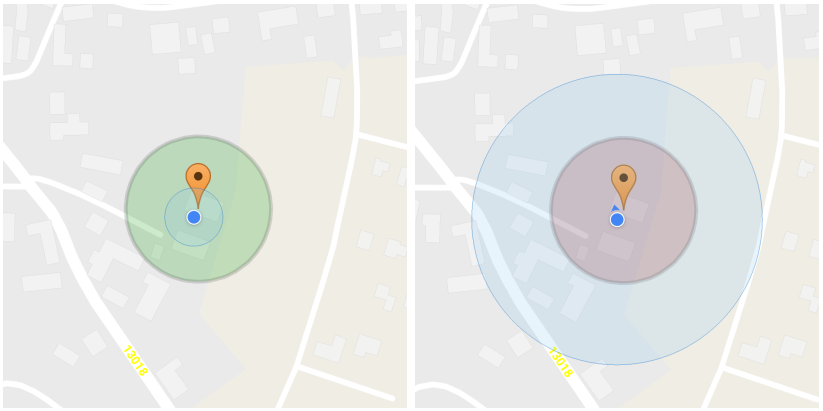
4.1.2 Lokalizace zařízení pomocí Geofencing API

Prostřednictvím Google Play Services je k dispozici také `GeofencingApi` [2]. Má stejné vlastnosti jako `FusedLocationProviderApi`, ale místo registrace žádosti o periodické aktualizace, lze pomocí tohoto API zaregistrovat v systému Android tzv. *geofence*. To jsou kruhové oblasti, které pak samotný systém hlídá a reaguje na detekované události. Každá geofence může detekovat tři události:

- ENTER – vstoupení do geofence
- DWELL – setrvání zařízení uvnitř geofence po danou dobu
- EXIT – opuštění geofence

Geofence s naznačením pohybu, který vyvolá danou událost, je zobrazena na obrázku 4.2. Pokud pak zařízení svou polohou vyvolá jakoukoliv z těchto událostí, je spuštěna žádající aplikace a jsou jí předána další data. Žádající aplikace tak nemusí být neustále spuštěna.

`GeofencingApi` je pro tyto účely optimalizované a využívá aktuálně nejpřesnější polohu. Nezapíná GPS modul a má nízkou spotřebu energie. Toto API je navíc z velké části



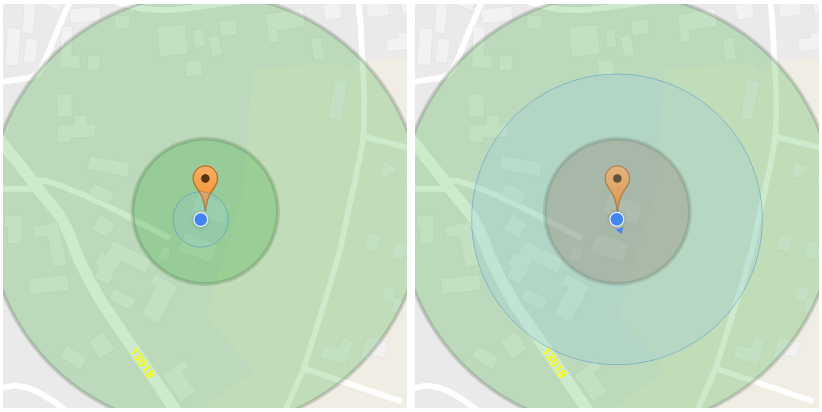
Obrázek 4.3: Na obrázku vlevo je geofence s poloměrem 50 metrů (větší kruh) a zařízení s polohou určenou uvnitř této geofence (menší modrý kruh). Poloha zařízení je přesná a geofence je tak ve stavu DWELL (geofence má zelenou barvu). Na obrázku vpravo je stejná geofence ve stavu EXIT (geofence má červenou barvu). Občasná ztráta přesnosti polohy zařízení způsobí opuštění místa.

pasivní, takže se snaží využívat žádosti o polohu od jiných aplikací. Pokud jakákoliv aplikace (například widget s počasím) zažádá o polohu, zkontroluje se s touto polohou také stav jednotlivých geofence. Je doporučeno, aby jednotlivé geofence měly minimální poloměr 100–150 metrů, aby nedocházelo k chybným událostem [2]. Maximální možný počet zaregistrovaných geofence je 100 [2].

4.2 Výsledná metoda

Může se zdát, že `GeofencingApi` je ideální pro tento typ aplikace – bylo pro takovéto případy užití vytvořeno a optimalizováno. Doporučovaná velikost geofence (100–150 metrů) je ale pro účely této aplikace příliš velká. Při testování se ukázalo, že opravdu systém detekuje stavy geofence s poloměrem menším než 100 metrů špatně. Aplikace velmi často hlásila, že zařízení opustilo místo a po několika minutách zase vstoupilo na místo, i když zařízení bylo stále na stejném místě. Většinu času není na zařízení GPS modul zapnut, proto se poloha určuje pouze pomocí WiFi sítě, která má přesnost nejčastěji okolo 50 metrů. Tato hodnota není konstantní a většinou kolísá mezi hodnotami 50 metrů až 100 metrů. To může způsobit, že geofence s poloměrem 50 metrů systém Android detekuje jako opuštěné, viz obrázek 4.3.

Oficiální dokumentace zmiňuje odstranění tohoto problému, buď zvětšením poloměru geofence, nebo detekováním zařízení uvnitř geofence až po setrvání uvnitř po delší dobu [2]. Zvětšení poloměru není žádoucí, proto byla nastavena doba setrvání na jednu minutu. Pokud je zařízení konstantně po tuto dobu na místě, detekuje aplikace zařízení uvnitř místa. Rozdíl v chybovosti ale nebyl skoro žádný a další zvýšení této doby by degradovalo rychlost rozpoznávání aplikace přítomnosti na místě. Odstranění tohoto problému bylo vyřešeno přidáním další geofence. Každé místo má svou původní geofence (minimálně 50 metrů velkou) a druhou vnější geofence s poloměrem o 75 metrů větší (číslo získáno experimentálně). Aby aplikace prohlásila, že zařízení opustilo místo, musí zařízení opustit obě geofence a aby aplikace prohlásila, že zařízení vstoupilo na místo, musí zařízení vstoupit do obou geofence.

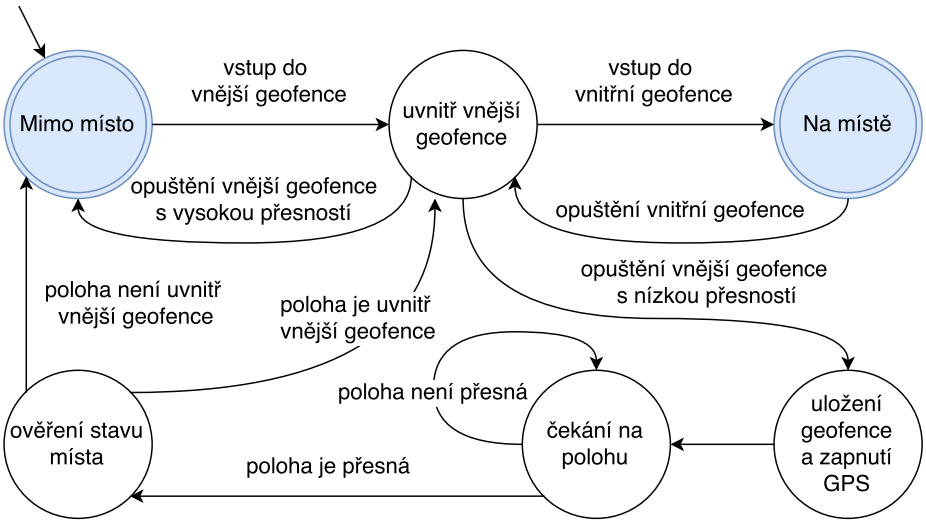


Obrázek 4.4: Použití dvou geofence u místa s poloměrem 50 metrů. Na obrázku vlevo má zařízení přesnou polohu a obě geofence jsou tak ve stavu DWELL (mají zelenou barvu). Na obrázku vpravo je vidět, že malá ztráta přesnosti způsobí opuštění jen vnitřní geofence. Vnější geofence je tak stále ve stavu DWELL.

