Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

GPS framework na platformě Android

Plzeň 2014 Viktor Vašina

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 7. května 2014

Viktor Vašina

Abstract

GPS framework on Android platform. This work analyzes possibility of determining location of mobile devices, especially for the Android platform. The work describes the locational methods and compares them. Further attention is paid to implementation of these methods on the Android platform. The final part summarizes procedures frequently used during implementation of applications using the position determining. Framework based on mentioned summary seeks to cover the frequently used actions to simplify the implementation of such applications. To better understand and verify functionality this part is finished by proposal and implementation of a sample application using the created framework.

Abstrakt

Tato práce je analýzou možností určování polohy mobilních zařízení, zejména pro platformu Android. Práce postupně popisuje jednotlivé lokační metody a srovnává je. Dále se zaměřuje na realizaci těchto metod na platformě Android. V další části se jsou shrnuty postupy často používané při implementaci aplikací využívajících určování polohy. Na základě zmíněného souhrnu je zároveň popsán framework, který se snaží časté úkony v sobě obsáhnout pro zjednodušení implementace podobných aplikací. Pro lepší pochopení a ověření funkčnosti je tato část zakončena návrhem a implementací ukázkové aplikace s využitím vytvořeného frameworku.

Obsah

1 Úvod

Cílem této práce je seznámit čtenáře s možnostmi lokace mobilních zařízení a využitím těchto technologií v mobilních aplikacích.

V první části práce popisuje, co je zeměpisná poloha a jak je reprezentována v přístrojích. Následně jsou tyto lokalizační metody popsány a porovnány.

Díky nabytým znalostem z první části je snazší pochopit, jak jsou implementovány tyto technologie na platformě Android. Jelikož v průběhu práce byly některé tyto technologie, resp. jejich implementace, změněny, věnuje se práce okrajově i vývoji lokalizace přístroje pod touto platformou. Zároveň práce postupně prochází frameworkem a umožňuje snáze pochopit, které funkcionality se v aplikacích často používají či jak je lze řešit. Práce uvažuje i potřebu sdílení polohy a jiných informací skrze internetovou síť.

Další částí je ukázková aplikace založená na frameworku, která bude využívat možnosti určování polohy mobilních zařízení a sdílení polohy přes webový server.

Po přečtení práce čtenář získá představu o principech určování polohy mobilních přístrojů a využití těchto informací. Práce popisuje, jak tato data sdílet s dalšími mobilními zařízeními či servery a jak jsou všechny tyto funkce zastřešeny operačním systémem Android.

2 Reprezentace zpageeměpisné polohy

Pro určování zeměpisné polohy je nutné definovat podobu souřadnic jednoznačně určujících polohu bodu na planetě.

2.1 Zeměpisná poloha

Běžně používaným způsobem pro určení polohy na povrchu planety je využití zeměpisné délky a šířky.

Zeměpisná délka je úhel, který svírá rovina nultého poledníku a poledníku, který protíná určovaný bod. Může nabývat hodnot od 0° do 180°. Směrem k východu od nultého poledníku se používá výraz východní délka, směrem na západ od nultého poledníku naopak západní délka.

Zeměpisná šířka je úhel, který svírá rovina rovníku a přímka procházející určovaným bodem a středem planety. Nabývá hodnot od 0° do 90°. Od rovníku směrem na sever se používá výraz severní šířka, směrem na jih jižní šířka.

Tyto dva údaje určují jednoznačnou polohu na povrchu planety. Pro elektronickou reprezentaci jsou ovšem nepohodlné kvůli atributům světových stran. Proto se pro elektronickou reprezentaci liší. Zeměpisná délka nabývá hodnot od $-180°$ (západní polokoule) do 180° (východní polokoule) a obdobně zeměpisná šířka nabývá hodnot od $-90°$ (jižní polokoule) do 90° (severní polokoule). Díky tomu lze reprezentovat polohu jen pomocí dvou hodnot. Tyto hodnoty ovšem mohou mít více formátů, přičemž nejběžnější z nich jsou popsány v tabulce 2.1.

Je jistě patrné, že pro reprezentaci ve výpočetních technologiích je nejvhodnější formát DDD a nadále bude využíván právě tento. Protože se pohybujeme v trojrozměrném (dále jen 3D) prostoru, chybí ještě jedna souřadnice. Chybějící souřadnicí je nadmořská výška, která je reprezentována kladnou hodnotou nad povrchem Země, resp. hladinou oceánu, a záporně pod hladinou.

Tabulka 2.1: Nejznámější způsoby zápisu zeměpisných souřadnic.

Zkratka	*Zápis*	*Popis*
DMS	49°30′30″	zápis stupňů, minut a sekund tak, jak jsme zvyklí
DMM	49 30.5	minuty zapsány dekadicky
DDD	49.508333	stupně zapsány dekadicky
Zkratky jsou odvozené z anglických slov pro stupně (Degrees), minuty (Minutes) a sekundy (Seconds). [5]		

2.2 Kartografická projekce

Pokud známe polohu bodu, je často vhodné ji uživateli nějak znázornit. Papírové mapy i obrazovky zařízení jsou ale ploché. Zde nastupuje nutnost projekce, tedy proces zobrazení povrchu 3D tělesa na 2D plochu. Stejně jako tenisový míček nelze rozvinout do roviny bez deformací, nelze takto rozvinout ani povrch geoidu (matematicky popsané těleso, které nejvíce odpovídá reálnému tvaru naší planety) či jiného zjednodušeného referenčního tělesa, jako je elipsoid nebo koule. Proto se postupem času vyvinuly projekce, které převádí povrch takových referenčních těles na povrchy těles, které již jsou rozvinutelné do roviny, jako je např. válec. Vždy dochází k deformacím a ztrátě přesnosti, ale v závislosti na zvoleném zobrazení se deformují více vzdálenosti nebo úhly v různých částech planety.

Analýza kartografických zobrazení však není předmětem této práce. Z tohoto důvodu bude nastíněna pouze UTM projekce.

2.2.1 UTM projekce

Zkratka UTM vznikla z ang. Universal Transverse Mercator projection[4]. Mercatorova projekce patří mezi válcová zobrazení. Jejím základem je zobrazení povrchu Země na válec, který je elipsoidu tečný v místě rovníku, a jeho osa je totožná s osou planety. Z tohoto postupu vyplývá, že zobrazení je nevhodné pro zobrazení polárních oblastí. Přívlastek transverzální znamená, že válec není tečný v místě rovníku, ale v místě zvoleného poledníku. Gaussovo zobrazení (Mercatorovo transverzální) se od UTM liší již jen několika konstantami a typem referenčního elipsoidu

UTM není pouze zobrazení elipsoidu na válec. Je to síť vzniklá zobrazením šedesáti zón po 6° zobrazených pomocí transverzálního Mercatorova

zobrazení. Takto byl systém vyvinut v roce 1947 armádou Spojených států amerických pro tvorbu vojenských map. Dnes UTM využívá mj. i společnost Google pro tvorbu svých map. Tím se toto zobrazení týká i této práce, protože jím je dané omezení souřadnic. Podklady Google Map jsou díky UTM omezeny, co se týče zeměpisné šířky, přibližně od $-85°$ do $85°$[6]. Použití UTM také osvětluje možné odchylky v souřadnicích získaných za pomoci jiného zobrazení.

3 Metody určování zeměpisné polohy

Nyní již máme možnost, jak popsat pozici na planetě, pokud zrovna držíme v ruce mapu. Jak ale získat souřadnice místa, kde právě stojíme? Za tímto účelem se během let vyvinulo několik strategií, od primitivních, jako je znalost noční oblohy a ročního období, po dokonalejší, kterou je například GPS. Jelikož jsou primitivní strategie pro naši práci nepoužitelné, nebudeme se jimi vůbec zabývat. Pro určování polohy se dnes používají zejména triangulace a satelitní pozicování a jejich další deriváty.

3.1 Triangulace

Jak název napovídá, metody založené na triangulaci využívají matematických znalostí o trojúhelnících. Historicky se vyvinuly pravděpodobně z důvodu, že se v terénu lépe měřil úhel než vzdálenost. Například vzdálenost cíle od hradeb šla odvodit ze znalosti vzdálenosti dvou hradních věží a úhlem mezi hradbou a cílem, jak je znázorněno na obrázku 3.1. S takto změřenými úhly z obou věží lze skrze větu o velikosti součtu vnitřních úhlů odvodit poslední úhel. Znalost všech tří úhlů a délky jedné strany trojúhelníku je na výpočet ostatních stran, popřípadě výšek trojúhelníku postačující. A vzdálenost cíle od hradby je ve skutečnosti jednou z výšek trojúhelníku.

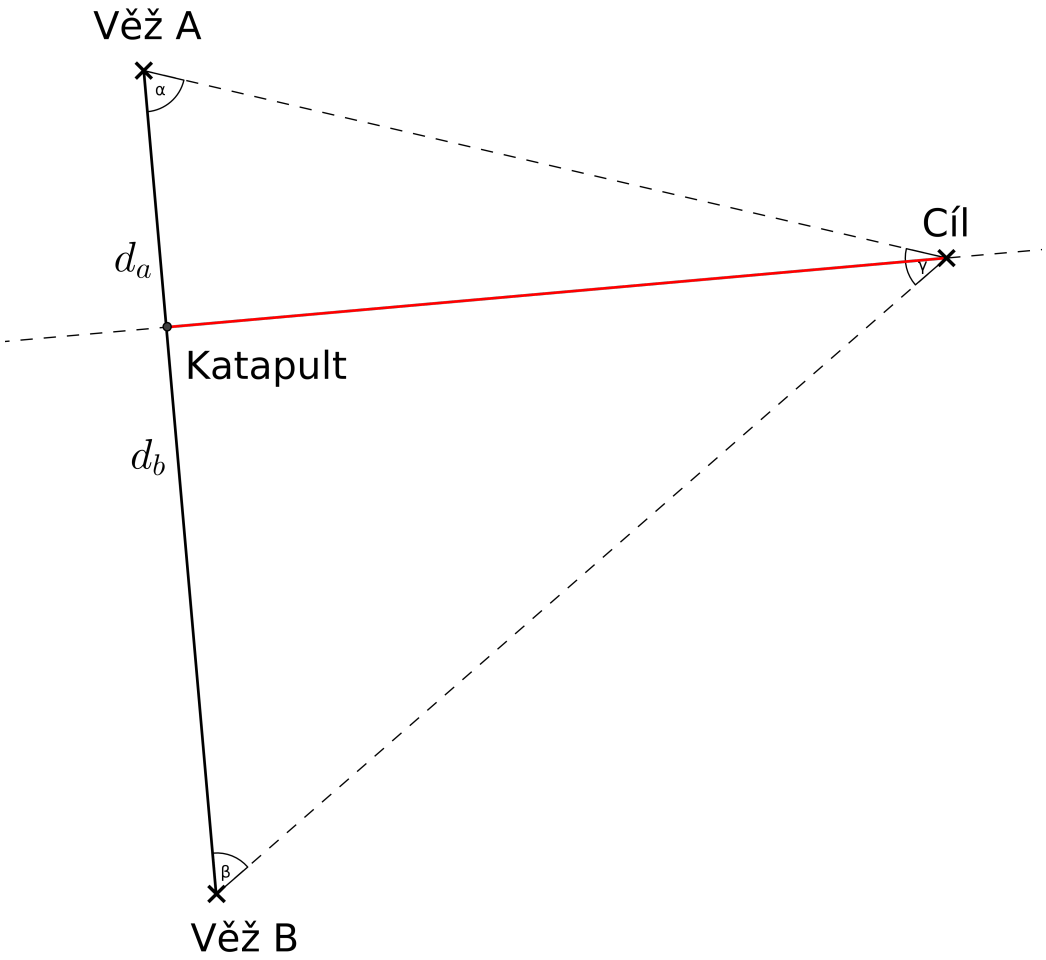
Obdobné postupy lze aplikovat i dnes, pokud jsou známy zeměpisné souřadnice referenčních bodů. Ve výše zmíněném příkladu by tedy stačilo navíc znát souřadnice obou věží. Triangulace se proto používá i dnes například při vytváření map, kdy se využívají pevně stanovené geodetické body. Dalším využitím je lokalizace mobilních zařízení.

