VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2016 Bc. Petr Šťastný

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

APLIKACE PRO INTELIGENTNÍ LOGOVÁNÍ POLOHY Z GPS PRO ZAŘÍZENÍ S OS ANDROID
APPLICATION FOR SMART LOGGING OF THE GPS POSITION FOR ANDROID OS DEVICE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Petr Štastný

VEDOUCÍ PRÁCE
SUPERVISOR
doc. Ing. Ivo Lattenberg, Ph.D.

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Petr Šťastný ***ID:*** 134633
Ročník: 2 ***Akademický rok:*** 2015/16

NÁZEV TÉMATU:

Aplikace pro inteligentní logování polohy z GPS pro zařízení s OS Android

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte problematiku měření polohy pomocí GPS. Navrhněte a naprogramujte službu pro OS Android, která bude sloužit k logování aktuální polohy zjištěné z GPS modulu. Ukládejte krom polohy také informace o rychlosti, výšce, přesnosti polohy a počtu dostupných družic. Vytvořte také obslužný program, který bude se službou komunikovat a bude umožňovat logování spustit či zastavit, bude zde možno nastavit parametry jako např. četnost logování, prahové parametry pro potlačení logování, když se zařízení nepohybuje apod. Další možností bude export do souboru formátu GPX nebo KML. Vytvořte dále jednoduchou aplikaci pro PC, která na mapovém podkladu zobrazí průběh polohy zařízení na podkladové on-line mapě.

DOPORUČENÁ LITERATURA:

[1] GRANT, A., Android 4 Průvodce programováním mobilních aplikací, Nakladatelství Computer Press, a.s. 2013, 656 stran, ISBN 9788025137826

[2] WATSON, B., C# 4.0 - řešení praktických programátorských úloh, Zoner Press, 2010, 656 s., ISBN 978-8--7413-094-6

[3] VIRIUS, M., C# 2010 Hotová řešení, Computer Press, 2012, 424 s., ISBN 978-80-251-3730-7

Termín zadání: 1.2.2016 ***Termín odevzdání:*** 25.5.2016

Vedoucí práce: doc. Ing. Ivo Lattenberg, Ph.D.
Konzultant diplomové práce:

doc. Ing. Jiří Mišurec, CSc., *předseda oborové rady*

ABSTRAKT

Cílem této práce je vytvořit samostatnou službu, která by zjišťovala, ukládala a přeposílala informace o poloze uživatele s operačním systémem Android. Informace by měli být přeposílány on-line. Ukládání polohy by mělo probíhat přes KML soubor, který by měla aplikace generovat.

KLÍČOVÁ SLOVA

GPS, Android, aplikace, lokace, umělá družice, zemský souřadnicový systém, Material Design, služba, KML, GoogleMap, GMapFX

ABSTRACT

The goal of this thesis is create standalone service whitch should look for location, save history of locations and send actual location of user using Android OS device. Location should be send on-line. Saving is made by KML file witch this aplication generates.

KEYWORDS

GPS, Android, application, location, satellite, geographic coordinates, Material Design, service, KML, GoogleMap, GMapFX

ŠŤASTNÝ, Petr *Aplikace pro inteligentní logování polohy z GPS pro zařízení s OS Android*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2016. 57 s. Vedoucí práce byl doc. Ing. Ivo Lattenberg, PhD.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Aplikace pro inteligentní logování polohy z GPS pro zařízení s OS Android“ jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu doc. Ing. Ivo Lattenbergovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .

podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

Úvod **11**

1 Globální poziční systém (GPS) **12**
1.1 Historie GPS . . . 12
1.2 Funkčnost systému . . . 13
1.2.1 Kosmický segment . . . 13
1.2.2 Řídící segment . . . 14
1.2.3 Uživatelský segment . . . 14
1.2.4 Známí rivalové [2] . . . 15
1.3 Souřadnicový systém . . . 16
1.3.1 GPX a KML soubory . . . 16

2 Operační systém Android **18**
2.1 Uživatelské prostředí . . . 18
2.1.1 Aplikace . . . 18
2.2 Vývoj Androidu . . . 19
2.2.1 Linux . . . 19
2.2.2 Organizace paměti . . . 19
2.2.3 Dostupnost aktualizací . . . 20
2.3 Vývoj aplikací . . . 20
2.3.1 Složení aplikací . . . 21
2.3.2 Vlastnosti a nástroje systému . . . 23