Pokud zařízení ztratí na přesnosti polohy, opustí se s velkou pravděpodobností pouze vnitřní geofence a zařízení se jeví jako stále na místě. Situace je znázorněna na obrázku 4.4.

Tento přístup značně zvýšil spolehlivost. Na druhou stranu zvýšil vzdálenost od místa, které zařízení musí urazit, aby aplikace detekovala místo jako opuštěné. Nebyl ale zvýšen poloměr místa, což je výhodnější oproti použití jedné velké geofence. Bohužel i tak v menší míře občas aplikace detekovala chybně místo jako opuštěné. Testováním bylo zjištěno, že tento problém způsobuje ztráta signálu WiFi. Stačí nepatrná ztráta signálu nebo přechod na jiný AP (přístupový bod) a zařízení se na okamžik přepne na mobilní sítě a získá polohu z tohoto zdroje. Mobilní sítě mají přesnost polohy i několik kilometrů, a proto tady použití dvou geofence u jednoho místa nepomůže. Při detekování události nabízí `GeofencingApi` možnost získat informaci, jaká byla přesnost polohy při vyvolání události. Pokud tedy při opuštění místa byla přesnost polohy nízká (menší než 75 metrů), uloží aplikace danou geofence a zaregistruje aktualizace polohy pomocí `FusedLocationProviderApi` v aktivním režimu. To způsobí vyšší spotřebu, ale při získání přesnější polohy může aplikace rozhodnout, jestli daná událost byla chybná či ne. Algoritmus je naznačen konečným automatem na obrázku 4.5.

Testování metody v průběhu jejího vytváření probíhalo primárně na dvou mobilních zařízení – OnePlus X a Sony Xperia Z3. Byl vytvořen prototyp aplikace s implementovanou metodou, která při detekci události vytvořila notifikaci obsahující akční tlačítka. Uživatel pak pomocí notifikace potvrdil tuto událost jako správnou nebo jako chybnou. Tato informace se následně odeslala na server pro analýzu. Na začátku experimentování byla chybovost u míst s poloměrem 50 metrů okolo 50 % (100 vzorků). S přidáním druhé geofence byla chybovost snížena na 17 % (200 vzorků). Po poslední modifikaci s ověřením stavu místa při nízké přesnosti polohy byla chybovost na těchto zařízení nulová. Odhadovaná spotřeba energie výsledné metody za 1 den uváděná systémem Android 6.0.1 se pohybuje okolo 20 mAh. Po implementování výsledné aplikace a následné nasazení na dalších 15 zařízení (beta testování) jsem se setkal s občasnou chybovostí u zařízení Huawei P8 Lite. U ostatních zařízení, podle slov dotazovaných uživatelů, problém s detekcí přítomnosti na místě není.



Obrázek 4.5: Algoritmus automatické detekce přítomnosti na místě s využitím dvou geofence (vnitřní a vnější) u jednoho místa. Pokud při opuštění vnější geofence byla přesnost polohy zařízení nízká, je zapnut GPS modul pomocí `FusedLocationProviderApi` a po získání dostatečné přesnosti polohy se vyhodnotí stav místa.

Kapitola 5

Implementace serverové aplikace

Serverová část zajišťuje zprostředkování informace o přítomnosti uživatelů na místech ostatním uživatelům služby Inspotis. V kapiotole 3 bylo definováno, které informace o uživatelích bude služba shromažďovat. Tato data musí být perzistentně uložena a dostupná pro klientské aplikace. Zároveň musí být dostupná jen autorizovaným uživatelům.

Pro uložení dat jsem zvolil databázi MySQL. Výsledné schéma databáze odpovídající ER diagramu na obrázku 3.1 lze vidět na obrázku 5.1.

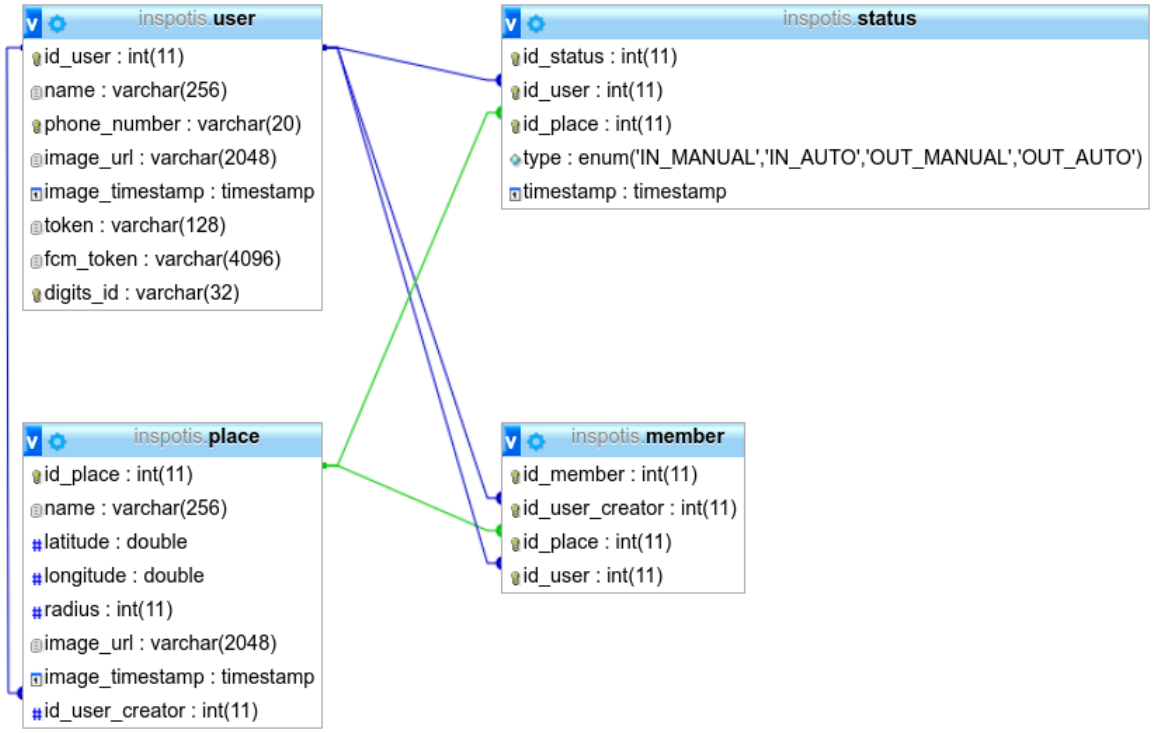
5.1 Rozhraní pro klientské aplikace

Klientské aplikace komunikují se serverovou částí prostřednictvím sítě Internet. Data uložená na serveru musí být proto klientským aplikacím přístupná pomocí síťového protokolu. Zvolil jsem protokol HTTP, který je jednoduchý, snadno se testuje a je pro něj dostupné velké množství implementací. Funkci HTTP serveru mohou vykonávat volně dostupné implementace Apache[1] nebo Nginx[2]. Apache nabízí možnost využít několika programovacích jazyků pro zpracování jednotlivých dotazů od klientů, jako například Python nebo PHP. Aby rozhraní vytvořené pomocí HTTP protokolu bylo samovysvětlující a dobře testovatelné, existuje architektura REST[3]. Ta využívá skladby URL dotazu a příkazů protokolu HTTP k popisu, co daný dotaz dělá. Příklad mapování HTTP příkazů na CRUD[4] operace s objekty je v tabulce 5.1. V REST API se pro definování kontextu vytváří tzv. *endpoint*. To je řetězec přidaný za bázovou URL adresu rozhraní, který definuje objekt, se kterým se pracuje.

HTTP příkaz	**CRUD**
POST	Create
GET	Read
PUT	Update/Replace
PATCH	Update/Modify
DELETE	Delete

Tabulka 5.1: Příklad mapování HTTP příkazů na operace CRUD. Toto mapování je doporučeno pro HTTP rozhraní respektující architekturu REST [10].

[1] Apache – https://httpd.apache.org/
[2] Nginx – https://www.nginx.com/
[3] REST – Representational state transfer
[4] CRUD – Create, Read, Update, Delete



Obrázek 5.1: Schéma relační databáze serverové aplikace služby Inspotis. Databáze vychází z ER diagramu na obrázku 3.1. Protože ER diagram neobsahoval žádné N:N vazby, nebyly při transformaci vytvořeny žádné další pomocné tabulky. Oproti ER diagramu se mírně změnil výčet atributů u některých entit.