3.1.1 Triangulace mobilního zařízení

Mobilní sítě jsou založeny na síti vysílačů. Tyto vysílače jsou pevná stanoviště a operátoři znají jejich polohu. Všechny vysílače mají svou unikátní identifikaci (dále jen Cell ID) a tuto lze spárovat s polohou. Pokud je telefon v dosahu signálu, je připojen na jeden z vysílačů, z kterého kromě dat získává i Cell ID. Telefon jakožto přijímač je schopný poskytnout i údaj o síle signálu. Vysílač vysílá do všech směrů nebo sektorově, takže je-li zanedbána

$$v = \sqrt{d_a \cdot d_b} \tag{3.1}$$

$$v = \tan\alpha \cdot d_a \tag{3.2}$$



Obrázek 3.1: Výpočet vzdálenosti cíle od hradeb. Vzorec 3.1 demonstruje využití Eukleidovy věty, vzorec 3.2 využití funkce tangens pro pravoúhlý trojúhelník.

nadmořská výška získáváme kruh, resp. kruhovou výseč značící dosah signálu vysílače.

3.1.1.a GSM síť

GSM síť (z fr. Global Spécial Mobile) se stala základním kamenem v mobilních komunikacích, díky němuž vznikly dnešní moderní standardy. Je založena na vysílačích (BTS) a mobilních telefonech (MT) jakožto přijímačích a dalších prvcích. Pro naše účely je důležité právě komunikace mezi MT a BTS.

BTS jsou osazeny vysílači s více frekvencemi, navíc jsou tyto vysílače zpravidla směrové (obr. 3.2), všesměrové vysílače již nejsou tak běžné. Komunikace funguje obdobně jako u vysílaček. Na jedné frekvenci tedy může ve stejný okamžik vysílat pouze jeden přístroj. Takto by mohl vysílač obsluhovat velmi málo MT, protože frekvence nelze navyšovat libovolně. V případě, že by dvě sousední BTS využívaly stejnou frekvenc,i docházelo by k interferenci. Proto je v GSM použit ještě časový multiplex.

Tím, že vysílač rozdělí příjem signálu na časová okénka, tzv. time sloty (TS), může na jedné frekvenci obsluhovat více MT. MT nahrává a následně komprimuje náš hlas po úsecích a odesílá komprimovaně na BTS v přidělených TS. BTS naslouchá v každém TS právě jednomu MT a ostatní ignoruje. Při komunikace z BTS do MT je systém analogický. Komunikace tedy není spojitá, ale funguje na principu "každý chvilku tahá pilku". Protože jsou TS tak krátké, hraje v přenosu roli už i vzdálenost mezi BTS a MT.

MT tak musí při větší vzdálenosti vysílat data v předstihu. Kvůli tomu musí BT a MT znát vzdálenost mezi sebou. Vzdálenost se v GSM určuje pomocí timing advance (TA). Tento parametr nabývá hodnot od 0 až do 63 a rozděluje dosah vysílače na pásma po 550 metrech. Vynásobením zjistíme, že v GSM se uvažuje maximální vzdálenost od vysílače 35 km kvůli rostoucí odchylce TA.

3.1.1.b Určení polohy jedním vysílačem

Pokud známe Cell ID vysílače, lze zjistit i jeho polohu a dosah. Tím již získáme pojem o poloze s přesností několika kilometrů. Pokud bychom znali i výkon BTS, můžeme uvážit i fakt, že síla signálu spolu se vzdáleností klesá.

To přináší další zpřesnění v podobě zjištění mezikruží, ve kterém se přístroj nachází. Další a používanější možností zpřesnění je využití TA parametru. Telefon poskytuje softwarové vrstvě jak TA, tak Cell ID. Se znalostí pozice BTS a vzdálenosti od ní určíme již mezikruží, ve kterém se MT pohybuje. V případě, že je vysílání BTS rozděleno na více sektorů, lze průnikem kruhové výseče a mezikruží polohu ještě zpřesnit. Z popisu metody je patrné, že se o triangulaci jako takovou nejedná.

3.1.1.c Určení polohy dvěma vysílači

Tato metoda vychází z předchozí. Data o BTS navíc umožňují zpřesnění, protože dvě zóny se někde protnou. V ideálním případě vznikne průnikem jediná relativně malá oblast a v oblastech větších měst již můžeme získat přesnost v řádech stovek metrů. V horším případě se mezikruží protnou a vzniknou dvě oddělené oblasti. V takovém případě je nutné získat nějaký pomocný bod, aby bylo možné určit, která oblast je námi hledaná. Další možností je upustit od mezikruží a jako výslednou oblast chápat průnik kruhů dosahu signálů (viz obr. 3.3). Jako výsledná pozice se volí geometrický střed oblasti.

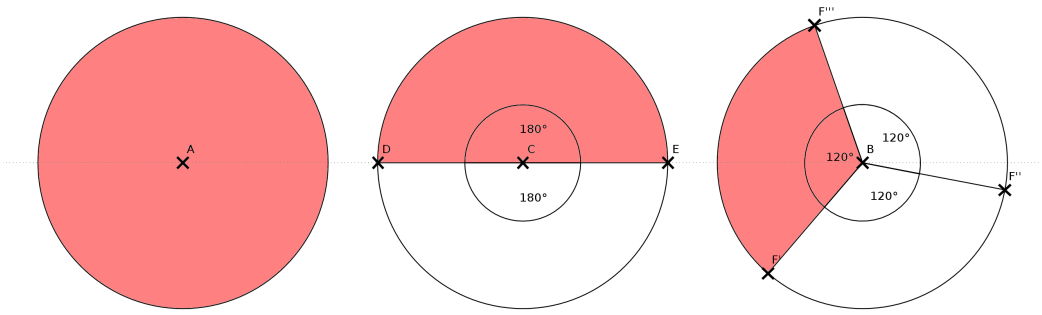
3.1.1.d Určení polohy více vysílači

Čím více vysílačů je k dispozici, tím menší je oblast vzniklá průnikem signálů (viz obr. 3.4).

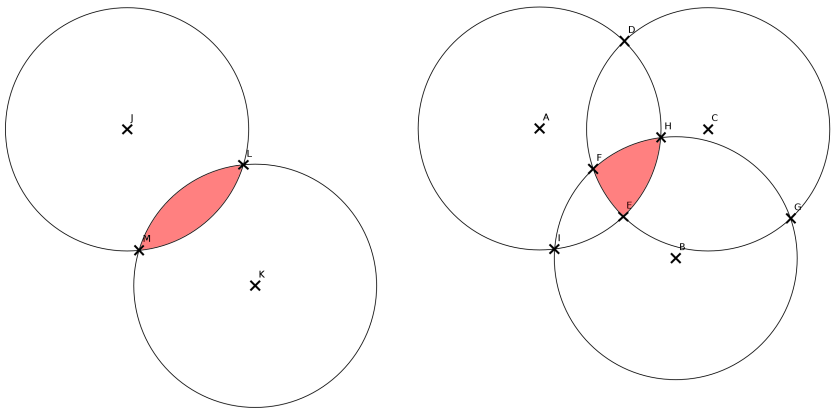
3.1.1.e Wi-Fi místo mobilní sítě

Výše zmíněné postupy jsou použitelné pouze pro MT. Zařízení bez možnosti příjmu síťového signálu mohou využívat Wi-Fi, která má podobné vlastnosti. Navíc dosah signálu je menší a tedy je přesnější. Proto lze k lokalizaci využívat obdobně i Wi-Fi signál.

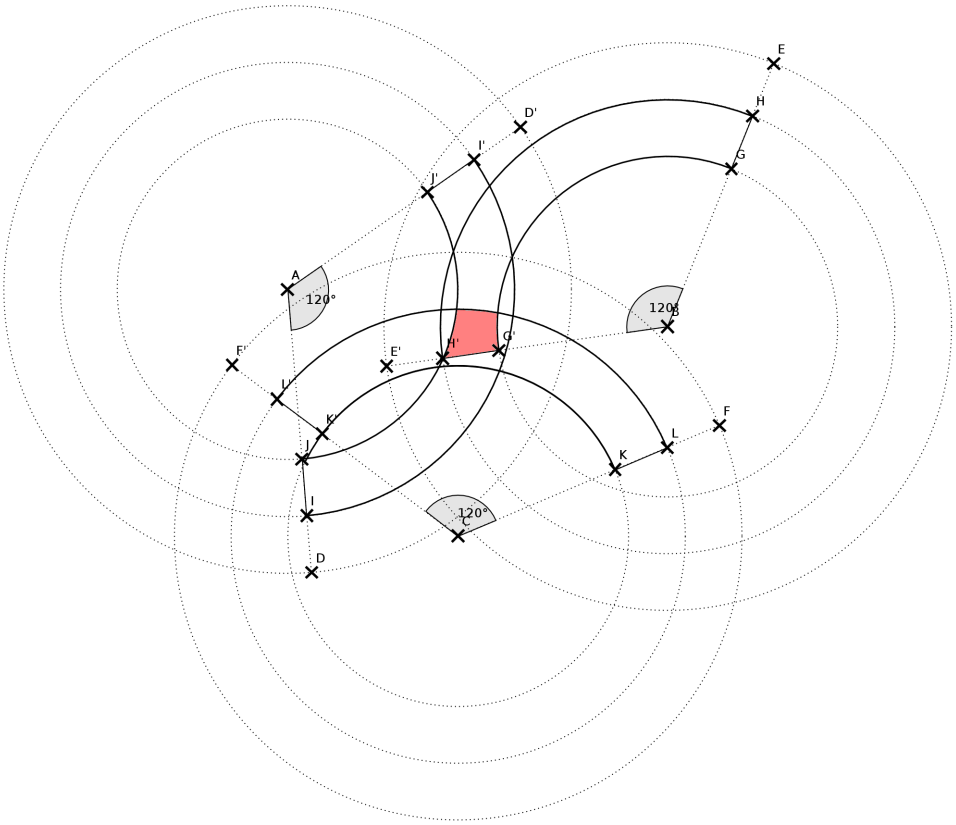
Wi-Fi je standard lokálních bezdrátových sítí dle specifikace IEEE 802.11. Vysílač připojený k internetu zprostředkovává zařízením v blízkém okolí bezdrátový přístup k síti. Běžně jsou využívána pásma 2,4 a 5 GHz. Vysílač je sdílený pro připojená zařízení a proto používá protokol CSMA/CA. Přístupový bod Wi-Fi je v jistém smyslu podobný BTS. Lze jej identifikovat dle jeho MAC adresy a díky tomu lze vytvářet databáze pozic přístupových bodů.



Obrázek 3.2: Typy vysílačů.



Obrázek 3.3: Průniky dosahů vysílačů.



Obrázek 3.4: Oblast průniku signálů zpřesněná TA a úhlem.

Bod C představuje BTS a úhel $F'CF$ představuje signál jednoho směrového vysílače. Bod F je ve vzdálenosti maximálního dosahu BTS C. Oblouk KCK' tvoří vnitřní hranici TA, oblouk LCL' hranici vnější. Obdobně je to pro BTS A a B. V červené oblasti se nechází MT.

3.1.1.f Shrnutí

Metoda je nenáročná na zdroje zejména z pohledu baterie. Pro rychlé zjištění polohy s větší přípustnou odchylkou je velmi vhodná. V porovnání s GPS krátce po zapnutí GPS na přístroji či v oblastech s velkým počtem vysokých budov je dokonce přesnější a rychlejší než GPS, zejména využívá-li ke zjištění polohy Wi-Fi. Nevýhodou je velká nepřesnost v oblastech s menším počtem vysílačů.

Jen málo zařízení zpřístupňuje seznam BTS v dosahu i softwarové vrstvě. V drtivé většině případů jsou softwaru zpřístupněny pouze údaje o aktuálně připojené BTS a tím lze využít pouze první metodu. Zeměpisné souřadnice BTS a jejich výkon operátoři neposkytují. Je tedy nutná jiná databáze, buď interní v přístroji nebo přístupná skrz internet. Příklady takových databází resp. služeb jsou zaneseny v tabulce 3.1.

Tabulka 3.1: Poskytovatelé souřadnic vysílačů.