3 Návrh a vývoj testovací aplikace **25**
3.1 Přístupová práva . . . 25
3.2 Location Strategies . . . 25
3.2.1 Test aplikace . . . 30

4 Lokalizace jako služba **31**
4.1 Koloběh života služby v OS Android . . . 31
4.2 Komunikace se službou . . . 33
4.3 Komunikace s okolním světem . . . 34

5 Aplikace GPS Finder **36**
5.1 Uživatelské prostředí . . . 36
5.1.1 Material Design . . . 36
5.1.2 Fragment . . . 38
5.2 Přístupová práva . . . 42

5.3 Generování GPX a KML souborů 43

6 Aplikace GPS Finder na PC **46**

6.1 Uživatelské prostředí 46

6.1.1 GMapFX 46

6.2 Komunikace s mobilním telefonem 48

6.3 Zobrazení KML souborů 48

7 Testování **50**

8 Závěr **52**

Literatura **54**

Seznam symbolů, veličin a zkratek **55**

Seznam příloh **56**

A Zdrojové kódy aplikací **57**

SEZNAM OBRÁZKŮ

3.1 Inicializace, hodnoty 0 27
3.2 Inicializace, poslední lokace 27
4.1 Rozdíl mezi nevázanou (vlevo) a vázanou (vpravo) službou[7] 31
5.1 NavigationDrawer aplikace GPS Finder 37
5.2 Obrazovka Map s aktuální polohou zařízení a polohou druhého zařízení 39
5.3 Obrazovka Settings 40
5.4 Obrazovka Current Location 41
5.5 Obrazovka About 42
6.1 Zobrazení mapy po inicializaci a zobrazení protější strany markerem . 46
6.2 Zobrazení KML souboru v mapě pomocí polygonu 49

ÚVOD

Svět mobilních telefonů nabízí mnoho možností jejich využití. Jedním z těchto způsobů je využití GPS komunikace k lokalizaci, ať už chtěné či nechtěné, vypočítávání rozdílů vzdáleností mezi dvěma zařízeními a podobně. Cílem této práce je vytvořit samostatnou službu, která bude schopna zjisťovat polohu zařízení na pozadí mobilního telefonu s operačním systémem Android. Tato služba by měla být schopna vysílat zjištěná data do Internetu a tyto data dále zpracovávat, ať už na telefonu nebo počítači. K tomu je uvažováno vytvořit samostatnou aplikaci, která bude zajišťovat chod služby na pozadí telefonu a bude nabízet přispůsobení této služby k obrazu uživatele.

V první kapitole bude představen GPS systém, jeho historie a také způsob komunikace, který využívá. Bude také srovnán s ostatními systémy dostupnými civilnímu obyvatelstvu. Druhá kapitola se týká operačního systému Android. Ten je v současné době nejvíce rozšířeným operačním systémem na mobilních platformách. Bude představen jeho základ a funkčnost, tak jako způsob vytváření softwaru třetích stran. Další kapitola se zaměřuje na návrh a řešení komunikace s třídou Location. Bude představeno řešení testovací aplikace, závěry z tohoto testu a v neposlední řadě porovnání s tímto problémem jako službou.

Následující kapitola se bude zabývat službou jako takovou. Budou zde popsány všechny body, které jsou součástí vzniklé třídy, jako je komunikace s aplikací či komunikací do Internetu. Další dvě kapitoly se budou zabývat uživatelským prostředím aplikace na telefonu a počítači. Bude také nastíněn způsob vývoje, co bylo použito a jak by se stávající stav mohl vylepšit. Poslední kapitola se bude již jen zabývat testováním celého projektu.

1 GLOBÁLNÍ POZIČNÍ SYSTÉM (GPS)

[1]

GPS (v anglickém originále Global Positioning System) je systém lokalize na zemském povrchu pomocí satelitů. Tento systém byl vyvinut ve spolupráci s armádou spojených států, pro lokalizaci svých jednotek na bojištích. Dnes je již GPS dostupná každému člověku, který má u sebe GPS lokátor nebo jakýkoli jiný přístroj, který tento lokátor obsahuje. V současnosti se jedná o jediný, plně funkční satelitní navigační systém.