Pro implementaci serverové aplikace jsem vybral kombinaci Apache, PHP a REST API. Aby byla přenášená data zabezpečena, využil jsem asymetricky šifrovanou verzi protokolu HTTP, tedy protokol HTTPS. Pro vybraný jazyk PHP existuje několik frameworků, které definování REST API značně zjednodušují. Použil jsem volně dostupný microframework Slim[5] pro jeho jednoduchost. Tento framework nabízí jednoduchou definici REST rozhraní, bez vytváření vlastního parseru dotazu a dalšího programování. Je tedy pak mnohem jednodušší se zaměřit na jádro aplikace. Příklad implementace endpointu pro získání uživatelů u daného místa ve frameworku Slim:

```
/**
 * GET /places/{id_place}/user_members
 */

$app->get('/places/{id_place}/user_members'),
        function (Request $request, Response $response, $args) {

    $idPlace = $request->getAttribute('id_place');

    // authorize and return members from database
    ...
});
```

[5]Slim – https://www.slimframework.com/

Framework Slim může vracet data klientské aplikaci v jakémkoliv formátu. Zvolil jsem serializační formát JSON. Další informace k tomuto frameworku lze nalézt v oficiální dokumentaci [4]. Vytvořené REST API je dostupné na bázové URL adrese rozhraní serverové aplikace[6]. Přehled všech vytvořených endpointů na této URL je v tabulce 5.2.

HTTP příkaz	**endpoint**
POST	`/login`
GET, PUT, DELETE	`/me`
PUT	`/me/name`
POST	`/me/image`
PUT	`/me/fcm_token`
GET	`/places`
POST	`/places/new`
GET, PUT, DELETE	`/places/{id_place}`
POST	`/places/{id_place}/image`
GET, PUT	`/places/{id_place}/status`
GET	`/places/{id_place}/user_members`
POST	`/places/{id_place}/user_members/new`
GET, DELETE	`/places/{id_place}/user_members/{id_user}`
GET	`/places/{id_place}/shared_members`
GET, DELETE	`/places/{id_place}/shared_members/{id_user}`
POST	`/users/by_contacts`
GET	`/users/{id_user}`
GET	`/members`
GET	`/members/{id_user}`
GET	`/members/{id_user}/places`
GET	`/members/{id_user}/places/{id_place}`
GET	`/members/{id_user}/places/{id_place}/status`
POST	`/message`

Tabulka 5.2: Přehled všech vytvořených endpointů pro klientskou aplikaci. Jednotlivé řetězce endpointů patří za bázovou URL adresu REST API serverové aplikace. Pokud řetězec endpointu obsahuje podřetězec ohraničený znaky „{“ „}“, znamená to, že tato část endpointu je variabilní.

5.1.1 Autentizace a autorizace uživatele

Klientská aplikace po prvním spuštění při ověřování telefonního čísla získá od služby Twitter digits ověřovací údaje. Popis a důvod pro výběr této služby je popsán v kapitole 6. Tyto ověřovací údaje klientská aplikace posílá společně se jménem a telefonním číslem uživatele na server přes endpoint `/login`. Ověřovací údaje ze služby Twitter Digits server ověří zasláním na servery této služby. K tomu je využita volně dostupná knihovna TwitterOAuth[7]. Podaří-li se toto ověření, je uživatel autentizován a jeho telefonní číslo je ověřeno pomocí služby Twitter Digits. Pokud tento uživatel ještě není v databázi, je mu vytvořen účet. Jeho číslo

[6] Bázová URL adresa REST API – https://inspotis.vlktomas.cz/api/
[7] TwitterOAuth (MIT licence) – https://github.com/abraham/twitteroauth

je verifikováno a převedeno do formátu E.164[8] pomocí knihovny libphonenumber for PHP[9]. Následně je vygenerován náhodný 64bajtový token, který je uložen do databáze k uživateli a poslán zpět klientské aplikaci. Tento token si klientská aplikace uloží a používá jej pro další komunikaci s REST API. Při dotazu klientské aplikace na jakýkoliv endpoint je tento token vyhledán v databázi a existuje-li, je uživatel autentizován. Komunikace mezi klientskou a serverovou aplikací je prováděna prostřednictvím šifrovaného protokolu HTTPS, tudíž nemůže jiná strana tento token při komunikaci odchytnout.

Autorizace je pak prováděna na úrovni aplikační vrstvy. Autentizovaný uživatel tak například nemá právo získat stav jiného uživatele na daném místě, pokud mezi sebou nemají žádnou vazbu. Podobně nemá uživatel přístup k místům, která mu nenáleží a nemá možnost upravit místo, které nevytvořil.

5.2 Zasílání zpráv klientským aplikacím

Serverová aplikace také zprostředkovává zasílání explicitních notifikací. K tomu je využita služba Firebase Cloud Messaging (FCM). Tato služba a důvod pro její výběr je blíže popsána v kapitole 6. Každému zařízení přiřadí FCM jedinečný identifikátor, který musí server uložit do databáze a použít při posílání zpráv přes servery služby FCM k adresaci zařízení [5]. Ke komunikaci se servery služby FCM je využita volně dostupná knihovna phpFCM[10].

[8] ITU-T E.164 – https://www.itu.int/rec/T-REC-E.164/en

[9] libphonenumber for PHP (Apache 2.0 license) – https://github.com/giggsey/libphonenumber-for-php

[10] phpFCM (MIT licence) – https://github.com/Paragraph1/php-fcm

Kapitola 6

Implementace klientské aplikace

Cílová platforma klientské aplikace je systém Android. V kapitole 3 byly definovány požadavky na funkčnost klientské aplikace a byl navržen princip uživatelského rozhraní aplikace s ohledem na případy užití uživatelem. Prvkem uživatelského rozhraní jsou i notifikace při změně stavu u místa či osoby a explicitní notifikace od jiných uživatelů. V sekci 3.2 pak bylo zmíněno, že uživatel bude autentizován při prvním spuštění aplikace pomocí telefonního čísla. Výsledná aplikace odpovídá těmto požadavkům, ale zároveň bylo třeba implementovat uživatelské rozhraní s ohledem na dodržení konvencí systému Android.

Kromě zobrazení informací o přítomnosti na místech ostatních uživatelů poskytuje aplikace možnost sdílet automaticky informaci o přítomnosti uživatele na jím definovaných místech. K tomu aplikace využívá metody popsané v kapitole 4. Aby byla informace o přítomnosti na místě dostupná ostatním uživatelům, sdílí aplikace tuto informaci prostřednictvím serverové aplikace, s kterou komunikuje pomocí REST API popsaného v předchozí kapitole.

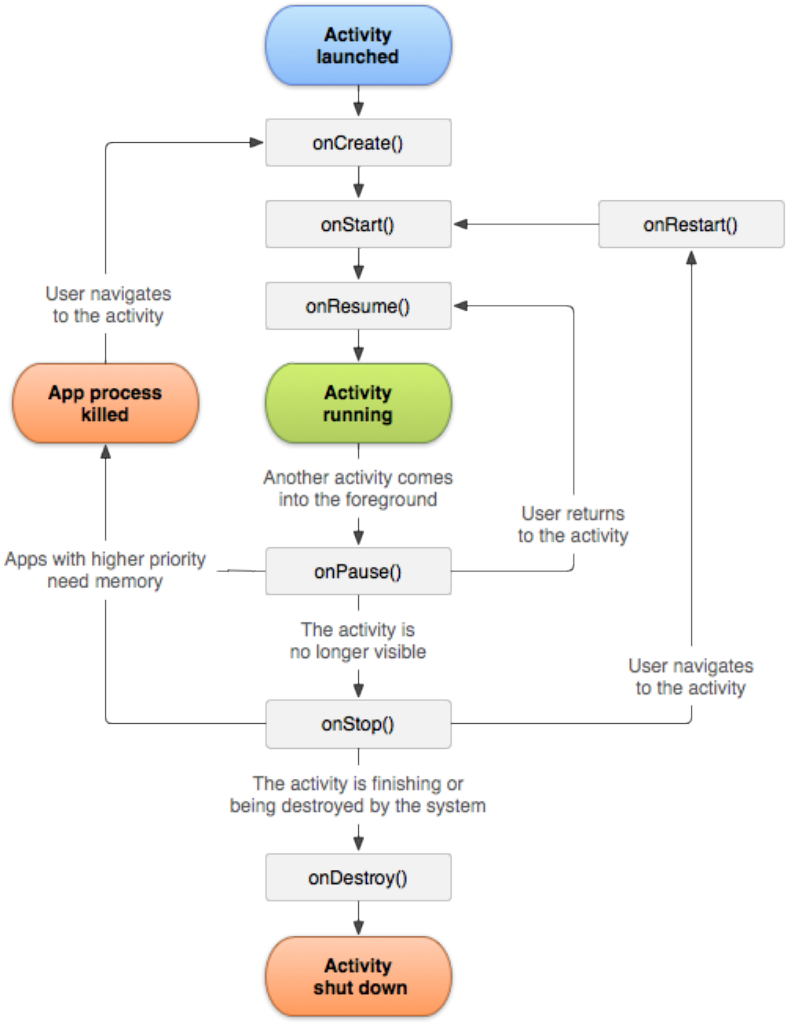
6.1 Programování aplikací na platformě Android