Název	*Dostupnost*	*Poznámky*
Geolocation API (Google)	Placené (ceny až po kontaktu s prodejcem)	Poskytuje souřadnice na základě JSON dotazu na webovou službu. Operuje s Cell ID i Wi-Fi MAC adresou. Další parametry jsou nepovinné a slouží ke zpřesnění polohy. Aktuální počet záznamů v databázi není znám. [11]
Navizon	Placené (ceny až po kontaktu s prodejcem)	Nabízí nejen určování polohy přes Wi-Fi a mobilní sítě mimo budovy, ale také tzv. Indoor řešení pro určení polohy uvnitř budov na základě Wi-Fi signálu. Služba funguje obdobně jako výše zmíněná přes HTTP protokol. Aktuální počet záznamů v databázi není znám.[12]
Location Api (Unwired Labs)	Placené (cena od 145$/měsíc)	Služba opět funguje na bázi JSON dotazů na servery. Zaměřuje se pouze na mobilní Cell ID. Databáze čítá 43.7 mil. záznamů a stále roste.[13]
OpenCellID	Bezplatné	Služba se zaměřuje pouze na mobilní Cell ID. Data jsou volně dostupná ve formě aktualizovaných csv souborů inkrementálně, tj. neni nutné pokaždé stahovat celou databázi. Pokud se tedy importuje databáze přímo do přístroje, lze tato data využívat kompletně bez využití internetu, což je proti předešlým výhoda. Navzdory faktu, že se jedná o databázi rozšiřovanou dobrovolníky, přesáhl počet záznamů již 4 mil.[22]

3.2 GPS

Global Positioning System [9] spravovaný Spojenými státy americkými využívá jako jádro 24 satelitů kroužících kolem Země po různých orbitách ve výšce přibližně 20 000 km. Dnes krouží kolem Země více těchto satelitů jak kvůli zlepšení přístupnosti, tak kvůli obměně satelitů. Satelity mj. obsahují velmi přesné atomové hodiny. Satelity vysílají, data z nichž je část šifrovaná. Veřejná část je pro civilní účely, šifrovaná je pro vojenské. Díky tomu si Spojené státy ponechaly možnost např. dočasně snížit přesnost civilní navigace v případě konfliktu.

Základní složkou vysílaného signálu je časová značka pocházející z hodin. Protože se satelity pohybují po deterministických orbitách, je možné dle časové značky a podpisu satelitu určit, odkud signál přišel. Díky vzdálenosti dochází ke zpoždění signálu. Přijímač GPS přijme časovou značku a porovná ji se svým vnitřním časem. Z rozdílu je schopen spočítat vzdálenost. Pokud tedy přijímač zachytí signál od 3 nebo více satelitů, je schopen určit svou polohu obdobně jako u triangulace BTS signálu. Ve skutečnosti známe body a vzdálenosti od nich, čímž získáváme průnik 3 či více koulí. Čím více signálů, tím bude poloha přesnější.

Je patrné, že tato metoda má své nedostatky. Jedním z nich je fakt, že při přenosu signálu může dojít ke zpoždění kvůli odrazu o překážku, jak se tomu děje hlavně v zastavěných oblastech. Na výpočet má vliv také to, je-li přijímač v pohybu. V neposlední řadě je problematický i fakt, že hodiny přijímače nejsou tak přesné jako atomové hodiny družice. Systém tedy nejlépe funguje v místech s dobrým výhledem na oblohu. Z toho vyplývá, že při použití v budovách chyba narůstá nebo se dokonce nepovede spojit s dostatečným počtem satelitů.

Dalším problémem je skutečnost, že po zapnutí GPS MT netuší o pozicích satelitů nic. Ze zachycených signálů musí zjistit, o jaký satelit se jedná a odkud vysílá. Tyto výpočty slouží k určení, které satelity mohou být využity ke stanovení polohy. Výpočty trvají dlouho, takže GPS přijímač je nějakou dobu po zapnutí neschopný určit polohu. Složité výpočty navíc spotřebovávají cennou energii z baterie.

3.2.1 A-GPS

Řešení je tzv. asistovaná GPS. Zařízení vybavená touto technologií využívají internet k tomu, aby odeslala zachycené signály a přibližnou polohu dle BTS na stroj s daleko větším výkonem. Ten již vrací informace o polohách satelitů a jejich vhodnosti k určení polohy. Tímto krokem se ušetří energie a zkrátí čas nutný k nastartování GPS přijímače.

3.2.2 Další satelitní systémy

GPS není jediný systém svého druhu. V Rusku vznikl GLONASS [10] sestávající z 24 satelitů, které se pohybují ve výšce 19 000 km. Systém byl původně vyvinut také jako vojenský. Dalším krokem tedy bylo nasazení přijímačů, které jsou schopné využívat k určení polohy jak satelity GLONASS, tak satelity GPS. Více satelitů znamená více signálů a tedy větší pravděpodobnost, že zkoumaný signál bude vhodný k určení polohy a MT se na něj může zamknout a to přináší rychlejší start lokalizace. Další výhodou je zpřesnění určení polohy.

Dalším takovým systémem je Galileo [8]. Jedná se o evropský ekvivalent GPS a GLONASS, který není vojenský a spravuje jej více evropských států. Systém je plánován pro 30 satelitů, ale v aktivním provozu jsou zatím 4.

3.3 Další možnosti určování polohy

MT v dnešní době obsahují mimo jiné i různé senzory.

Polohové určující aktuální polohu, resp. náklon zařízení.

Pohybové určující vektor zrychlení zařízení při pohybu.

Magnetometrické suplující kompas.

Jiné měřící např. teplotu, úroveň světla či vlhkost.

Využití kompasu jako takového asi není nutné popisovat. Zajímavější je ovšem senzor detekující vektor zrychlení. Tento způsob využívají například

některé navigace v případě, že vypadne signál GPS. Poslední známou pozici a rychlost uvažují jako výchozí bod. Další pohyb je vypočítáván pomocí vektoru zrychlení přístroje. Jedná se pouze o výpočet přibližného pohybu, protože senzory v MT jsou pro tuto metodu dosti nepřesné. Sám jsem provedl pokus a měřil hodnoty vektoru zrychlení u MT ležících nehybně na podložce. Všechny směrové složky vektoru kmitaly a neustalovaly se.

Tato metoda navíc nepočítá s jevy, jako je setrvačnost pohybu, takže při prudším brzdění vzniká další nepřesnost pokud není telefon pevně spojen s automobilem. Pro použití v pěší navigaci by musela být aplikace přizpůsobena konkrétnímu uživateli, aby byla schopna rozpoznat např. chůzi do schodů, běh apod. Toho by bylo možné docílit učením aplikace v terénu. Nicméně i v tomto případě by se projevily nepřesnosti senzoru. Tento typ určování polohy by mohl posloužit jako základ další práce a nebude zde již dále rozebírán.

Popsané metody jsou často zahrnuté v API systémů zařízení, takže vývojář pouze musí zvážit, kterou z metod použít a API následně vrátí zeměpisnou polohu.

4 Využití lokalizace mobilních zařízení

Možností využití lokalizace MT je mnoho a určitě nedokážu postihnout všechny. Pokusím se tedy přiblížit ty, které považuji za nejvýznamnější.

Navigace uživatele mezi stanovenými body ať při jízdě autem, na kole nebo při chůzi. MT dnes dokáží nahradit navigační přístroje. Toto využití je limitováno baterií (zejména v případě využití pro cyklistiku a pěší chůzi) a objemem stahovaných dat. Navigace pro telefony jsou totiž v základě dvojího typu:

- Online využívající služby na internetu jako zdroj grafiky i pro výpočty.
- Offline provádějící výpočty na přístroji a využívající mapové podklady uložené na přístroji.

Geocoding je proces, při kterém jsou adresy, názvy podniků nebo firem či jiná klíčová slova převáděna na zeměpisné souřadnice. Stejně jako u navigace mohou být zdroje online i offline.

Reverzní geocoding je opak předchozího procesu. K zadaným zeměpisným souřadnicím jsou vyhledávány ulice anebo tzv. POI (points of interest) body, což jsou firmy, kavárny, muzea atd.

Cílená reklama, kdy je na reklamní plochy uvnitř aplikací umisťován reklamní obsah firem z blízkého okolí.

Sdílení polohy s jinými uživateli stejné aplikace. Tento princip nachází využití např. v seznamkách, hlídání dětí rodiči nebo hlídání zaměstnanců atp.

5 Vývoj pro platformu Android

Protože OS Android vyvinula a vyvíjí společnost Google, ta samá společnost dává k dispozici i vývojové nástroje. Dlouhou dobu byly tyto nástroje zřejmě jedinou použitelnou variantou. Tato situace se ale změnila. Objevují se další možnosti, jak vyvíjet pro tuto platformu.

Android SDK nejrozšířenější a nejznámější způsob, jak vyvíjet aplikace pro Android. Jako jazyk se využívá Java. Spolu s dalšími balíčky (např. emulátory, ladící nástroje) tvoří plugin pro vývojové prostředí Eclipse. Další možností je stáhnout si rovnou upravenou verzi Eclipse, tzv. ADT Bundle nebo alternativu Android Studio, což je upravená verze IntelliJ IDEA.

Android NDK umožňující implementaci nativního kódu v C nebo C++. Tyto nástroje Google doporučuje používat s rozmyslem, protože zejména správa paměti není na OS Android jednoduchá. Vhodné využití nachází pro náročné úkoly pro procesor s minimem alokované paměti.

Xamarin umožňující vývoj aplikací v jazyce C# a s prostředky .NET. Jako vývojové prostředí lze využít Visual Studio. Navíc umožňuje i přenositelnost aplikací mezi OS Android, iOS a Windows.

Qt velmi populární framework pro implementaci uživatelsky příjemných aplikací v C++. Nyní lze tuto technologii využít i pro implementaci aplikací pro Android. I zde je možná přenositelnost.

PhoneGap je framework umožňující skrze známé technologie CSS, HTML a Javascript přistupovat k API různých systémů, jako je Android, iOS, Symbian a další.

Android SDK je nejrozšířenějším nástrojem a je tedy i nejlépe dokumentován ([1], [2]), proto bude ve zbývající části práce používána práve tato technologie.

5.1 Lokalizace zařízení s OS Android

Možnosti jsou závislé od hardwaru MT, ale operační systém (OS) Android umožňuje vývojáři vysokoúrovňový přístup k informacím z GPS, resp. GLO-

NASS, vysílačů mobilní sítě, Wi-Fi sítí i pohybových senzorů. Nízkoúrovňové informace sice zpřístupňuje také, ale ne všechny. Např. nám poskytne Cell ID, ale zeměpisné souřadnice již ne. V takovém případě máme možnost dotázat se jedné ze služeb z tabulky 3.1.

5.1.1 Google Play Services

Než se začneme věnovat další problematice, je nutné rozebrat tuto knihovnu [14]. Nebyla tu po celou dobu existence OS Android, přišla později a jejím hlavním úkolem je ulehčení práce vývojářům. Nabízí mj.

- jednodušší přístup ke službám Googlu, jako Mapy, Google+ aj.
- updaty aplikací
- implementaci často řešených problémů, jako black box (získání polohy od nejlepšího poskytovatele)

Při používání této knihovny je však nutné uvědomit si, že některé custom ROM[1] nejsou schopné ji nainstalovat a tím nemohou vaši aplikaci využívat. O tomto problému jsem se sám přesvědčil na Samsung Galaxy Y s upravenou ROM Titanium2.

Knihovna byla uvedena až v průběhu vývoje aplikace, což mělo za následek změnu některých konstrukcí. Kvůli tomu a výše zmíněnému problému bude v práci občas uvedena i alternativa bez použití této knihovny.

5.1.2 Kritéria

MT nabízejí více zdrojů, jak získat polohu. Aplikace se tedy musí nějak dozvědět, z kterého zdroje získávat informace. To lze vynutit přímo specifikací konkrétního zdroje nebo použitím objektu *Criteria*. Ten v sobě vývojářem udržuje nastavené preference na spotřebu energie, přesnost, rychlost získání polohy atp. Objekt *LocationManager* je pak na základě preferencí schopný vybrat nejlepšího providera (viz listing 5.2). Android aktuálně využívá 3 providery polohy (viz tabulka 5.1).