Vývoj systému organizovalo Ministerstvo obrany Spojených Států Amerických (United States Department of Defense) a prvně byl tento systém spuštěn v roce 1978. Jeho oficiální název je NAVSTAR GPS (NAVigation Signal Timing And Ranging Global Positioning System). Družicová konstelace je udržována a řízena padesátým „vesmírným“ oddílem vzdušných sil Spojených Států Amerických (50th Space Wing of the United States Air Force). Přesto, že údržba systému stojí ročně přibližně 400 milionů dolarů, je GPS pro civilní užívání zcela zdarma. Po technologické a vědecké stránce je GPS pravděpodobně nejdůležitějším praktickým důkazem Einsteinovy obecné teorie relativity.

1.1 Historie GPS

Konečná podoba systému GPS je odvozena od podobných pozemních radiových navigačních systémů. Jedním z nich je systém LORAN, který vznikl počátkem čtyřicátých let 20.století v USA a byl použit již během druhé světové války. Další inspirací pro systém GPS bylo vypuštění Sputniku Sovětským svazem v roce 1957. Tým amerických vědců, vedených Dr. Richardem B. Kershnerem, monitoroval radiové vysílání Sputniku.

Vědci zjistili, že díky Dopplerově efektu frekvence vysílaného signálu byla vyšší, když se Sputnik přibližoval a naopak nižší, když se vzdaloval. Uvědomili si, že při znalosti jejich přesné polohy na Zemi mohli určit přesnou polohu Sputniku na jeho oběžné dráze (při známých parametrech oběžné dráhy Sputniku) pomocí měření frekvenční změny. První družicový navigační systém Transit, byl používán námořnictvem Spojených Států, poprvé byl úspěšně otestován v roce 1960. Tento systém využíval pět družic a byl schopen určit polohu jednou za hodinu.

V roce 1967 námořnictvo Spojených Států vyvinulo družici Timation, ta ve svém vybavení jako první nesla do vesmíru přesné hodiny (GPS je založen na synchronizaci přesného času). Pozemní navigační systém Omega, fungující na porovnávání

fází signálu, byl v sedmdesátých letech 20. století prvním celosvětovým radiovým navigačním systémem. První experimentální GPS družice Bloku I (Block-I GPS) byla vypuštěna v únoru roku 1978. GPS družice byly nejdříve vyráběny společností Rockwell International a v současné době je vyrábí společnost Lockheed Martin.

V roce 1983, poté co sovětský stíhací letoun sestřelil korejské civilní letadlo KAL 007 v zakázaném sovětském leteckém prostoru a zahynulo všech 269 lidí na palubě, americký prezident Ronald Reagan oznámil, že GPS systém bude po dokončení přístupný i civilnímu použití. Bylo tomu hlavně proto, aby mohl být systém v krizových situacích využit pro lokalizaci raněných. V roce 1985 bylo vypuštěno deset dalších experimentálních družic Bloku I. 14. února 1989 byla vynesena na oběžnou dráhu moderní družice Bloku II, první svého druhu.

Do konce roku 1993 se GPS stal funkčním (zajištěno minimálního počtu družic pro určení polohy kdekoli na Zemi). 17. ledna 1994 bylo poprvé na oběžné dráze 24 družic potřebných pro plnou funkčnost systému. V roce 1996 americký prezident Bill Clinton oficiálně uznal důležitost GPS jak pro civilní, tak pro armádní sektor a vydal směrnici, která GPS definuje jako systém dvojího využití. Také založil Správní orgán GPS (Interagency GPS Executive Board), pro správu GPS jako národního majetku USA. V roce 1998 viceprezident Al Gore oznámil plán na modernizaci GPS, a to přidáním dvou civilních signálů pro vylepšení přesnosti a spolehlivosti, obzvláště ve vztahu k letecké bezpečnosti. 2. května 2000 byla vypnuta „Selective Availability“, což umožnilo civilním uživatelům přijímat plnohodnotný signál. V roce 2004 americký prezident George W. Bush nahradil správní orgán GPS jiným orgánem, a to National Space-Based Positioning, Navigation, and Timing Executive Committee.

1.2 Funkčnost systému

GPS se dělí na tři hlavní segmenty. Jsou to: kosmický segment (Space Segment, SS), řídící segment (Control Segment, CS) a uživatelský segment (User Segment, US).

1.2.1 Kosmický segment

Kosmický segment je tvořen GPS družicemi. 24 družic je rovnoměrně rozloženo v šesti oběžných rovinách. Oběžné roviny jsou centrické vzhledem k Zemi. Roviny mají přibližně sklon k rovníku 55° a jsou k sobě posunuty o 60° podél rovníku (posunutí rektascenze výstupních uzlů).