Android je operační systém založený na linuxovém jádře. Mobilní aplikace se podle oficiální dokumentace pro tento systém programují v tzv. *Java API frameworku* někdy též nazývaným *Android framework*. V tomto frameworku se aplikace programují v jazyce Java a následně jsou jednotlivé aplikace spouštěny odděleně v instanci virtuálního stroje Android Runtime (ART) [8].

Existují ale i jiné přístupy k vytváření aplikací na platformě Android, například Xamarin Platform[1], kde se aplikace vytváří v jazyce C#. Další způsob realizace je pomocí `WebView`, kdy se aplikace vytváří pomocí webových technologií jako CSS, HTML a JavaScript. Komponenta `WebView` pak takto vytvořené rozhraní vykreslí jako webovou stránku. Takto funguje například framework Apache Cordova[2]. Výhodou Xamarin Platform a Apache Cordova je, že umožňují vytvářet tímto způsobem multiplatformní aplikace. Výhodou oficiálního přístupu je spousta dostupných knihoven a oficiální podpora.

Zvolil jsem postup podle oficiální dokumentace, tedy použití Android frameworku a vytváření aplikací v jazyce Java. V tomto frameworku jsou fundamentálním prvkem aplikace tzv. *aktivity* definované třídou `Activity`. Tyto aktivity jsou pak většinou jednotlivé obrazovky aplikace. Na rozdíl od toho, jak je v jazyce Java obvyklé definovat vstupní bod

[1] Xamarin Platform – https://www.xamarin.com/platform
[2] Apache Cordova – https://cordova.apache.org/

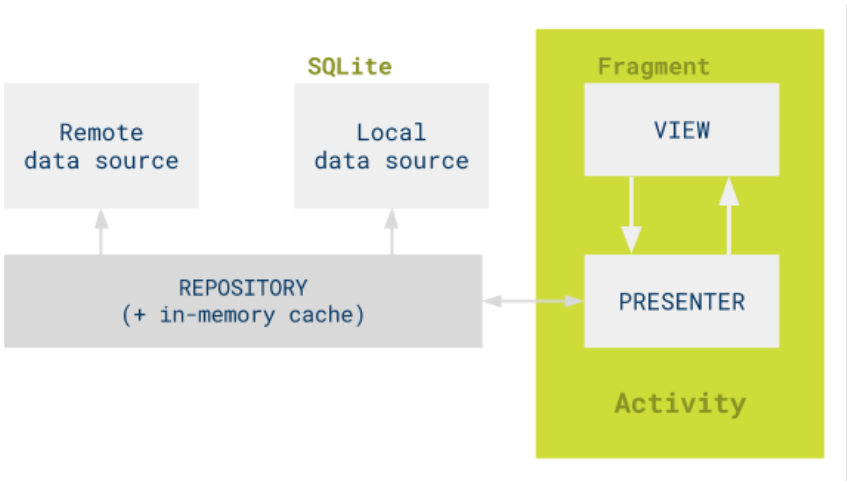


Obrázek 6.1: Životní cyklus aktivity. Chování aktivity je možné ovlivnit implementací daných metod, které jsou volány v jednotlivých fázích životního cyklu aktivity [1].

aplikace pomocí metody `main()`, je v Android frameworku vstupním bodem aplikace jedna z aktivit. Systém Android iniciuje kód v instanci aktivity vyvoláním specifických metod, které odpovídají konkrétním fázím jejího životního cyklu [1]. Životní cyklus aktivity a přehled jednotlivých metod je na obrázku 6.1.

6.2 Architektonický návrhový vzor Model–View–Presenter a jeho použití

Android framework je postavený na třídách `Activity` jako na hlavních komponentách. To není ideální, může se stát, že pak zdrojové soubory těchto tříd mají tisíce řádků a ovládají všechno – od komunikace se serverem až po zobrazování tlačítek. Existuje proto několik způsobů přerozdělení, které toto řeší. Společnost Google zveřejnila příklady několika architekto-



Obrázek 6.2: Schéma architektonického návrhového vzoru MVP pro aplikace vytvářené v Android frameworku. Žlutě zvýrazněná část je původní třída `Activity`, která je v tomto návrhovém vzoru dekomponována na Presenter a View. Komponenta Model je v obrázku pojmenována jako Repository.

nických návrhů[3], které se dají v Androidu použít. V této aplikaci byl využit architektonický návrhový vzor Model-View-Presenter (MVP), kvůli jeho jednoduchosti a přímočarosti. Pro odlišení od programových tříd budou komponenty návrhového vzoru v následujícím textu takto odlišeny. Architektura je zobrazena na obrázku 6.2[4].

Z obrázku 6.2 je patrné, že o data se stará část Repository (Model), která tato data může získat z jakéhokoliv zdroje – na obrázku ze vzdáleného zdroje a z místního zdroje (SQLite databáze). Vytváří tedy vrstvu nad daty a zbývajícím částem aplikace poskytuje jednotné rozhraní. Vlastní logiku aplikace zajišťuje komponenta Presenter. Připravuje data k zobrazení pro komponentu View a přijímá nová data z View. Komponenta View se pak pouze stará o správné zobrazení dat a odezvu na události od uživatele.

Oproti původní koncepci Android frameworku je tato architektura mnohem flexibilnější a přehlednější. Každá třída `Activity` má vlastní komponenty Presenter a View, kdežto Model je samostatný a nezávislý. Třída `Activity` se pouze stará o věci spojené s Android frameworkem a o svůj životní cyklus. Veškerá logika aplikace je přemístěna do Presenteru, kde v nejlepším případě vůbec není potřeba žádné součásti Android frameworku. Část View může vykonávat samotná aktivita a nebo pro ještě větší flexibilitu samostatná třída `Fragment`.

Aby byl zdrojový kód co „nejčistší“ a dobře upravitelný, využil jsem navíc sbírky rad a konvencí používaných při programování na platformě Android[5,6].

[3] Android Architecture Blueprints (Apache 2.0 licence) – https://github.com/googlesamples/android-architecture/

[4] Zdroj schématu – https://github.com/googlesamples/android-architecture/tree/todo-mvp/

[5] Android Guidelines (Apache 2.0 licence) –https://github.com/ribot/android-guidelines

[6] Best practices in Android development (Apache 2.0 licence) – https://github.com/futurice/android-best-practices

6.3 Uživatelské rozhraní aplikace s využitím Material Designu

V kapitole 3 v sekci 3.4 bylo navrženo rozmístění ovládacích prvků a zobrazení kolekcí a struktur klientské aplikace. Uživatelské rozhraní klientské aplikace tak z tohoto návrhu vychází a zároveň dodržuje konvence systému Android.