[1]Upravená verze OS Android zpravidla vyvinutá nadšenci s cílem vylepšit funkce OS nebo dostat na stará zařízení vyšší verze OS Android.

Tabulka 5.1: Provideři zeměpisné polohy OS Android [16].

Zdroj polohy	*Popis*
GPS_PROVIDER	zastřešuje zdroje polohy z GPS a GLONASS
NETWORK_PROVIDER	umožňuje přístup k souřadnicím získaným díky mobilní síti a Wi-Fi sítím
PASSIVE_PROVIDER	využívá jako zdroje polohy jiné aplikace, tj. sám neumožňuje získání souřadnic ze sítě nebo GPS

5.1.3 Třída LocationListener

Vysokoúrovňový přístup k získávání polohy, jak název rozhraní napovídá, funguje na principu posluchače, který se zaregistruje spolu se svým nastavením v systému. Tento posluchač je nadále notifikován o změnách polohy a reaguje na ně. Tuto třídu využívá jak řešení bez využití Google Play Services, tak i s využitím této knihovny. Pro použití je nutné buď implementovat rozhraní jinou třídou nebo v místě použití vytvořit anonymní třídu (viz listing 5.1).

Implementovaného posluchače stačí již jen přihlásit pro odběr změn zeměpisné polohy. Pro tyto účely slouží přetížená metoda *requestLocationUpdates()* (viz listing 5.3 a 5.4).

Metoda nabízí i další varianty, např. s využitím objektu *PendingIntent*, který umožňuje vykonat jiné aplikaci kus kódu s právy aktuální aplikace. Knihovna Google Play Services umožňuje zjednodušení, které bude popsáno v analýze frameworku.

Listing 5.1: Jednorázové získání polohy s využitím kritérií.

```
//ziskani LocationManagera jakozto sluzby systemu
LocationManager mgr = (LocationManager)
    getSystemService(LOCATION_SERVICE);
//nastaveni preferenci pro ziskavani polohy
Criteria kriteria = new Criteria();
//vyzadani nejen lokace ale i smeru kompasu
kriteria.setBearingRequired(true);
kriteria.setBearingAccuracy(Criteria.ACCURACY_LOW);
//pouze bezplatne sluzby a vysoka presnost
kriteria.isCostAllowed(false);
kriteria.setAccuracy(Criteria.ACCURACY_FINE);
//vysledny identifikator ma podobu retezce
String nejlepsi = mgr.getBestProvider(criteria, true);
//ziskani posledni zname polohy ze zdroje urceneho na zaklade
    preferenci
Location location = mgr.getLastKnownLocation(best);
```

Listing 5.2: Implementace rozhraní LocationListener anonymní třídou.

```
LocationListener ll = new LocationListener() {
        public void onLocationChanged(Location location){
                //rekace na novou polohu napr. posunuti mapy
        }

        public void onProviderEnabled(String provider){
                //reakce na zapnuti providera polohy
                //nutne pro implementaci rozhrani, metoda muze byt
                    prazdna
        }

        public void onProviderDisabled(String provider){
                //reakce na vypnuti providera polohy
                //nutne pro implementaci rozhrani, metoda muze byt
                    prazdna
        }

        public void onStatusChanged(String provider, int status,
            Bundle extras){
                //reakce na zmenu stavu providera
                //napr. pokud provider najednou neni schopny delsi
                    dobu ziskat polohu
                //jako v pripade GPS uvnitr budovy
                //nutne pro implementaci rozhrani, metoda muze byt
                    prazdna
        }
}
```

Listing 5.3: Odběr změn lokace s vyžádaným providerem.

```
LocationManager mgr =
    (LocationManager)getSystemService(LOCATION_SERVICE)
//implementace neni dulezita
LocationListener ll = new LocationListener(){...}
//minimalni cas pro obdrzeni nove lokace
//0 znamena, ze bude obdrzena kazda zmena
int minCas = 0;
//minimalni zmena vzdalenosti pro obdrzeni nove lokace
float minVzdalenost = 0;
//prihlaseni ll pro odber zmen lokace
mgr.requestLocationUpdates(LocationManager.GPS_PROVIDER, minCas,
    minVzdalenost, ll);
...
//zruseni odberu zmen lokace
mgr.removeUpdates(ll);
```

Listing 5.4: Odběr změn lokace s providerem voleným systémem.

```
LocationManager mgr =
    (LocationManager)getSystemService(LOCATION_SERVICE)
//implementace neni dulezita
LocationListener ll = new LocationListener(){...}
//minimalni cas pro obdrzeni nove lokace
//0 znamena, ze bude obdrzena kazda zmena
int minCas = 0;
//minimalni zmena vzdalenosti pro obdrzeni nove lokace
float minVzdalenost = 0;
//vytvoreni defaultnich preferenci
Criteria kriteria = new Criteria();
//prihlaseni ll pro odber zmen lokace, provider bude zvolen
    systemem dle preferenci
//posledni parametr null nahrazuje objekt Looper
//proto bude notifikovano pouze volajici vlakno
mgr.requestLocationUpdates(minCas, minVzdalenost, kriteria, ll,
    null);
...
//zruseni odberu zmen lokace
mgr.removeUpdates(ll);
```

6 Location Framework

Předmětem této práce je framework pro zjednodušení implementace aplikací využívajících zeměpisnou polohu (location-based aplikacích, dále jen LB). Měl by tedy zahrnovat funkcionality, které se často vyskytují v těchto aplikacích. Dále by měl splňovat požadavek jednoduché modifikace nebo rozšíření.

LB aplikace kromě zjištění polohy potřebují tuto polohu také vizualizovat nebo sdílet. Pro vizualici je vhodná mapa a pro sdílení internet. Proto framework obsahuje balíčky pro práci se sítí, s mapovými podklady i s daty uživatelů.

6.1 Modely

V lokalizačních aplikacích je vhodné udržovat polohu objektu (i v případě pouze vaší polohy) spolu s dalšími informacemi. Protože je Java objektově orientovaný (dále OO) jazyk, nabízí se varianta použití modelových tříd. Tyto třídy mají bezparametrický kontruktor, gettery a settery pro své atributy (zpravidla primtivní datové typy). Toto řešení má hned několik důvodů:

- přehlednost návrhu
- snadné rozšíření tříd o další atributy
- možnost použití knihoven třetích stran pro serializaci a deserializaci dat

Tyto třídy jsou obsaženy v balíčku *Objects*. Balíček obsahuje třídy, které se snaží postihnout základní potřeby pro LB aplikace.

Point obsahuje atributy pro popis, identifikaci bodu na mapě.

Player je potomkem třídy *Point* a obsahuje navíc atributy pro popis stavu a jména.

Group umožňuje shluknout instance *Point* či její potomky do slovníku, přičemž jako klíč slouží jejich id. Protože je klíč datový typ integer, je zde místo *HashMapy* využito třídy *SparseArray*, která umožňuje jako klíč slovníku využít právě primitivní datový typ integer.

Game shlukuje instance třídy *Group* opět pomocí *SparseArray*. Umožňuje také uložení kritérií hry, jako např. prostor vymezený souřadnicemi.

JSONPuzzle je společný předek pro všechny předcházející třídy. Obsahuje pouze dvě metody pro serializaci a deserializaci objektů do, resp. z, formátu JSON, čehož se využívá při přenosu dat.

6.2 Vizualizace na mapě

V okamžiku, kdy jsou k dispozici modely a jejich zeměpisné souřadnice, je v řadě aplikací vhodné je vizualizovat na mapě. Mapa v tomto případě neznamená pouze obrázek místa na Zemi. Jedná se o komplexní řešení zahrnující:

- umístění ukazatele na mapu dle souřadnic
- zjištění zeměpisných souřadnic konkrétního pixelu na obrazovce
- pohyb a zoom mapy
- kreslení lomené čáry (tzv. polyline) na mapy dle zadaných zeměpisných souřadnic

Všechny tyto funkce pracují nad mapou jako takovou. Ovšem proč s sebou nosit celý atlas, pokud nám stačí mapa města? Ve skutečnosti je mapa, kterou vídáme na obrazovkách zařízení, složena z obrázků, tzv. dlaždic. Dlaždice lze jednoznačně identifikovat na základě zoomu, zeměpisných souřadnic a typu zobrazení (3D, satelitní, turistické atd.). Implementací existuje mnoho, lze je však rozdělit zhruba do několika skupin.

XHTML řešení využívající k zobrazení a práci s mapou vestavěný prohlížeč s jádrem WebKit[1]. Mezi taková řešení patří například Google Maps V3 či Leaflet [20], která stojí na Javascriptu, CSS3 a HTML. Tato řešení jsou tím pádem použitelná i pro desktopové aplikace. Vhodnější je dle mého názoru spíše Leaflet, protože již při návrhu bylo počítáno s tím, že technologie bude určená i pro telefony a tablety. Nevýhodou tohoto návrhu je nárůst objemu dat stažených i odeslaných prostřednictvím

[1]Renderovací jádro pro prohlížeče využívající prohlížeče jako Google Chrome, Safari nebo Opera

internetu. Kladem je fakt, že pokud bude celá aplikace implementována ve vestavěném prohlížeči, je snadno přenositelná na jiné platformy.

Kompletně offline řešení, která se nespoléhají na žádné další služby ani v otázce výpočtů ani v otázce mapových podkladů. Tato řešení jsou vhodná pro navigace, protože vyžadují pouze prvotní připojení k internetu pro instalaci a stažení mapových podkladů. Z toho vyplývá, že aplikace včetně podkladů může zabrat stovky MB na úložišti.

Hybridní model využívající výhody obou předchozích metod.

6.2.1 Integrace Google Maps Android API

Ve frameworku je využito veřejného API od Googlu pro zjednodušení práce s mapami.

Patří mezi hybridní varianty a je v aktuální verzi 2 (V2). Verze 1 (V1) již není podporována, v době započetí práce však byla aktuální. V1 byla méně přívětivá, jelikož funkce i mapové podklady byly pevně svázány. Pro zobrazení mapy se využíval widget[2] *MapView*. To nutilo vývojáře přijmout politiku Googlu ohledně využití map, což mj. znamenalo zákaz ukládání nebo modifikace dlaždic a omezený počet bezplatných stažení dlaždic. Pokud vývojářům z nějakého důvodu tato pravidla nevyhovovala, museli hledat alternativy nebo přijít s vlastní implementací.

Jednou z alternativ byl *OSMDroid*, což byla alternativní implementace *MapView* využívající jako zdroj dlaždic projekt *OpenStreetMap* (dále OSM) [21]. OSM je projekt nabízející dlaždice mapy světa pod otevřenou licencí. Dlaždice jsou tvořeny dobrovolníky a tvoří tak alternativu k podkladům od Googlu, protože je lze modifikovat či ukládat.

Mapy V2 již obsahují lepší řešení. Mapa se zobrazuje do widgetu *Fragment* a zdroj dlaždic lze specifikovat předáním instance rozhraní *TileProvider*. Ta musí implementovat jedinou metodu *getTile(int x, int y, int zoom)*, která vrací dlaždici pro zadané parametry. Co bude zdrojem dlaždic, je již na vývojáři. Pro OSM je to URL serveru, který vrací dlaždice. Další možností je čerpat dlaždice přímo z úložiště, což přináší rychlejší zobrazení dlaždice a možnost využití i mimo dosah mobilní sítě.

[2]Prvek uživatelského rozhraní jako např. tlačítko nebo editační pole.

6.2.2 Rozšiřující API frameworku

Třídy zjednodušující vizualizace jsou obsaženy v balíčku *Maps*.

SimpleUrlTileProvider je implementace abstraktní třídy *UrlTileProvider*. Vyžaduje pouze URL adresu serveru, který poskytuje dlaždice. Servery zpravidla fungují tak, že je volána URL ve formátu *http://server.com/z/x/y.png* a server vrací jako odpověď dlaždici. Proto instance vyžaduje URL serveru, ve které jsou za proměnné *x, y a z* uvedené placeholdery[3], za které jsou pak konkrétní proměnné dosazované při dotazu na server. Dalšími parametry jsou předpokládané rozměry dlaždic.