Družice obíhají ve výšce přibližně 20000 km, každá družice oběhne svou dráhu dvakrát za hvězdný den, takže přeletí nad stejným místem na Zemi jednou za den (oběžná doba je rovna 11h a 58 min,to znamená, že pozorovatel na Zemi uvidí

družici vycházet, pohybuje se stále po stejné trajektorii vždy o 4 minuty dříve). Oběžné dráhy jsou navrženy tak, že alespoň šest družic je vždy viditelných téměř z kteréhokoli místa na Zemi.

K lednu 2007 bylo v GPS konstelaci 29 aktivně vysílajících GPS družic. Doplňkové „extra“ družice vylepšují přesnost výpočtů GPS přijímačů, protože poskytují nadbytečná měření. Díky redundantnímu počtu satelitů se rozestavění celé konstelace změnilo na nesouměrné, což zajišťuje při selhání několika družic to, že systém zůstává plně funkční i při výpadku toku dat (to se týká hlavně spolehlivosti a dostupnosti celého systému).

1.2.2 Řídící segment

Dráhy letu GPS družic jsou sledovány monitorovacími stanicemi v těchto lokalitách: Havajské ostrovy, Kwajalein, Ascension, Diego Garcia a Colorado Springs, Colorado. Sledovací data (tracking information) jsou poslány do hlavní řídící stanice, která se nachází na Letecké základně Schriever (Schriever Air Force Base) v Colorado Springs, Colorado. Základna je pod velením Leteckých Sil Spojených Států, jmenovitě je spravována Druhou „Vesmírnou řídící“ skupinou (2nd space Operations Squadron, 2nd SOPS).

2nd SOPS pravidelně posílá každé GPS družici aktualizaci navigačních dat (využívá přitom pozemní antény, které jsou součástí zařízení stanic Ascension, Diego Garcia, Kwajalein a Colorado Springs). Tyto aktualizace synchronizují družicové atomové hodiny s přesností do jedné mikrosekundy a upravují družicové efemeridy, které jsou posléze vysílané družicí. Aktualizace jsou vytvořeny pomocí Kalmanova filtru, který využívá data od pozemních monitorovacích stanic, informace o „vesmírném počasí“ a další různé zdroje dat.

1.2.3 Uživatelský segment

GPS přijímače uživatelů tvoří uživatelský segment GPS. GPS přijímače se obecně skládají z antény, procesoru přijímače, a vysoce stabilních hodin (často s pasivní elektronickou součástkou zvanou krystal). Také mohou být vybaveny displejem, na kterém se uživateli mohou zobrazit údaje o jeho poloze a rychlosti. U GPS přijímače se často uvádí počet kanálů, které značí počet družic od kterých je přijímač najednou schopen přijímat signály. Původně se jednalo o 4 až 5 kanálů, ale v současné době se tento počet zvýšil na standardních 12 až 20 kanálů.

Součástí GPS přijímačů mohou také být zařízení pro příjem diferenciálních korekcí nebo také zařízení pro přenos dat do PC nebo jiných zařízení (např. bluetooth).

1.2.4 Známí rivalové [2]

Systém GPS není jediný svého druhu. Má několik rivalů na poli celosvětových lokalizačních systémů. Nejznámější a největší konkurence pro GPS jsou tyto systémy uvedené níže.

GLONASS

Ruský navigační systém, který začal vznikat již za studené války. Tento systém je v mnohém podobný GPS. Jeho charakteristickým znakem je konstelace jeho identického opakování rozmístění družic kolem Země, každých osm dní. Každá „orbitální rovina“ obsahuje 8 družic, po jednom hvězdném dni, v ní dochází k neidentickému opakování rozmístění družic (jiná družice zaujme místo původní).

Kompletní systém se skládá ze 24 družic, z toho 3 jsou záložní, v každé ze tří oběžných rovin jedna. V každé rovině je osm družic, identifikovatelné jsou pomocí pozičního čísla (číslo určuje odpovídající rovinu oběžné dráhy a pozici v rámci této roviny: 1-8, 9-16, 17-24). Roviny oběžných drah jsou vzájemně posunuty o 120° (podél roviny rovníku), družice v jedné rovině jsou vzájemně posunuty o 45°. Oběžné dráhy jsou kruhové, se sklonem k rovině rovníku 64.8° a hlavní poloosou o délce 25440 km. Družice obíhají Zemi ve výšce 19100km a každá z nich oběhne Zeměkouli každých 11 hodin a 15 minut. Družice využívají pro komunikaci FDMA multiplex.