6.3.1 Material Design

Společně s verzí Androidu 5.0[7] byl představen nový vizuální jazyk – Material Design. Tento jazyk vytváří jednotný vizuální styl napříč všemi zařízeními. Obsahuje mnoho principů a doporučení dobrého návrhu uživatelského rozhraní. Hlavním stavebním prvkem Material Designu je tzv. *materiál*. To je metafora, označující povrch inspirovaný papírem a inkoustem [7]. Uživatelské rozhraní se pak skládá z několika takovýchto materiálů, které jsou přes sebe naskládány v 3D prostoru a vrhají tak na sebe stín. Důležité ale je, že všechny materiály mají stejnou nepatrnou výšku (jako list papíru). Dalším principem je typografie a tvoření vizuálně rozpoznatelné struktury pomocí plochých a kontrastních prvků. V neposlední řadě jsou v Material Designu velmi důležité pohyby neboli animace. Ty by měly pomáhat uživateli pochopit uživatelské rozhraní. Více informací lze nalézt v oficiální dokumentaci [7].

Protože jsou uživatelé platformy Android zvyklí na konvence a vzhled Material Designu, rozhodl jsem se tímto vizuálním jazykem řídit a zjednodušit tak uživateli učení se uživatelskému rozhraní.

6.3.2 Technologie pro tvorbu uživatelského rozhraní na platformě Android

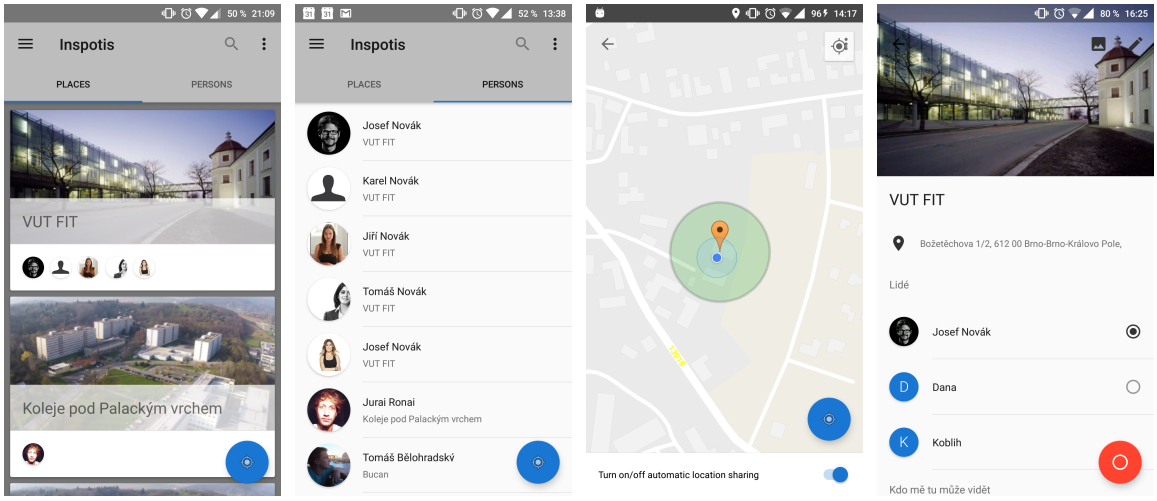
Android framework nabízí vytváření nativních uživatelských rozhraní. Tato rozhraní mohou být definována pomocí souboru XML nebo vytvářena programově v jazyce Java [11]. Nativní uživatelské rozhraní lze vytvořit také pomocí Xamarin Platform, kde se definuje pomocí souborů XAML. Aplikace je pak vyvíjena v jazce C#. Další způsob realizace, který již ale není nativní, je pomocí webových technologií, pokud je aplikace vyvíjena například pomocí Apache Cordova. Oproti nativnímu rozhraní nemusí být vizuálně poznat rozdíl. Mírný rozdíl může být znát v odezvě reakce uživatelského rozhraní a animací. Tento přístup se velmi často používá, pokud je aplikace vyvíjena multiplatformě pomocí webových technologií. Uživatelské rozhraní je pak na všech platformách totožné a má jedny zdrojové soubory.

Zvolil jsem nativní přístup pomocí jazyku Java a souborů XML. Obrovskou výhodou je mnoho již vytvořených oficiálních komponent respektujících Material Design. Zároveň je velká podpora vytváření nativního uživatelského rozhraní ve vývojářském software Android Studio[8].

6.3.3 Prototypy uživatelského rozhraní a testování použitelnosti

Uživatelské rozhraní bylo vylepšováno testováním použitelnosti. Aby bylo uživatelské rozhraní použitelné, mělo by splňovat atributy zmíněné v kapitole 3 v sekci 3.4. Bylo vytvořeno několik prototypů, které byly následně testovány na reálných uživatelích, a bylo sledováno,

[7] Android Lollipop – https://developer.android.com/about/versions/lollipop.html
[8] Android Studio – https://developer.android.com/studio/index.html



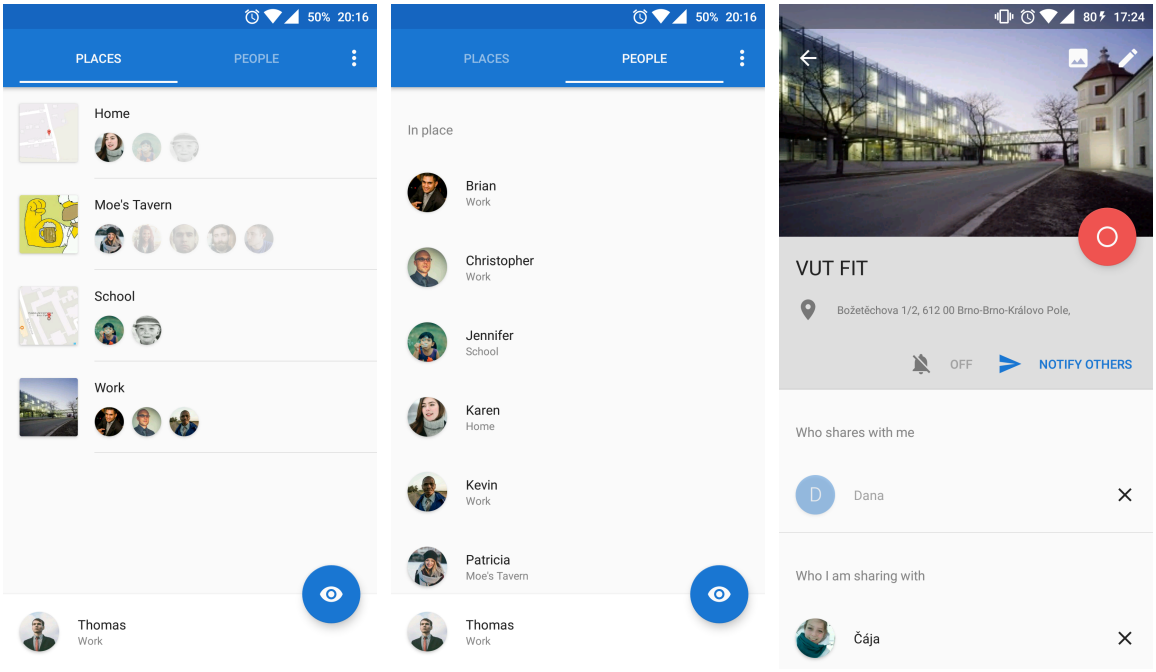
Obrázek 6.3: Prototyp uživatelského rozhraní vycházející z prvotních návrhů na obrázku 3.3 s použitím Material Designu.

jak uživatel rozhraní chápe a které úkony mu dělají problémy. Příklad takových prototypů je na obrázku 6.3. Následně bylo uživatelské rozhraní opakovaně upravováno a testováno na uživatelích. Typický scénář testování byl inspirován Krugem [14] a obsahoval tyto kroky:

- *Představení aplikace* – byla sledována reakce na popis aplikace a často kladené otázky (popis aplikace byl na základě toho postupně upravován)
- *Otázka na využití aplikace* – jestli by uživatel aplikaci použil, a pokud ano tak na co
- *Instalace aplikace a přihlášení* – po instalaci bylo požadováno, aby uživatel nahlas sděloval, co vidí, co dělá a zároveň, aby jeho používání aplikace bylo viditelné
- *Testování případů užití uživatelského rozhraní*

  1. Vytvoření místa
  2. Sdílení tohoto místa jiným uživatelům
  3. Zjistit, kdo je na vytvořeném místě
  4. Zjistit, kde je jiný uživatel
  5. Manuálně dát vědět svou přítomnost na vytvořeném místě
  6. Zapnout automatické sdílení informace o přítomnosti na místě