MixedTileProvider umožňuje oproti předchozí třídě jednu zásadní věc: ukládat dlaždice do úložiště. Funguje tak, že jako parametr vyžaduje instanci *UrlTileProvideru*. V okamžiku, kdy je vyžadována dlaždice, je nejdříve prozkoumáno úložiště. Pokud dlaždice v úložišti není, je stažena prostřednictvím *UrlTileProvideru*, zobrazena a uložena v úložišti pod svými souřadnicemi. Dalším parametrem je časový údaj, který říká, po jakou dobu jsou dlaždice chápány jako aktuální. Neaktuální dlaždice bude v případě požadavku přepsána nově staženou verzí.

MapObjectsManager zjednodušuje správu objektů na mapě. Objekt mapy umí zobrazovat markery a kreslit na mapu, bohužel tyto objekty nejsou bez uchování reference dále přístupné. Manager obsahuje seznamy, v kterých tyto objekty uchovává a lze přes ně i iterovat. Do listů lze vkládat a odebírat a gettery vrací seznamy pouze v readonly podobě.

RectangleBoundsMarkerDragListener implementuje rozhraní *OnMarkerDragListener*, které umožňuje reagovat na události při drag & drop pohybu s markerem po mapě. Tato třída poskytuje dva markery, díky kterým je na mapě vykreslen obdélník. Pohybem s markery se mění tvar obdélníku. Toto je užitečné pro vymezování hranic např. pro hru nebo pro při hlídání dětí.

CircleBoundsMarkerDragListener je obdobou předchozí třídy. Liší se pouze tím, že dva markery určují střed a poloměr kruhu.

BoundedOnCameraChangeListener je implementace rozhraní *OnCameraChangeListener*, které umožňuje reagovat na událost přesunu po mapě

[3]Charakteristické textové sekvence, ktelé lze snadno najít a následně nahradit jiným textem.

skrze metodu *onCameraChangeListener(CameraPosition position)*. Jak název napovídá, umožňuje třída vymezit zeměpisný prostor, ve kterém se může uživatel pohybovat. Tato skutečnost pomáhá snížit objem stažených dat, protože pokud pohled mapy nevystoupí z omezeného prostoru, nebudou se stahovat ani dlaždice mimo tento prostor. Problémem při implementaci se ukázal být efekt plynulého zpomalení používaný při scrollování v systému Android. Spočívá v tom, že pokud uživatel na dotykové obrazovce pohne rychle prstem, je mapa přesunuta stejným směrem a odpovídající intenzitou. Díky tomu se mi nedařilo docílit toho, aby byla mapa limitována striktně na vymezený prostor, protože odstranění efektu mělo za následek nepohodlné zacházení s mapou pro uživatele. Zvolil jsem tedy strategii, která tento efekt ponechává. Rozhraní posluchače je implementováno tak, že při ukončení pohybu mapy kontroluje pozici středu zobrazení. Pokud je tento bod mimo vymezené hranice, je mapa vycentrována na předem stanovené souřadnice. Grafika tak zůstala plynulá a ovládání uživatelsky příjemné.

6.3 Knihovny

Ve frameworku samozřejmě nesmějí chybět ani konstanty nebo knihovní funkce. Ty jsou obsažené v balíčku *Libs*. Při rozšiřování frameworku nebo jeho využití jsem počítal s tím, že budou třídy z balíčku editovány popř. další knihovní třídy budou umístěny pro přehlednost právě sem.

GameStates je třída sloužící k umístění konstant týkajících se hry. Ve frameworku obsahuje jen konstantu pro nespecifikované ID hry.

PlayerStates je obdoba předchozí třídy, ale je určena pro konstanty týkající se hráčů, tj. instancí třídy *Player*. Ve frameworku obsahuje pouze konstantu pro nespecifikované ID hráče.

SimpleStringAESCrypto je knihovní třída obsahující metody pro symetrické šifrování řetězců algoritmem AES na základě poskytnutého řetězce - klíče. Z klíče je vygenerován pomocí SHA256 hash, který je následně využit pro šifrování, resp. dešifrování. Hlavním důvodem pro vznik této knihovny byla možnost poskytnout vývojářům zabezpečit síť'ovou komunikaci bez využití protokolu HTTPS.

EasyMap je knihovní třída obsahující metody pro

- výpočet reálné vzdálenosti dvojice zeměpisných souřadnic
- konverzi zeměpisných souřadnic reprezentovaných starší třídou *Location* na novější reprezentaci *LatLng*
- výpočet koordinátů dlaždice ze zeměpisných souřadnic a zoomu, což lze využít pro stažení dlaždic předem bez nutnosti využití widgetu mapy
- získání parametrů pro vytvoření instance *LatLngBounds*[4] z dvojice zeměpisných souřadnic nezávisle na vzájemné poloze
- vytváření kopií instancí *PolygonOptions* a *CircleOptions*, které se využívají pro drag & drop vymezování hranic, ovšem bez zkopírování konkrétních zeměpisných bodů

GlobalVars je třída využívající návrhový vzor jedináček, která umožňuje ukládání globálních objektových proměnných. Třída ukládá pod Stringové klíče libovolné instance objektů a umožňuje správu této kolekce po celou dobu běhu aplikace. Data však nejsou perzistentní a dochází k jejich ztrátě po vypnutí přístroje.

6.4 Komunikace

Důležitou a velmi proměnlivou součástí LB aplikací je sdílení pozice a dalších dat. Při návrhu frameworku bylo nutné zvážit několik hledisek.

6.4.1 Spojení s aplikací

Aplikace pro Android se mj. skládají z tzv. aktivit. Aktivita je v podstatě uživatelské rozhraní, na které se díváme. Jakmile je na telefonu něco zobrazeno souvisí to s aktivitou. Aktivita má své vlákno, které je spjaté s uživatelským rozhraním. Pokud se rozhodneme v tomto vlákně spustit dlouhý výpočet nebo např. stahování dat, přestane uživatelské rozhraní reagovat, protože bude vytížené právě tímto výpočtem. Pokud přestane aktivita reagovat po několik sekund, je systémem násilně ukončena. Pro dlouhotrvající úkony na pozadí existuje několik možností jak je implementovat.

[4]Obdélníkový prostor nad nímž lze se lze dotázat, zda daná zeměpisná souřadnice leží uvnitř.

V LB aplikacích komunikujících přes internet je potřebné, aby aplikace neustále získávala informace od ostatních účastníků a zároveň odesílala lokální údaje. Pro tento účel by se hodilo samostatné vlákno se smyčkou, která by tuto smyčku umožňovala nejlépe i v okamžiku, kdy je aplikace např. minimalizována.

Pro takové účely existuje třída *Service*. Služba běží na pozadí aplikace nehledě na aktivity a je možné ji nechat vykonávat kód, i když je aplikace minimalizována nebo se aktivita překresluje z důvodu otočení obrazovky. Taková služba je obsažena v balíčku *Networking*. Protože využití frameworku musí být dostatečně flexibilní, je třída abstraktní. Třída v sobě zahrnuje kostru pro několik častých úkonů spojených s LB aplikacemi.

6.4.2 Způsob přenosu

Pokud aplikace zahrnuje komunikaci s jinými zařízeními, je nutné zvolit způsob přenosu dat. Pokud vynecháme, pro tento případ, poněkud exotické možnosti (ftp, e-mail atp.), tak zůstane několik nejpoužívanějších protokolů.

TCP a UDP
: Protokoly pro přímé využití pracnější. Skýtají však možnost komunikovat i s zařízeními, které neumí obsluhovat HTTP požadavky. Toto řešení bylo původně vybráno pro realizaci uzavřeného bytového protokolu z důvodu úspory přenášených dat. Od Javy 1.4 je součástí API balíček *java.nio.channel*, který obsahuje mj. třídu *SocketChannel*. Díky této třídě ve spolupráci s třídou *Selector* lze se sockety pracovat obdobně jako v jazyce C.

  Instance *SocketChannel* se otevřou a připojí na adresu a port. a zaregistrují u *Selectoru*. *Selector* následně hlídá a případně poskytuje instance *SocketChannel* pro čtení nebo zápis. Tím lze se sockety pracovat jako s bytově orientovanými streamy. Nevýhodou tohoto přístupu je komplikovanost a při snaze zmenšit objem dat se stává těžce rozšiřitelným např. v případě přidávání atributů k jedné z modelových tříd.

Google Cloud Messaging (GCM)
: Služba poskytovaná Googlem, jež možňuje odesílat data z aplikačního serveru na zařízení Android bez předchozího vyžádání. To je výhoda oproti HTTP protokolu, kde je tato funkčnost zastoupena pravidelnými dotazy na server.

GCM je zdarma a samozřejmě poskytuje více funkcí jako např. odeslání jedné zprávy na více zařízení nebo zpětnou komunikaci ze zařízení na server. Aplikační server musí implementovat *GCM Server* a aplikace zase *GCM Client*. Mezi nimi stojí *GCM Connection Servery*, které předávají zprávy (viz obrázek 6.1).

GCM rozeznává zařízení, resp. uživatele díky jejich Google účtům, které jsou nutné pro možnost např. stahování aplikací z Google Play Store[5]. GCM nebyla vybrána, protože pokud by se nějaká aplikace měla rozšiřovat i na jiné OS pro mobilní zařízení, musel by aplikační server rozeznávat z jaké platformy požadavek přišel a např. v případě Window 8 by musel využívat služby Azure[6]. Při testovací implementaci aplikačního serveru v jazyce Python navíc došlo k potížím, kdy data ze serveru odešla bez ohlášení chyby, ale na zařízení nedorazila a na Google Developers Console[7] nebyla žádná data započítána.

HTTP

Velmi rozšířený internetový protokol pro přenos dokumentů HTML i dalších souborů. Funguje systémem dotaz-odpověď. Uživatel vznese textový požadavek na konkrétní URL, přičemž součástí požadavku můžou být další parametry. Server odpovídá textovou hlavičkou obsahující informace o kódování textu, verzi HTTP protokolu, typu vraceného obsahu atp. Následuje prázdný řádek a samotný obsah, např. webová stránka (viz listing 6.2).

Typ obsahu, tzv. MIME Type určuje, zda jde o HTML stránku, pdf soubor nebo hudební záznam mp3. Lze tedy přenášet i textová data, která poskytují dostatečnou flexibilitu pro framework. HTTP protokol navíc funguje nad TCP, takže není nutné řešit pořadí, popř. výpadek datových packetů.

HTTP protokol je navíc známý a implementace serveru spočívá v implementaci webového serveru. K tomu dnes již slouží mnoho frameworků (Django, .NET MVC, Zend), které práci značně urychlí. Další výhodou je fakt, že jeden server dokáže obsloužit jakékoliv zařízení, které dokáže zasílat HTTP požadavky nehledě na platformu. Může tedy obsluhovat jak mobilní zařízení, tak běžný počítač, aniž by se musela implementace nějak lišit. Nevýhodou je, že server nemůže sám bez vyzvání zaslat data

[5]Online obchod s aplikacemi pro OS Android.

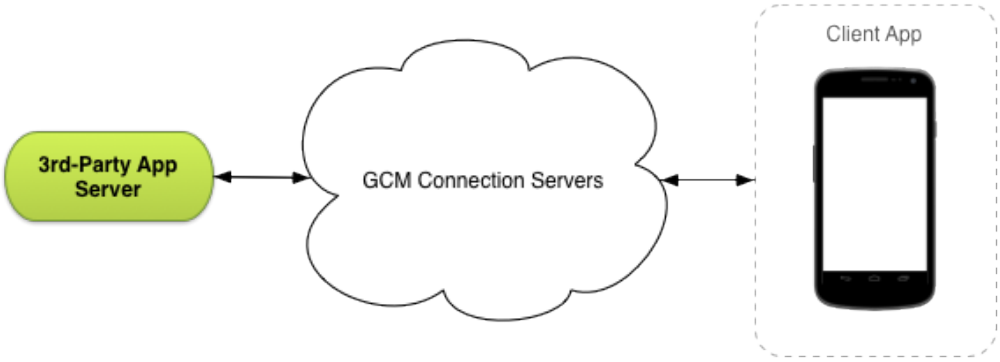
[6]Cloudová platforma společnosti Microsoft využívaná k hostování a škálování webových aplikací skrze datacentra Microsoftu.