GALILEO

Jedná se o evropský poziční systém, který měl být funkční již v roce 2010. Nové datum spuštění je naplánováno na rok 2018. Jedná se o projekt, který není narozdíl od GPS a GLONASS převážně vojenský. Jedná se o civilní systém a do budoucna se počítá pouze s civilní obsluhou tohoto systému. V plánu je systém sestávající se z 30 družic (27 operačních + 3 záložní) obíhajících ve třech rovinnách po kruhových dráhách na střední dráze Země (MEO - Medium earth Orbibit) ve výšce 23222km. Každá rovina má svírat úhel 56°, toto umožní využívat navigační systém až na 75° zeměpisné šířky. Systém by měl být spolehlivý díky velkému množství družic. Galileo umožní každému přijímači měřit polohy s přesností jednoho metru. Systém bude nabízet 4 základní druhy služeb:

- Základní služba (OS) - signál zdarma, základní signál;
- Komerční služba (CS) - kódovaná příplatková komunikace, využití přístupového klíče na straně přijímače;
- Veřejně regulovaná služba (PRS) - dva šifrované signály, kontrolovaný přístup, využití pro bezpočnostní složky státu;

- Vyhledávací a záchranná služba (SAR) - nouzová lokalizace v celosvětové družicové záchranné službě COSPAS-SARSAT;

BeiDou-2, Compass

Tento čínský systém navigace se odlišuje od všech ostatních sytému tím, že používá geostacionární družice. Nepotřebuje proto tolik družic jako ostatní systémy, ale také to znamená, že je pokryta pouze oblast, nad kterou je družice nastálo umístěna. Tento systém Beidou 1, funkční předchůdce BeiDou-2, ze kterého vychází, je v současné době funkční pouze v těchto souřadnicích: 70° až 140° východní délky a 5° až 55° severní šířky. Do budoucnosti se počítá s 35 družicemi, z toho 5 geostacionárních. Ty budou zajišťovat pokrytí celého zemského povrchu. Služba bude jak civilní tak vojenská. Přesnost civilní verze bude 10m.

1.3 Souřadnicový systém

Souřadnicový systém je systém udávání polohy pomocí zemských délek a šířek. Jako jsou 4 světové strany, jsou i 4 zemské souřadnice, 2 zemské délky (severní, jižní) a 2 zemské šířky (západní a východní). Kombinace dvou (jedné délky a jedné šířky) udává přesné místo na Zemi. Dělící pásmo délek je nultá rovnoběžka (rovník), u šířek je to pak nultý poledník.

GPS systém neudává tyto souřadnice tak jak by bylo hodno, udá pouze číselnou hodnotu. Souřadnicový systém je rozdělen do 4 kvadrantů, podobně jako Kartézský systém souřadnic. Pokud je hodnota záporná jedná se o hodnotu v záporné části souřadnicového pole (např.: u zemské délky by se jednalo o jižní délku, jižní polokoule). Příkladem může být výpis polohy města Brna: $49.195155; 16.607616$ tj. $49°11'42.6''$ severní délky; $16°36'27.4''$ východní šířky. Tento výpis je v české konvenci, v anglické by vypadal takto: $49°11'42.6''N; 16°36'27.4''E$. Tento systém zaručuje, že žádná dvě různá místa nemohou mít stejnou souřadnici.

1.3.1 GPX a KML soubory

Soubor typu KML (Keyhole Markup Language) je soubor, který vznikl pro zobrazování geografických dat napříč všemi zařízeními. Je založen na xml standardech. Funguje na bázi tagů, to znamená, že vždy nějaký tag začíná sekci a ukončuje ji stejný tag v kombinaci se zpětným lomítkem. Všechny tagy jsou case-sensitive a musí být v dokumentu umístěny v přesném pořadí, jaké je definováno. Skladbě těchto dokumentů se práce bude věnovat později, až při tvorbě těchto dokumentů v kapitole 5.3. Tyto souboru vznikly, při vzniku programu Earth View, který na trh uvedla firma

Keyhole. V roce 2004 firmu odkoupila společnost Google a přejmenovala program na Google Earth.