Sledování uživatelů při používání aplikace přineslo mnoho cenných poznatků. V původním prototypu na obrázku 6.3 bylo nutné pro sdílení místa toto místo upravit kliknutím na ikonku v pravém rohu (zobrazení místa na obrázku 6.3 vpravo). Při testování se ale ukázalo, že buď trvalo uživateli dlouho, než ho to napadlo nebo to uživatel vůbec nenašel. V další verzi bylo proto sdílení místa přesunuto přímo na obrazovku se zobrazeným místem. Dále bylo zjištěno, že uživatel u některých úkonů chce vyvolat kontextovou nabídku u místa v seznamu míst pomocí dlouhého stisku. Aplikace tak v seznamu míst umožňuje takto vyvolat kontextovou nabídku. V případě užití „Zapnout automatické sdílení informace o přítomnosti na místě“ uživatelé často hledali tuto možnost v nastavení aplikace namísto na obrazovce s aktuálním stavem a polohou uživatele, jež lze vidět na obrázku 6.3 na třetí obrazovce



Obrázek 6.4: Aktuální podoba uživatelského rozhraní vycházející z prototypů, které byly testované na uživatelích a následně iterativně vylepšované. Třetí obrazovka s mapou a tlačítkem zapnout/vypnout sdílení polohy byla odstraněna a byla nahrazena tlačítkem „oko“, jež lze vidět na první nebo druhé obrazovce.

zleva. Tuto obrazovku jsem proto odstranil a nahradil ji spodní lištou s aktuální textovou informací o stavu uživatele a jedním akčním tlačítkem na zapnutí/vypnutí automatického sdílení informace o přítomnosti na místě. Takto bylo provedeno mnoho dalších úprav.

Příklad toho, jak aplikace aktuálně vypadá, lze vidět na obrázku 6.4. V této aktuální verzi uživatelského rozhraní jsou všechny nejčastější případy užití pro uživatele dostupné ihned po otevření aplikace. Výjimkou je případ užití „Manuálně dát vědět určitým osobám svoji přítomnost na daném místě“, který je dostupný po výběru místa.

6.4 Komunikace se serverovou aplikací

V kapitole 5 je popsáno rozhraní serverového API. K tomu, aby se aplikace mohla k tomuto API připojit a získávat data, bylo třeba uskutečňovat HTTP dotazy. Aplikace musí tyto dotazy provádět asynchronně, aby neblokovaly uživatelské rozhraní aplikace.

Pro tyto účely byla použita knihovna Retrofit[9], díky níž je jednoduché definovat REST API pomocí Java anotací, následně volat asynchronní dotazy a případně reagovat na nastalou chybu. Navíc umožňuje serializaci Java objektů do formátu JSON před odesláním dotazu nebo deserializaci Java objektů z formátu JSON po přijetí odpovědi ze serveru. Ke konverzi Java objektů do serializačního formátu JSON a naopak je využita knihovna GSON[10]. Příklad deklarace endpointu serverového API pomocí knihovny Retrofit:

[9]Retrofit (Apache 2.0 licence) – https://github.com/square/retrofit
[10]GSON (Apache 2.0 licence) – https://github.com/google/gson

```
public interface InspotisApi {

    @GET("places/{id_place}/user_members")
    Call<List<Member>> getMembers(@Path("id_place") String Id);

    ...

}
```

Pokud pak aplikace potřebuje provést asynchronně HTTP dotaz, je volána pomocí knihovny Retrofit jen daná metoda deklarovaného endpointu:

```
Retrofit retrofit = new Retrofit.Builder()
                .baseUrl("https://inspotis.vlktomas.cz/api/")
                .addConverterFactory(GsonConverterFactory.create())
                .build();

mApi = retrofit.create(InspotisApi.class);

// asynchronous get members of the place with id 42
Call<List<Member>> call = mApi.getMembers("42");
call.enqueue(new Callback<List<Member>>() {

    ...

});
```

V předchozí ukázce kódu si lze všimnout vytvoření instance třídy Retrofit, při níž bylo knihovně nastaveno, jakou má použít bázovou adresu serverového rozhraní a že data přenášená mezi knihovnou a serverem budou serializována/deserializována pomocí knihovny GSON (`GsonConverterFactory`). Následně bylo knihovně nastaveno, které endpointy jsou na REST rozhraní definovány pomocí Java rozhraní `InspotisApi` obsahující deklaraci endpointů. Nakonec je asynchronně uskutečněn dotaz na daný endpoint a lze definovat `Callback`, pokud jsou data přijata nebo se něco pokazí. Takto lze získat seznam uživatelů jako seznam instancí třídy `Member` přímo ze serveru bez toho, aby se manuálně parsoval řetězec JSON vrácený ze serveru nebo byly vytvářeny nějaké objekty tříd. Podobným způsobem je jednoduše a elegantně definováno celé rozhraní pro všechny endpointy serverové aplikace. Více informací lze nalézt na webových stránkách projektu[11].

6.4.1 Autentizace uživatele

V sekci 5.1 je popsáno, že pro autentizaci vůči serveru používá aplikace endpoint `/login`. Následně přijme klientská aplikace od serverové aplikace token, který si aplikace uloží a pak ho používá pro autentizaci při dotazech na ostatní endpointy. V sekci 3.2 pak bylo zmíněno, že uživatel bude autentizován při prvním spuštění aplikace pomocí telefonního čísla.

Problémem oproti ověření pomocí emailu nebo účtů třetích stran je, že pro ověření telefonního čísla je potřeba na dané číslo zavolat nebo poslat SMS s ověřovacím kódem, což není oproti emailovým službám zdarma. Ceny různých SMS bran a služeb nabízející ověřovací SMS nejsou vysoké, ale ani zanedbatelné. Využil jsem tedy službu Twitter Digits, která nabízí tyto služby zdarma a neomezeně [3].

[11] Retrofit – http://square.github.io/retrofit/

Služba Twitter Digits nabízí SDK pro ověření čísla pro aplikace na systému Android. Po integraci do aplikace a vyvolání ověření čísla je zobrazen formulář pro zadání telefonního čísla. Po vyplnění a potvrzení je na toto číslo zaslána ze služby Twitter Digits ověřovací SMS. Ta obsahuje šestimístný kód, který musí uživatel zadat do aplikace. Po potvrzení je ověřovací kód zaslán na servery služby Twitter Digits, kde je zkontrolován. Je-li správný, pak je uživatel vlastníkem daného čísla. V oficiální dokumentaci je zmíněno, že celý tento proces se dá z velké části zautomatizovat [3]. Twitter Digits SDK může zjistit pomocí Android frameworku z některých mobilních zařízení telefonní číslo samostatně bez uživatele. Následně může ověřovací SMS ihned po přijetí aplikace automaticky přečíst, zpracovat a nakonec také odstranit. Uživatel se systémem Android 6.0 a vyšším tak jen povolí daná práva aplikaci a pak může být celý proces otázkou několika sekund bez zásahu uživatele. Bohužel při testování této funkce ve verzi Twitter Digits SDK 1.3.12 se automatické ověření čísla zprovoznit nepodařilo.

Pokud byl proces ověření službou Twitter Digits úspěšný, jsou aplikaci zaslány ověřovací údaje. Tyto údaje potvrzují vlastnictví čísla. Aplikace pak zasílá tyto údaje společně se svým jménem a číslem při autentizaci vůči serverové aplikaci na endpoint `/login`. Serverová aplikace pak tyto údaje ověří a jsou-li správné, vrátí klientské aplikaci autentizační token.