[7]Online centrum pro vývojáře, kde mohou spravovat Googlem poskytnuté služby pro své aplikace.

na klienta, takže klient se musí pravidelně dotazovat na nová data.

Klientská implementace obnáší vytvoření dotazu, naplnění parametry a odeslání na požadovanou URL. Jedna z možností implementace je ke shlédnutí v listingu 6.1.

Tato možnost komunikace byla nakonec vybrána pro framework i přes její nedostatek. Protože zařízení by mělo pravidelně zasílat na server dotaz se svou lokací, může jako odpověď mj. dostat i zprávu, zda jsou k dispozici nějaká nová data odkud je stáhnout.



Obrázek 6.1: Schéma GCM komunikace. [15]

6.4.3 Formát přenášených dat

Při přenášení dat v textovém formátu je pro interpretaci dat mnoho možností. Já jsem vybral JSON (z ang. JavaScript Object Notation). Důvodů bylo hned několik.

- velmi rozšířený formát serializace dat
- rozšíření objektů o atributy není problém
- využívaný i službami třetích stran jako např. Google Geolocation API
- díky rozšířenosti existují již hotová řešení pro serializaci a deserializaci objektů
- oproti jiným formátům jako např. XML, je objemově úsporný

JSON dokáže reprezentovat (viz listing 6.3):

1. Řetězce
2. Čísla
   (a) Celá
   (b) S plovoucí desetinnou čárkou
3. Speciální hodnoty
   (a) True nebo False
   (b) Null
4. Neindexované pole hodnot
5. Objekty, resp. pole, indexované řetězcovými klíči

Pravidla lze kombinovat, takže JSON umí přenášet i pole objektů. Při práci s řetězci lze využít escapování pro znaky jako konec řádku(\n), tabulátor(\t) nebo znaky UTF-16 zapsané hexadecimálně (\uXXXX).

6.4.3.a Reakce na zprávy

Framework počítá s dvěma druhy zpráv. První typ je vlastně odpověď na HTTP požadavek na server. Odpověď je zpracována v abstraktní metodě *processResponse(HttpResponse response, String url)*, která je volána uvnitř metody *sendDataToServer(String data, String serverUrl)*. Konkrétní implementace je tedy na programátorovi. Parametr *url* je vhodné použít k rozlišení, jak má být s odpovědí naloženo. Tím se nabízí implementace v podobě stromu podmínek, kdy klíčem bude právě adresa, odkud odpověď přišla.

Druhý případ je reakce na zprávy od aplikace. Ty mohou přijít od uživatele skrze uživatelské rozhraní nebo od aplikace jako takové např. od vlákna, které bude pravidelně vykonávat nějakou činnost.

V této fázi bylo nutné zvolit vhodný způsob doručování zpráv do služby. Jedním ze způsobů je ponechat si referenci na službu, díky čemuž k ní lze přistupovat přímo. Tento způsob je ovšem nepohodlný pro aplikace, kde bude využito mnoho aktivit. Mnou zvolený způsob je robustnější. Zakládá se na zasílání instancí třídy *Intent*. Instance *Itent* v sobě obsahuje mj.

Listing 6.1: Implementace HTTP požadavku pro OS Android.

```
//nastaveni hlavicky pro odesilani metodou POST
HttpClient httpclient = new DefaultHttpClient();
HttpPost httpPost = new HttpPost("serverUrl");
List<NameValuePair> params = new ArrayList<NameValuePair>(1);
params.add(new BasicNameValuePair("data", data));
try {
        //pridani parametru do pozadavku
        httpPost.setEntity(new UrlEncodedFormEntity(params));
        //zachyceni reakce na pozadavek
        HttpResponse httpResponse = httpClient.execute(httpPost);
        //zpracovani odpovedi
        processResponse(httpResponse);
} catch (Exception e) {
        osetreni vyjimky
        handleError(e);
}
```

Listing 6.2: Příklad HTTP komunikace skrze telnet.

```
telnet w3c.org 80 -4
Trying 128.30.52.45...
Connected to w3c.org.
Escape character is '^]'.
GET / HTTP/1.1

HTTP/1.0 403 Forbidden
Cache-Control: no-cache
Connection: close
Content-Type: text/html

<html><body><h1>403 Forbidden</h1>
Request forbidden by administrative rules.
</body></html>
```

Listing 6.3: Ukázka dat ve formátu JSON.

```
# neindexovane pole
[1, 2.5, true, "text"]

# indexovane pole resp. instance objektu
{
    "var1":1,
    "var2":2.5,
    "var3":true,
    "var4":"text"
}

# neindexovane pole objektu
[
        {
        "var1":1,
        "var2":2.5,
        "var3":true,
        "var4":"text"
        },
    {
        "var1":1,
        "var2":2.5,
        "var3":true,
        "var4":"text"
        }
]
```

- typ akce reprezentovaný řetězcem
- slovník proměnných s řetězcovými klíči

Tyto instance lze pak skrze systém zasílat broadcastem[8]. Služba nebo aktivita, která implementuje instanci *BroadcastReceiver*, tj. jeho metody *onReceive(Context context, Intent intent)*, může na zaslané instance *Intent* reagovat nebo je nechat být. Instanci třídy *BroadcastReceiver* je nutné ještě registrovat u instance *LocalBroadcastManager* spolu s instancí *IntentFilter*, která obsahuje seznam akcí pro odebírání

6.4.3.b Odesílání zpráv

Zprávy mohou být na server odesílány skrze HTTP požadavek nebo skrze *Intent* do jiné aktivity, resp. služby. OS Android využívá *Intenty* i pro spouštění aplikací nebo např. otevírání internetových stránek. Díky komunikaci přes *Intenty* může být framework součástí tohoto toku informací systémem a reagovat např. i na nízký stav baterie apod.

6.4.3.c Aktualizace lokace přístroje

Součástí služby na pozadí je i pravidelné získávání nové lokace přístroje. Pro tento případ byla zvolena jiná varianta než v listingu 5.3, resp. 5.4. Knihovna Google Play Services nabízí zjednodušení pro odběr aktuální polohy i nastavení způsobu odběru. Proces je jednoduchý, ale na slovní popis zdlouhavý, proto je názornější ukázka v listingu 6.4.

Zjednodušené nastavení spočívá ve volbě druhu odběru lokace metodou *setPriority(int type)* prostřednictvím konstant z třídy *LocationRequest* [17] (viz tabulka 6.1). K tomu je třeba nastavit ještě intervaly pravidelného aktivního odběru metodou *setInterval(int interval)* a pasivního odběru *setFastestInterval(int interval)*. Aktivní odběr znamená, že je lokace vyžádána. Pasivní funguje tak, že lokace jsou obdrženy od jiných běžících aplikací, které si aktivně polohu vyžádaly.

Protože služba sama implementuje rozhraní *LocationListener*, může na změnu polohy ihned reagovat sama nebo zasláním zprávy některé z aktivit.

[8]Hromadná zpráva pro všechny.

Pro použití ve službě stačí implementovat abstraktní metodu *setupPropertiesOfLocationRequest()* a v ní provést nastavení instance *LocationRequestu*. Spojení již služba obstará sama.

Tabulka 6.1: Možnosti odběru polohy *LocationClientem*.

Konstanta	*Popis*
PRIORITY_BALANCED_POWER_ACCURACY	Časté kompromisní nastavení. Dlouhé intervaly aktivních požadavků jsou prokládány pasivním získáváním polohy.
PRIORITY_HIGH_ACCURACY	Požadavek nejvyšší možné přesnosti a pravidelného odběru.
PRIORITY_LOW_POWER	Stačí nižší přesnost, což šetří baterii. Např. pokud je zapojená GPS, tak je využita pouze síť k určení polohy.
PRIORITY_NO_POWER	Slouží k nastavení pouze pasivního příjmu polohy.

6.5 Integrace frameworku

Celý framework je zabalený jako **.jar* soubor a stačí jej k projektu pouze připojit. V prostředí *Eclipse* je postup následující:

1. vytvořit v projektu adresář *libs*, pokud již neexistuje
2. nakopírovat do adresáře knihovnu frameworku
3. přidat knihovnu do *Build path* projektu přes *Project -> Properties -> Java Build Path -> Libraries -> Add JARs*

Listing 6.4: Přihlášení k odběru polohy s využitím Google Play Services [18].

```
public class MainActivity extends Activity implements
    ConnectionCallbacks,OnConnectionFailedListener,LocationListener
    {

private LocationClient lc;
private LocationRequest;

// metoda zajistujici iniciaci aktivity
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // protoze trida implementuje rozhrani ConnectionCallbacks a
    // OnConnectionFailedListener lze vyuzit jako argumenty this
    lc = new LocationClient(this, this, this);
    // pripojeni clienta pro odber sluzby
    lc.connect();
    // vytvoreni prazdneho pozadavku
    lr = LocationRequest.create();
    // nastaveni zpusobu ziskavani lokace
    lr.setPriority(LocationRequest.PRIORITY_BALANCED_POWER_ACCURACY);
    // interval pro vyzadani lokace aktivne
    lr.setInterval(100000);
    // interval pro obdrzeni lokace od jinych aplikaci pasivne
    lr.setFastestInterval(10000);
}

// volana ve chvili, kdy je LocationClient pripojen
// pozadavek je vhodne umistit zde, protoze metoda lc.connect()
// je asynchronni
@Override
public void onConnected(Bundle dataBundle) {
       lc.requestLocationUpdates(lr, this);
}

// volana pri notifikaci o zmene polohy
@Override
     public void onLocationChanged(Location location) {...}

...
```

7 Aplikace

Pro ověření použitelnosti frameworku je vhodná implementace aplikace, která jej využívá. Zvolil jsem aplikaci pro sdílení pozice a vizualizaci účastníků na mapě.

Uživatel může vytvořit skupinu a její ID a heslo předat jiným uživatelům, kteří se mohou do skupiny přihlásit. Uživatelé v rámci skupiny se vzájemně vidí na mapě. Uživatel může být přihlášen ve více mapách a mezi skupinami může vybírat.

Aplikace ovšem slouží k ověření funkčnosti a nalezení případných možností vylepšení. K jejímu detailnímu pochopení může čtenář nahlédnout do přiložených zdrojových kódů.

7.1 Datový model

Model (viz Obrázek 7.3) sestává ze tří tabulek. Skutečnost, že uživatel (*User*) může být přihlášen ve více skupinách (*Groups*), vede na vazbu m:n, která je rozložena v tabulce *GroupUser*.

Na zařízení však databázi využívat nebudeme, aby bylo možné ukázat možnosti frameworku. Proto jsou v aplikační části data reprezentována za pomoci modelových tříd (viz Obrázek 7.4).

7.2 Použití frameworku

Aplikace využívá kromě modelových tříd i implementaci abstraktní třídy *NetworkService - BaseService*. Implementací devíti abstraktních metod se z této služby stává komunikační centrum aplikace.

getBinder vrací objekt *IBinder*. Ten je vrácen při spuštění služby z aktivity jako vázané. *IBinder* pak zpřístupňuje veřejné metody a proměnné.

buildTrayNotification je obalovací metodou volanou již při vytváření služby. Jejím účelem je zajistit, aby služba měla neustálý záznam panelu

notifikací a tím bylo zřejmé, že aplikace běží.

setupPropertiesOfLocationRequest je také obalovací metodou vybízející k inicializaci a nastavení instancí *LocationClient* a *LocationRequest* pro pravidelné získávání změn polohy.

processResponse je metoda volaná pro zpracování odpovědi na HTTP požadavek. Uvnitř metody je vhodné rozlišit zpracování odpovědi na základě URL nebo akce, ze které odpověď přišla.

handleSendDataError je využita v případě, že dojde k vyvolání výjimky při odesílání HTTP požadavku. V případě *BaseService* dochází k odeslání broadcastové zprávy o chybě při odesílání a reakce je ponechána na odběratelích.

handleSendWorkerThreadError zajišťuje reakci na chybu ve vlákně, které obstarává spooling intentů odesílaných na server. *BaseService* tuto událost implementuje opětovným spuštěním vlákna.

setupRepeater je obalovací metoda k inicializaci a nastavení *TimerTask* a *Timer*. To lze využít pro pravidelné úkony jako v případě *BaseService*, kde je odesílána vlastní pozice na server. Spustit či zastavit tyto opakované akce lze zasláním broadcastové zprávy s příslušným identifikátorem.

customActions je důležitou metodou. Parametrem je zpráva zaslaná broadcastem a tím lze uvnitř metody vytvořit rozhodovací strom na základě typu akce ve zprávě.