V roce 2008 se stal soubor typu KML ve verzi 2.2 standardem Open Geospatial Consortium (OGC), což je nezisková organizace, která zaštiťuje standardy týkající se geolokačních dat a jejich zpracování.

Existují také soubory typu KMZ, což je zip vytvořený ze souborů KML. Tento zip obsahuje `doc.kml` a dále různé jiné soubory, jako jsou obrázky, které se mohou používat na vizualizaci textur při prezentacích.

Soubor typu GPX (GPS Exchange Format) je soubor, také založený na xml standardech. Byl vytvořen v roce 2002 jako standard pro softwarové aplikace. Může být použit k zobrazení bodů, stop a cest. Základní aplikací mohou být soubory, pro zobrazování cest na jakémkoliv podkladu a přiřazení k nim různých fotografií s označením polohy. Tomuto formátu se bude práce také věnovat později v kapitole 5.3

2 OPERAČNÍ SYSTÉM ANDROID

[3]

Android je operační systém vyvinutý hlavně pro dotyková mobilní zařízení jako jsou smartphone nebo tablet. Android je majetkem společnosti Android Inc., která patří pod skupinu Google Inc. Poprvé se objevil na trhu v roce 2007. V tomto roce také spolu s ním vznikla Open Handset Alliance, což je společenství 86 harwarových, softwarových a telekomunikačních společností, které se věnuje prosazování open standardù v oblasti mobilních zařízení.

Google vydal Android jako open-source systém s licencí Apache. Původně záměr Google byl vytvářet všechny aplikace, ale po nějaké době se „androidoví“ nadšenci sami pustili do vývoje aplikací. Pro toho kdo umí Javu to není nic složitého, jelikož aplikace se píší v upravené verzi Javy. Hotovou aplikaci může vývojář nahrát na Google Play (pùvodně Android Market), kde je většina z nich zdarma. Jen pro zajímavost, v únoru roku 2016 bylo na Google Play již umístěno kolem 2000000 aplikací pro Android a od začátku tohoto serveru bylo staženo přes 65 miliard aplikací.

První telefon s OS Android se začal prodávat v říjnu 2008 a koncem roku 2010 se stal nejprodávanějším operačním systémem pro mobilní telefony. Porazil tak Symbian (systém vyvinutý společností Nokia pro jejich přístroje), který byl do té doby na prvním místě. Celosvětově se Android dodává v 75 % ze všech mobilních zařízeních, s 1,4 miliardy aktivovanými zařízeními a s 1,5 milióny aktivací každý den. V současné době se začíná Android prodávat i v zařízeních A/V techniky jako jsou televize apod.

2.1 Uživatelské prostředí

Uživatelské prostředí Androidu, u mobilních telefonů, je založeno na přímé interakci mezi uživatelem a operačním systémem pomocí dotykové obrazovky. Prostředí je velmi interaktivní a tzv. „user friendly“. Jelikož ne všechno se dá pomocí dotyků na obrazovce ovládat, je zařízení většinou vybaveno akcelerometry a gyroskopem pro efektivnější ovládání systému. Pomocí akcelerometrů se například rozhoduje o natočení obrazovky.

2.1.1 Aplikace

Počet aplikací pro Android pořád stoupá, hlavně stoupá jejich počet od třetích stran. Aplikace mohou vývojáři nahrávat na on-line obchod Google Play nebo na spřátelený obchod Amazon Appstore. Další možností je nahrát aplikaci do přístroje přímo,

například přes kabel z počítače (toho se využívá především při vývoji k testování na hardwarovém zařízení). Google Play umožňuje uživatelům sdílet, aktualizovat nebo stahovat aplikace přímo od Googlu nebo od ostatních vývojářů a také je nainstaluje do zařízení s plnou požadovanou kompatibilitou. Aplikace mohou být tříděny do různých skupin například podle typu přístroje (jsou pak viditelné jen uživatelům s vybraným typem přístroje) nebo například podle regionů prodeje.

K vývoji aplikací může vývojář využít mnoho nástrojů. Dlouho rozšířenou variantou bylo propojení Android SDK a vývojového prostředí Eclipse, nástrojem pro vývoj v jazyce Java. Dnes se již dá také využít nástroje, který je určený přímo pro vývoj Android aplikace. Tento nástroj se jmenuje Android Studio a je k dispozici zdarma na vývojových stránkách Androidu [4]. Nástroj je do určité míry kopií Eclipse, ale upravený na plnou kompatibilitu s přístroji běžících na platformě Android. U varianty s Eclipse se často stávalo, že ne vždy prostředí dobře komunikovalo s přístrojem a poté vznikali nemalé chyby při vývoji.