6.4.2 Přijímání zpráv a výsledná architektura

Důležitou součástí aplikace je funkce informování uživatele o nastalé změně. Například uživatel může být notifikován, pokud někdo dorazil na místo. Může být také notifikován explicitní notifikací od jiného uživatele. Existují dva přístupy, jak může aplikace na tyto události reagovat:

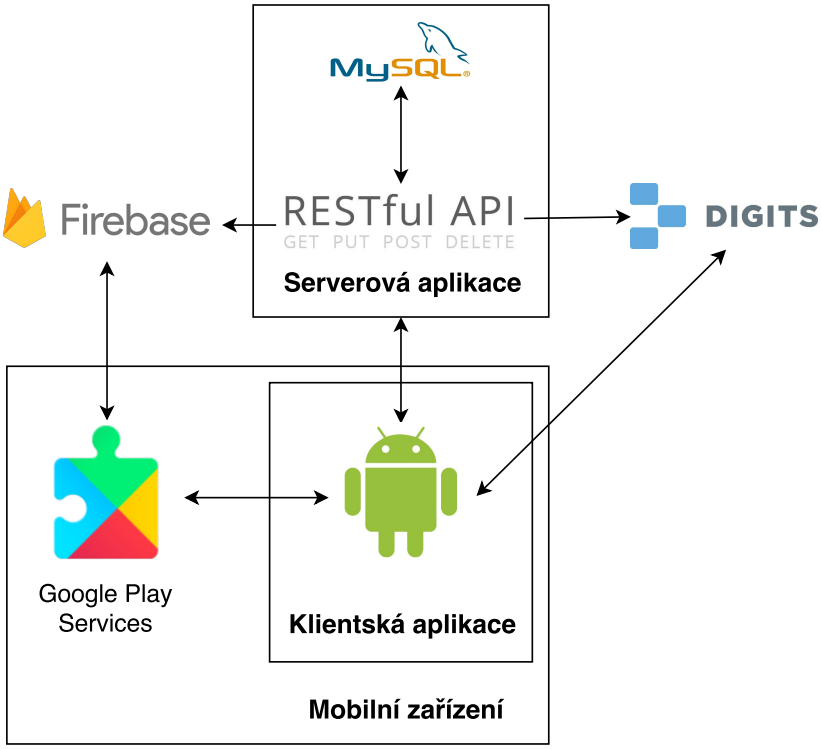
- tzv. *polling* – periodické dotazování se serveru, jestli nastala změna
- otevřený socket, přes který může server odeslat data v jakoukoliv chvíli

První zmíněný přístup vyžaduje, aby aplikace měla spuštěnou službu a například každých 5 minut kontaktovala server. Neustále spuštěná služba i opakované ustavování komunikačního kanálu je velmi energeticky náročné. Druhý přístup ustavuje komunikační kanál pouze jednou, ale i tak musí být spuštěná nějaká služba, která udržuje kanál otevřený například pomocí *keep-alive*. Aplikace klade důraz na nízkou spotřebu energie. Využil jsem proto druhý přístup a to pomocí služby Firebase Cloud Messaging (FCM), která umožňuje přijímání tzv. *push notifikací*.

Služba FCM vylepšuje druhý přístup tak, že je zabudována v komponentě Google Play Services a poskytuje všem aplikacím na zařízení jeden otevřený kanál k serverům FCM. Je tak méně energeticky náročná, protože udržuje jen jednu spuštěnou službu a jeden otevřený kanál pro několik aplikací. Tato služba je zdarma a bez omezení [5].

Klientské aplikace musí být v rámci služby FCM jednoznačně adresovatelné. Služba FCM identifikuje každé zařízení vlastním tokenem. Tento token je dostupný prostřednictvím FCM SDK integrované v aplikaci. Aplikace pak musí tento token odeslat serverové aplikaci. Serverová aplikace tak zná token služby FCM každé klientské aplikace. Přes tento token může server kontaktovat ostatní klientské aplikace. Klientské aplikace nemůžou přímo přijmout zprávu z této služby. O přijetí se stará komponenta Google Play Services. Pokud Google Play Services přijmou zprávu, spustí službu dané aplikace, která na tuto zprávu může zareagovat.

Celá architektura komunikace služby Inspotis je znázorněna na obrázku 6.5, kde jsou naznačeny i jednosměrné a obousměrné komunikace mezi součástmi a ostatními službami.



Obrázek 6.5: Architektura komunikace klientské a serverové aplikace a ostatními službami.

Kapitola 7

Zveřejnění aplikace na Google Play a zhodnocení

Aplikace byla v průběhu vývoje testována na reálných uživatelích. K jednodušší distribuci jsem využil obchod *Google Play*[1]. U této služby musí být vytvořen účet a musí být zaplacen registrační poplatek 25 $. Následně lze prostřednictvím *Google Play Console*[2] vytvořit záznam aplikace a zveřejnit jej v obchodě Google Play. Klientskou aplikaci jsem zveřejnil pod názvem Inspotis. Záznam v obchodě obsahuje také logo, popis, úvodní obrázek a snímky obrazovky aplikace. Vytvořil jsem dvě jazykové verze aplikace – anglickou (výchozí) a českou.

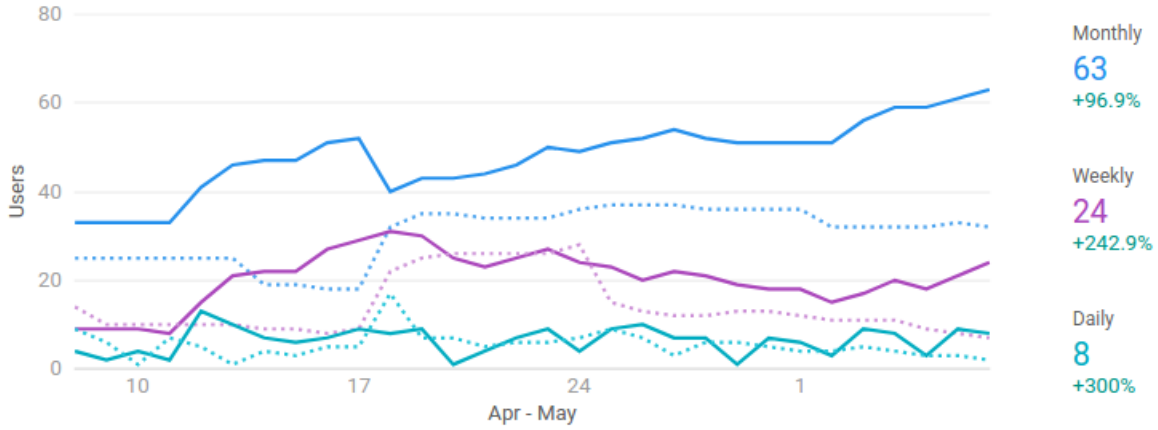
7.1 Alfa a beta verze a testování

V kapitole 4 v sekci 4.2 byl popisován postup vývoje metody automatické detekce přítomnosti na místě. Bylo třeba tuto metodu postupně vylepšovat, testovat ji a odstraňovat chyby. K tomu jsem využil zveřejnění aplikace v alfa verzi v obchodě Google Play. Takto zveřejněná aplikace nelze v obchodě vyhledat a lze omezit přístup k její instalaci jen vybraným uživatelům podle emailové adresy. To mi umožnilo rychle vydávat aktualizace a distribuovat je ihned na mobilní zařízení testerů.

Po vyladění metody automatické detekce přítomnosti na místě bylo třeba testovat použitelnost uživatelského rozhraní a opravovat programové chyby aplikace. Byl proto vytvořen minimální funkční produkt (MVP[3]) a aplikace byla povýšena do beta verze v obchodě Google Play. Od alfa verze se beta verze nijak neliší. U beta verze jsem ale již nedefinoval emaily testerů a zvolil jsem otevřené testování přes odkaz. Přístup k instalaci aplikace tak nebyl omezen emailovými adresami, ale kdokoliv, kdo měl odkaz, mohl testovat. Aplikaci jsem takto rozšířil na dalších 15 zařízení. S testery jsem byl v kontaktu a prováděl jsem testování použitelnosti a ptal se jich na připomínky k aplikaci. Díky tomu jsem do aplikace přidával nové funkce a vylepšoval uživatelské rozhraní.

K odhalení programových chyb aplikace jsem v klientské aplikaci integroval službu *Firebase Crash Reporting*, která při pádu aplikace ihned odešle chybové hlášení bez zásahu uživatele. Chybová hlášení pak Firebase Crash Reporting inteligentně sdružuje a zjednodušuje odhalení chyb aplikace. Služba Google Play Console také nabízí zobrazení chybových

[1] Google Play – https://play.google.com/store
[2] Google Play Console – https://play.google.com/apps/publish/
[3] MVP – Minimum viable product



Obrázek 7.1: Počet uživatelů používající aplikaci během jednoho měsíce opakovaně. Osm uživatelů používá aplikaci denně a 24 uživatelů týdně.

hlášení, ale aby se chybová hlášení v této službě zobrazily, musí uživatel explicitně odeslat chybové hlášení prostřednictvím dialogu při pádu aplikace. Po odstranění chyb jsem aplikaci povýšil do produkční verze. Takto zveřejněná aplikace je již volně dostupná a lze ji vyhledat v obchodě Google Play pod názvem Inspotis.