Potomky *NetworkService* lze jako jiné služby spouštět z aktivity metodou *startService(Intent i)*, kde je nutné vložit jako data seznam řetězcových identifikátorů pod klíčem *NSI_REGISTER_ACTIONS*, na které bude služba reagovat. Pokud nebude identifikátor v seznamu, budou takto označené broadcastové zprávy službou ignorovány.

Při vlastní implementaci třídy *NetworkService* je vhodné si nejdříve prohlédnout tuto abstraktní třídu, protože již obsahuje několik konstant a proměnných, resp. atributů, které stačí inicializovat jako v případě *TimerTask, Timer, LocationClient* atp.

Pro zobrazování mapy je využit *MixedTileProvider*, který umožňuje ukládání mapových dlaždic do úložiště.

Aplikace slouží k demonstraci funkčnosti frameworku, proto bylo vynecháno šifrování dat. Při reálném nasazení by bylo vhodné zabezpečit přenášená data skrze spojení https nebo šifrováním JSON dat za pomoci AES knihovny obsažené ve frameworku.

7.3 Komunikace s frameworkem

Každá aktivita nebo služba, která chce s frameworkem, resp. se službou *BaseService*, komunikovat, musí implementovat vlastní *BroadcastReceiver*, ve kterém bude reagovat na příchozí zprávy. Tento pak musí zaregistrovat pomocí *LocalBroadcastManageru*, pomocí kterého musí i zasílat broadcastové zprávy službě.

Je patrné, že v aplikaci se vyskytuje mnoho řetězcových konstant jak pro názvy akcí *Intentů*, tak pro klíče k proměnným obsaženým uvnitř *Intentů*. Všechny tyto konstanty jsou umístěné ve třídě *ConnConsts*.

7.4 Uživatelská dokumentace

Při prvním spuštění uživatel v sekci *Settings* vyplní URL serveru a připojí se (viz obr. 7.2). Tím se verifikuje, zda je server k dispozici. Pokud ano, uloží se adresa k dalšímu využívání a uložena již zůstane perzistentně.

V sekci *Manage User* může uživatel vytvořit nového uživatele či změnit svůj stávající komentář. Ovšem je nutné při každém spuštění aplikace, aby se uživatel přihlásil. Tím se nejen ověří jeho údaje, ale také stáhnou skupiny, do kterých je přihlášen.

V záložce *Groups* může spravovat skupiny, v nichž je přihlášen, nebo vytvořit novou. V seznamu skupin může volit a tím filtrovat uživatele zobrazené na mapě v záložce *Map* (viz obr. 7.1).

7.5 Instalace aplikace

V nastavení zařízení s OS Android je nutné povolit instalaci z neznámých zdrojů. Pak lze aplikaci instalovat kliknutím na odkaz s **.apk* souborem.

Další variantou využívanou i při testování je připojení telefonu k PC a následné spuštění projektu v prostředí *Eclipse* nebo *Android Studiu*. V tomto případě je ovšem nutné mít nainstalovaný potřebný USB driver, aby PC bylo schopné zařízení detekovat a komunikovat s ním.

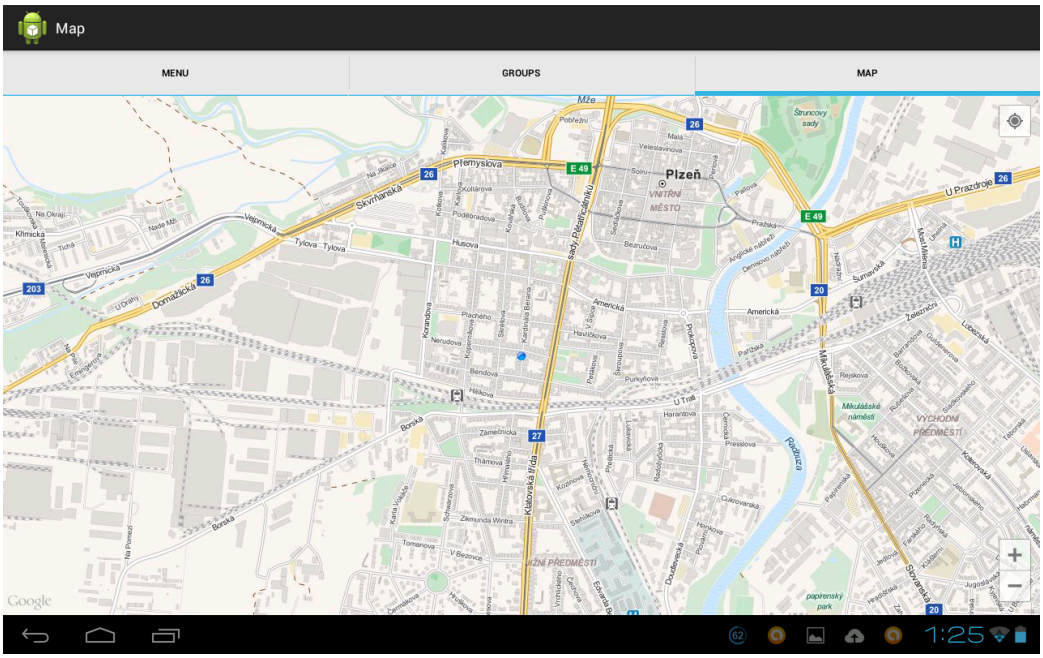
7.6 Testování

Testování aplikace probíhalo na domácí Wi-Fi síti. Jako server sloužilo PC s OS GNU/Linux (Elementary OS Luna), jako klientská zařízení tablet Ainol Novo Flame (Android ICS) a Samsung Galaxy Y (Android Custom ROM odpovídající verzi 4.4).

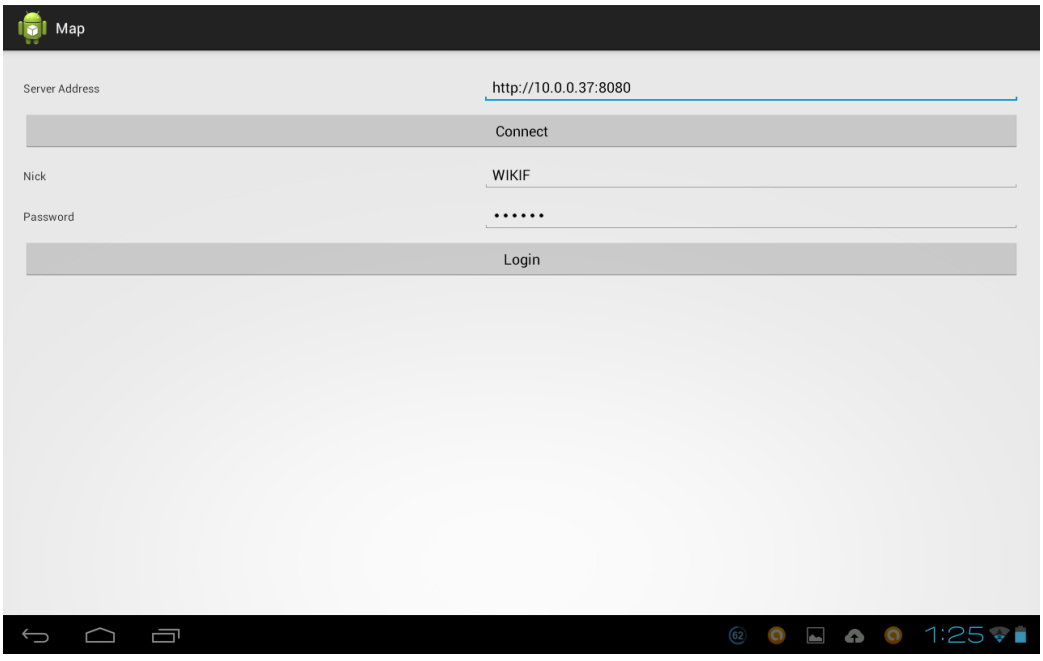
Server byl nejdříve testován zasíláním ručně vytvořených POST requestů, teprve následně byl testován s fyzickými zařízeními. Kontrola spočívala ve vyhodnocení JSON odpovědí a zároveň kontrole tabulek v databázi pomocí příkazů *select*.

Aby bylo možné přístroje pohodlně testovat, byla zvolena varianta podvržených pozic. Telefonům byly nastaveny falešné pozice a bylo ověřeno, že se hráči na mapě uvidí. Další fází testování byl pohyb zařízení kolem domu na stejném serveru a Wi-Fi. Jako zdroj pozic bylo využito GPS.

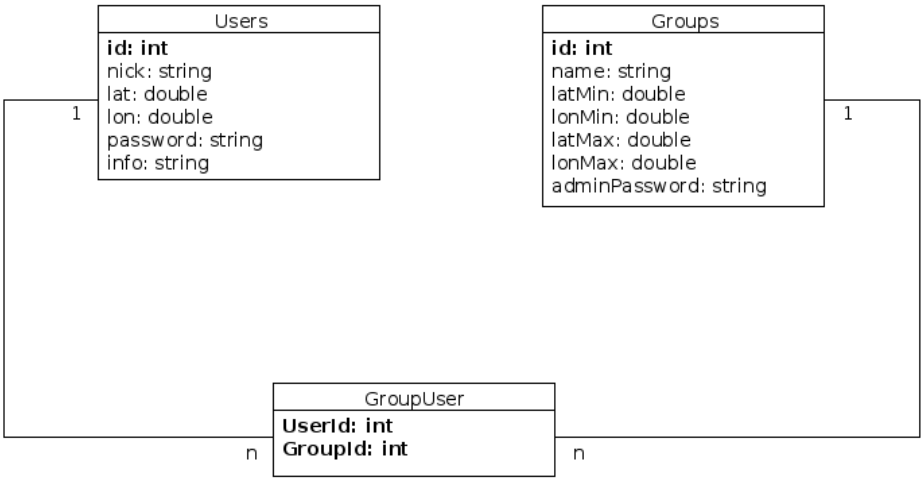
Poslední fází bylo testování reakcí na záměrně vytvořené chybové situace jako je vypadnutí serveru či ruční smazání uložených mapových podkladů.



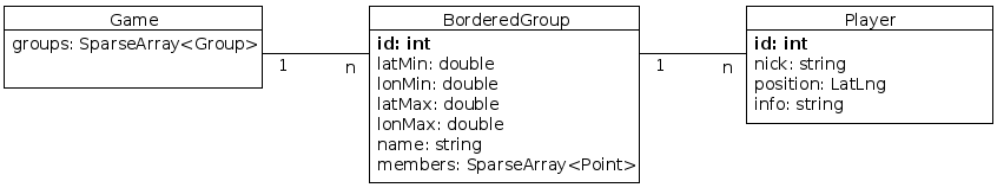
Obrázek 7.1: Ukázka vizualizace mapy s podklady OpenStreetMaps.



Obrázek 7.2: Aktivita pro ověření spojení se serverem a přihlášení.



Obrázek 7.3: Diagram využívaných entit.



Obrázek 7.4: Diagram entit v mobilní aplikaci.

8 Server

Druhá část aplikace je serverová část. Pro ukázkovou aplikaci bylo několik požadavků na server:

- známý programovací jazyk
- jednoduchá implementace
- přenositelnost
- jednoduchá instalace

Tyto požadavky splňuje framework *web.py* [24]. Jak koncovka napovídá, jedná se o jazyk Python [3]. Celý server byl implementován a testován na operačním systému GNU/Linux, konkrétně Elementary OS verze Luna. Proto je i veškerá instalace popsána pro tento OS.

8.1 Struktura serveru

Veškeré soubory jsou obsaženy v adresáři src/WebServer (viz příloha A).

- create.sql - skript pro vytvoření tabulek
- server.py - samotné jádro serveru
- userLib.py - knihovna pro správu uživatelů
- groupLib.py - knihovna pro správu skupin
- groupUserLib.py - knihovna pro přihlášení a odhlášení ze skupiny
- aesCrypto.py - knihovna pro šifrování a dešifrování textu metodou AES

8.1.1 Databáze

Jako databáze byla z následujících důvodů zvolena Sqlite3.