7.2 Zhodnocení používání aplikace pomocí analytických nástrojů

Služba Google Play Console nabízí různé statistiky o aplikaci. Společně se záznamy v databázi serverové aplikace lze určit počet uživatelů a zhodnotit její úspěšnost. Ve službě Inspotis je k 8. květnu zaregistrováno 39 uživatelů. Z těchto uživatelů je ale nejméně 15 lidí, kteří mi pomáhali při testování aplikace. V obchodě Google Play si aplikaci zobrazilo 168 lidí, z nichž 53 aplikaci nainstalovalo. Z celkového počtu instalací tedy 14 lidí neprošlo přes registraci. Od uživatelů má aplikace v obchodě Google Play 12 hodnocení a pět recenzí s výsledným průměrem 5,0 hvězdiček.

Dále jsem v klientské aplikaci integroval službu *Firebase Analytics*. Ta umožňuje zaznamenávání událostí v aplikaci. Sledoval jsem například, jak uživatelé využívají systém pozvánek integrovaný v aplikaci pomocí *Firebase Dynamic Links*. Zjistil jsem, že bylo vygenerováno 10 pozvánek šesti různými uživateli a aplikaci následně nainstaloval přes pozvánku jeden uživatel. Dále umožňuje Firebase Analytics zobrazit aktivitu uživatelů. Jak lze vidět na obrázku 7.1, někteří uživatelé aplikaci po instalaci opakovaně používají. Počet aktivních uživatelů za celý měsíc (65 uživatelů) se liší od údaje z Google Play Console (53 uživatelů), protože Firebase Analytics počítá každé nové sezení jako jednoho uživatele. Pokud uživatel tedy odinstaluje a znovu nainstaluje aplikaci, je započítán dvakrát.

Po dobu vývoje byla aplikace nainstalována celkem na 45 různých zařízení a na osm různých verzí systému Android (od verze 4.1 až po 7.1). Po měsíčním provozu aplikace v produkční verzi jsem zaznamenal ve Firebase Crash Reporting tři pády aplikace. Chyby způsobující pád byly následně opraveny.

Kapitola 8

Závěr

Tato práce popisuje aplikaci pro systém Android, která nabízí uživatelům možnost sdílet jejich přítomnost na určitých místech a přitom držet velkou kontrolu nad svým soukromím. Nejprve byla porovnána obdobná řešení tohoto problému a z těchto poznatků byly definovány požadavky na funkčnost služby Inspotis. Následně byla implementována serverová aplikace a klientská aplikace. Klientská aplikace byla průběžně testována na reálných uživatelích a bylo iterativně vylepšováno její uživatelské rozhraní a přidávány dodatečné funkce podle odezvy od uživatelů. Nakonec byla aplikace zveřejněna k dispozici zdarma v obchodě Google Play pod názvem Inspotis a byly zhodnoceny její dosavadní výsledky po měsíčním provozu.

Výsledná aplikace dokáže spolehlivě detekovat zařízení na místech s poloměrem 50 metrů a větších. Odhadovaná spotřeba energie aplikace za 1 den uváděná systémem Android 6.0.1 se pohybuje okolo 20 mAh. Tato hodnota byla naměřena na více mobilních zařízeních. V aplikaci je zaregistrováno 39 lidí a nebyly zaznamenány negativní recenze nebo hodnocení aplikace. Rozhraní `GeofencingApi` systému Android s mírnými úpravami funguje spolehlivě a není energeticky náročné. Společně se službou Firebase lze vytvořit systém sdílení polohy mezi uživateli bez potřeby aktivní spuštěné součásti aplikace. Vývojáři mohou využít metodu detekce přítomnosti na místě použitou v této aplikaci.

Do budoucna bych chtěl přidat v klientské aplikaci lokální databázi pro ukládání načtených dat. To zrychlí a zefektivní chod aplikace (nyní se jakákoliv informace vždy stahuje ze serveru). Dále je v plánu rozšířit tuto aplikaci na platformu iOS a vytvořit webovou prezentaci. Bylo by také vhodné vytvořit malé reklamní kampaně na cílové skupiny uživatelů a propagovat aplikaci.

Literatura

[1] Activity. *Android Developers*, cit. 11.4.2017.
URL https://developer.android.com/guide/components/activities/intro-activities.html

[2] Creating and Monitoring Geofences. *Android Developers*, cit. 29.3.2017.
URL https://developer.android.com/training/location/geofencing.html

[3] Digits. *Fabric for Android documentation*, cit. 11.4.2017.
URL https://docs.fabric.io/android/digits/overview.html

[4] Documentation. *Slim Framework*, cit. 11.4.2017.
URL https://www.slimframework.com/docs/

[5] Documentation. *Firebase*, cit. 11.4.2017.
URL https://firebase.google.com/docs/

[6] Location Strategies. *Android Developers*, cit. 29.3.2017.
URL https://developer.android.com/guide/topics/location/strategies.html

[7] Material design guidelines. *Material design*, cit. 11.4.2017.
URL https://material.io/guidelines/

[8] Platform Architecture. *Android Developers*, cit. 11.4.2017.
URL https://developer.android.com/guide/platform/index.html

[9] Receiving Location Updates. *Android Developers*, cit. 29.3.2017.
URL https://developer.android.com/training/location/receive-location-updates.html

[10] REST API Tutorial. *Using HTTP Methods for RESTful Services*, cit. 11.4.2017.
URL http://www.restapitutorial.com/lessons/httpmethods.html

[11] User Interface. *Android Developers*, cit. 11.4.2017.
URL https://developer.android.com/guide/topics/ui/index.html

[12] 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*, July 2012, ISSN 0730-3157, s. 613–622.

[13] Das, A.; Bonneau, J.; Caesar, M.; aj.: The Tangled Web of Password Reuse. In *NDSS*, ročník 14, 2014, s. 23–26.

[14] Krug, S.: *Don't make me think revisited: A common sense approach to web and mobile usability. Berkeley.* CA: New Riders Press, 2014, ISBN 978-0-321-96551-6.

[15] Toch, E.; Cranshaw, J.; Drielsma, P. H.; aj.: Empirical Models of Privacy in Location Sharing. In *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*, UbiComp '10, New York, NY, USA: ACM, 2010, ISBN 978-1-60558-843-8, s. 129–138.

[16] Toch, E.; Cranshaw, J.; Hankes-Drielsma, P.; aj.: Locaccino: A Privacy-centric Location Sharing Application. In *Proceedings of the 12th ACM International Conference Adjunct Papers on Ubiquitous Computing - Adjunct*, UbiComp '10 Adjunct, New York, NY, USA: ACM, 2010, ISBN 978-1-4503-0283-8, s. 381–382.

Příloha A

Obsah CD

Přiložené CD obsahuje zdrojové soubory a další materiály vytvořené v rámci bakalářské práce. Následující struktura uvádí nejdůležitější adresáře a soubory.

- `/`
  - `bachelor-thesis/` – zdrojové soubory této technické zprávy pro LaTeX
  - `excel-at-fit/` – materiály ke konferenci Excel@FIT
    - `2017-ExcelFIT-Inspotis/` – zdrojové soubory článku pro LaTeX
    - `2017-ExcelFIT-Inspotis.pdf` – článek
  - `inspotis-android/` – zdrojové soubory klientské aplikace
    - `design/` – propagační grafika aplikace
    - `README.md` – základní informace a návod na sestavení
  - `inspotis-api/` – zdrojové soubory serverové aplikace
    - `sql/` – schéma relační databáze pro MySQL a vzorová data
    - `README.md` – základní informace a návod na sestavení
  - `inspotis-website/` – zdrojové soubory webové prezentace (ve vývoji)
  - `bachelor-thesis.pdf` – technická zpráva ve formátu PDF
  - `bachelor-thesis_print.pdf` – technická zpráva ve formátu PDF určena pro tisk
  - `inspotis-release.apk` – instalační soubor klientské aplikace ve verzi 1.1.35
  - `poster.pdf` – plakát pro prezentování aplikace
  - `video.mp4` – video pro prezentování aplikace