- je obsažena v řadě distribucí GNU/Linux již v základu
- není třeba řešit přístupová práva
- veškerá data v jediném přenosném souboru

Databáze odpovídá diagramu 7.3. Umožňuje tedy udržovat uživatele přihlášené do více skupin. V databázi je nastavené kaskádové mazání, pokud je tedy smazána skupina nebo uživatel, jsou související záznamy odstraněny z rozkladové tabulky.

Úskalím Sqlite3 je fakt, že po spuštění nepracuje s cizími klíči. Toho lze docílit příkazem *PRAGMA foreign_keys=ON;*. Ovšem pokud je příkaz odesílán ze serveru, je odeslán jako samostatný požadavek. Proto bylo nutné přístup pozměnit a dotazy na databázi provádět uvnitř transakce, když je nejdříve zaslán tento příkaz a teprve následně samotný dotaz.

8.1.2 Instalace databáze

Pokud distribuce neobsahuje Sqlite3, lze ji nainstalovat přes manager, jako je *Synaptic* nebo například příkazem *sudo apt-get install sqlite3*. K vytvoření tabulek v databázi slouží skript create.sql ve složce src/WebServer. Příkazem *sqlite3 gps.db < create.sql* se v aktuálním adresáři vytvoří soubor gps.db, který bude obsahovat prázdnou databázi.

8.1.3 Jádro serveru

Jádro je tvořeno souborem server.py. Využívá framework *web.py* a při příchozím POST požadavku předává JSON z požadavku, ID uživatele a kontext databáze příslušné metodě z přítomných knihoven. Ta vykoná práci nad databází a vrací JSON odpověď, která je odeslána jako odpověď na HTTP požadavek.

8.1.4 Instalace a spuštění

Pro spuštění serveru je nutná instalace frameworku *web.py*. To je dle mého názoru nejjednodušší skrze program *easy_install* obsažený v balíku *python-setuptool*. Nejdříve je tedy nutné instalovat tento balík příkazem *sudo apt-get install python-setuptools* a následně instalovat framework *sudo easy_install web.py*.

Server se spouští přes příkazovou řádku *python server.py 0.0.0.0:8080*. IP adresa by měla být samozřejmě nahrazena veřejnou IP adresou stroje.

Server posílá na výstup vyškeré chybové zprávy, dotazy do databáze či záznamy o příchozích HTTP dotazech a kódy HTTP odchozích odpovědí.

9 Závěr

Cílem práce bylo vytvoření knihovny zjednodušující implementaci location-based aplikací pro platformu Android. K tomu bylo nezbytné analyzovat a porovnat možnosti lokalizace mobilních přístrojů.

Po prvotní analýze lokalizačních metod bylo zjištěno, jakým způsobem jsou tyto metody zpřístupněny na platformě Android. Zde je nutné poznamenat, že API Google Map se v průběhu práce změnilo a bylo nutné analýzu provést znovu.

Na základě znalostí z předchozích částí práce bylo ještě nutné nalézt společný průnik vlastností často vyžadovaných u location-based aplikací. Následovala implementace samotného frameworku, kdy bylo nezbytné vyřešit několik problémů jako např. dostatečně komplexní předávání zpráv uvnitř aplikace. Tyto problémy se ukázaly jako zásadní, protože framework využívá několik vláken, která nemohou přímo měnit uživatelské rozhraní. Naopak hlavní vlákno aplikace nemůže v novějších verzích Androidu např. odesílat přímo http požadavek.

Poslední částí byl návrh a implementace ukázkové aplikace a potřebného webového serveru. Protože se jedná o demonstrační aplikaci, je navržena dosti genericky a pro další využití by bylo vhodné ji ještě modifikovat.

Framework vývojáři usnadňuje časté úkony, jako ukládání mapových podkladů, komunikace se serverem, šifrování dat, omezení prostoru mapy či asynchronní zpracování zpráv. Skrze abstraktní službu vývojáře vede vývojem nové aplikace a pomáhá mu neztratit se v, dle mě dosti složitém, systému komunikace vláken. Další možné vylepšení vidím v analýze a případné implementaci určování polohy pouze na základě dat z integrovaného akcelerometru.

Seznam použitých zkratek

API	Aplication Programming Interface
apod.	a podobně
atd.	a tak dále
atp.	a tak podobně
BTS	Base Transceiver Station vysílač/přijímač
CELL ID	unikátní identifikátor BTS
GCM	Google Cloud Messaging
GPS	Global Positioning System
GSM	Groupe Spécial Mobile
JSON	JavaScript Object Notation
LB	Location-based
mj.	mimo jiné
MT	mobilní telefon
např.	například
obr.	obrázek
OO	objektově orientovaný
OS	operační systém
OSM	OpenStreetMaps
POI	Point Of Interest
popř.	popřípadě
TA	Timing Advance
tj.	to jest
TS	Time Slot
tzv.	takzvaný
UTM	Universal Transverse Mercator
V1	Android Google Maps API verze 1
V2	Android Google Maps API verze 2
2D	dvourozměrný
3D	trojrozměrný

Zdroje

[1] Mark L. Murphy: *Android 2 Průvodce programováním mobilních aplikací*
Computer Press, a. s., 2011
ISBN 978-80-251-3194-7

[2] Grant Allen: *Android 4 Průvodce programováním mobilních aplikací*
Computer Press, a. s., 2013
ISBN 978-80-251-3782-6

[3] Daryl Harms, Kenneth McDonald: *Začínáme programovat v jazyce Python*
Computer Press, a. s., 2008
ISBN 978-80-251-2161-0

[4] *Válcová zobrazení jednoduchá* [on-line] [cit. 10. 2. 2014]
`http://gis.zcu.cz/studium/mk2/multimedialni_texty/index_soubory/hlavni_soubory/jednoduch_soubory/valec.html`

[5] *Using geographic coordinates* [on-line] [cit. 11. 2. 2014]
`https://support.google.com/earth/answer/148106?hl=en`

[6] *Stackoverflow - Maximum Lat and Long bounds for the world - Google Maps API LatLngBounds()* [on-line] [cit. 11. 2. 2014]
`http://stackoverflow.com/questions/11849636/maximum-lat-and-long-bounds-for-the-world-google-maps-api-latlngbounds/13824556#13824556`

[7] Dr. Ondřej Franěk: *Lokalizace volajícího při tísňovém volání z mobilního telefonu* [on-line] [cit. 15. 2. 2014]
`http://www.zachrannasluzba.cz/odborna/0306_lokmt.htm`, 2003

[8] Tomáš Doseděl: *Není jenom GPS: přehled navigačních systémů* [on-line] [cit. 22. 2. 2014]

http://www.mobinfo.cz/neni-jenom-gps-prehled-navigacnich-systemu/, 2012

[9] *GPS - Space Segment* [on-line] [cit. 1. 3. 2014]
http://www.gps.gov/systems/gps/space/, 2014

[10] *GLONASS - Guide* [on-line] [cit. 1. 3. 2014]
http://glonass-iac.ru/en/guide/navfaq.php

[11] *Google Geolocation API* [on-line] [cit. 5. 3. 2014]
https://developers.google.com/maps/documentation/business/geolocation/

[12] *Navizon - Wi-Fi and Cell-ID location* [on-line] [cit. 5. 3. 2014]
http://www.navizon.com/wifi-cell-tower-location-database

[13] *Unwired Labs - Location API* [on-line] [cit. 5. 3. 2014]
http://unwiredlabs.com/

[14] *Google Play Services* [on-line] [cit. 10. 3. 2014]
http://developer.android.com/google/play-services/index.html

[15] *Google Cloud Messaging for Android* [on-line] [cit. 12. 3. 2014]
http://developer.android.com/google/gcm/gcm.html

[16] *Android Developer Reference - LocationManager* [on-line] [cit. 14. 3. 2014]
http://developer.android.com/reference/android/location/LocationManager.html

[17] *Android Developer Reference - LocationRequest* [on-line] [cit. 14. 3. 2014]
http://developer.android.com/reference/com/google/android/gms/location/LocationRequest.html

[18] *Android Developer Reference - Receive Location Updates* [on-line] [cit. 14. 3. 2014]
http://developer.android.com/training/location/receive-location-updates.html

[19] *Android Developer Reference - TileOverlay* [on-line] [cit. 14. 3. 2014]
http://developer.android.com/reference/com/google/android/gms/maps/model/TileOverlay.html

[20] *An Open-Source JavaScript Library for Mobile-Friendly Interactive Maps* [on-line] [cit. 5. 3. 2014]
`http://leafletjs.com/`

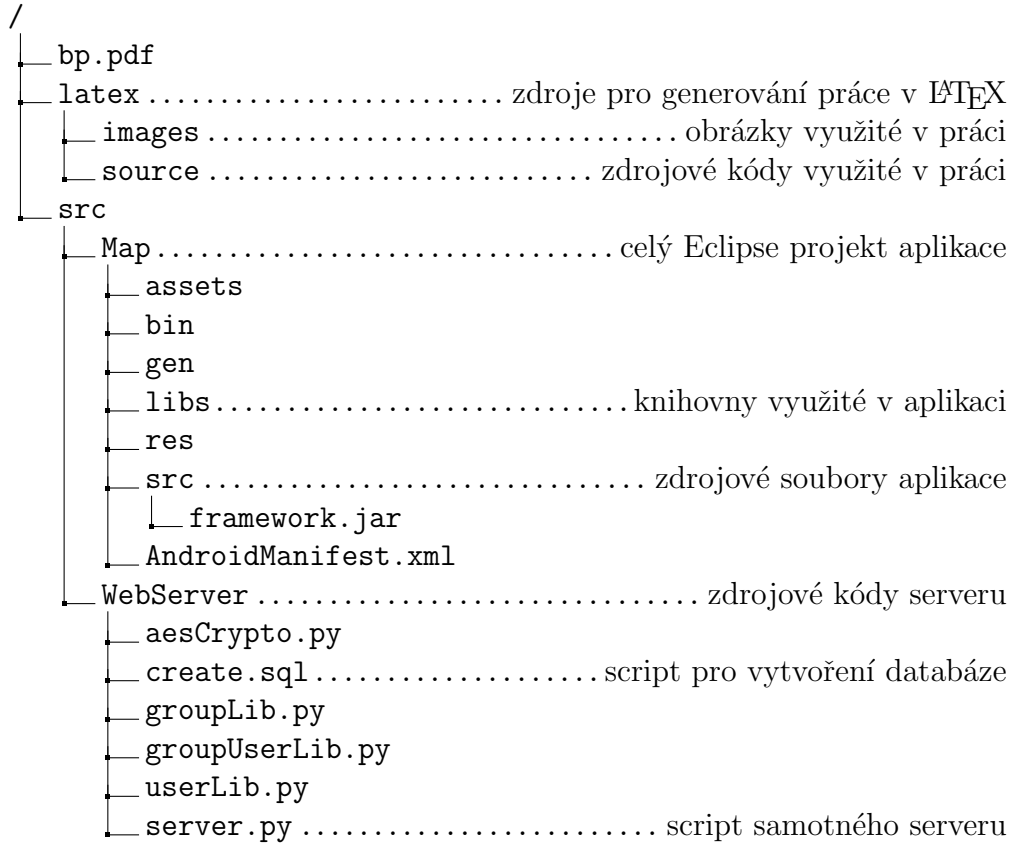
[21] *OpenStreetMap - About* [on-line] [cit. 20. 3. 2014]
`http://www.openstreetmap.org/about`

[22] *OpenCellId* [on-line] [cit. 6. 3. 2014]
`http://opencellid.org/#&action=statistics.measurements&sortByRadio=2&dateFrom=&dateTo=&mcc=&mnc=`

[23] *MapQuest - Open APIs* [on-line] [cit. 2. 4. 2014]
`http://developer.mapquest.com/web/products/open`

[24] *Web.py Framework - Tutorial* [on-line] [cit. 10. 4. 2014]
`http://webpy.org/docs/0.3/tutorial`

A Adresářová struktura práce



Adresářová struktura přiloženého CD. Méně významné soubory, jako konfigurační soubory Eclipse projektu nebo generované soubory nutné pro spouštění Android projektu na telefonu, jsou vynechány.