K mání jsou i jiné způsoby vývoje, například pomocí NDK (Native Development Kit), který se využívá pro programování v jazycích C a C++. Tento způsob vývoje, ale není moc rozšířený.

2.2 Vývoj Androidu

2.2.1 Linux

Android se zakládá na Linuxovém jádru Linux Kernel s knihovnami, s API napsanými v C a aplikacemi, které jsou kompatibilní s Java knihovnami založenými na licenci Apache. Android používá Dalvik virtual machine[1] s just-in-time kompilací k rozběhnutí dex[2] kódù, které jsou překládány z Java bytecodů. V praxi to funguje tak, že zdrojový kód napsaný v Javě, je přeložen do .jar aplikace, která je přes Dalvik přeložena na dex kód, ten je pak vyhodnocen systémem a spuštěn. Tento zdánlivě komplikovaný způsob je vytvořen k úspoře operační paměti a výpočetního výkonu.

2.2.2 Organizace paměti

Jelikož zařízení s OS Android jsou především napájeny přes baterie, je tento systém designován s pamětmi RAM k udržení co nejmenší spotřeby energie. Tímto se liší od stolních počítačů, které jsou připojeny do rozvodné sítě a spotřeba energie není až tak důležitá. Když se aplikace již nepoužívá je uložena do paměti, odříznuta od

[1] DVM–virtuální prostředí, ve kterém se spouští aplikace. Navrhnuto a realizováno lidmi z Google. Chová se podobně jako Java Virtual Machine, ale je jednodušší a ořezanější.

[2] Zdrojový kód aplikací na úrovní assembleru.

jakékoli spotřeby energie a čeká na své opětovné spuštění. Navíc to ve skutečnosti zrychluje její pozdější náběh při obnovení, jelikož systém ji nemusí pokaždé spouštět od začátku, ale naváže v ní na tom místě, kde byla odložena do paměti, je pouze vyvolána a nepotřebuje tolik energie jako při studeném startu.

Android si také sám organizuje paměť. Pokud je jí nedostatek tak aplikace, která byla použita jako první a je tedy „nejstarší“, je ukončena a na její místo se posouvá 2. „nejstarší“. Tato funkce je velmi podobná funkci garbage collectoru v jazyku Java.

2.2.3 Dostupnost aktualizací

Google poskytuje hlavní aktualizace systému Android každých 6–9 měsíců. Jsou poskytovány koncovým uživatelům přes Internet. Poslední aktualizací je 6.0 Marshmallow. Na cestě je již také nová verze, prozatím označována jako N.

V porovnání s jinými mobilními operačními systémy, jmenovitě iOS od Applu, se nové verze Androidu dostávají k uživatelům o mnoho později, typicky o několik měsíců od uvedení na trh. Je to hlavně díky tomu, že Android je poskytován několika výrobcům hardwaru a každý z nich má jiné nároky na výkon systému. Výrobci si tyto verze musí upravovat a to právě způsobuje to zpoždění, které například u iOS nehrozí. Někteří výrobci mobilních zařízení někdy ani nechtěli poskytovat nové verze systému na starší modely, ale raději vytvořili nový telefon a ten nabízeli s novou verzí operačního systému. Kvůli tomu se Google rozhodl v roce 2011 vytvořit úmluvu Android Update Aliance, ve které říká, že každý mobilní přístroj musí být aktualizován novou verzí systému minimálně jednou za 18 měsíců.

2.3 Vývoj aplikací

[5]

Operační systém Android využívá především mobilní telefony. I když Google oznámil, že by Android mohl najít daleko rozsáhlejší uplatnění (počítače, palubní desky automobilů), v současné době se klade důraz na vývoj pro mobilní telefony. Pro vývojáře to má své výhody i nevýhody.

Mezi nesporné výhody patří, že jsou „chytré“ telefony nezvykle přitažlivé pro uživatele. Počátek internetových služeb v mobilních telefonech se datuje již do poloviny 90. let od vytvoření jazyka Handheld Device Markup Language (HDML). Skutečně se telefony schopné připojení k Internetu prosadili až v posledních několika letech. Díky trendům, jako je psaní textových zpráv nebo díky zařízení iPhone od Apple, získávají telefony, prostřednictvím nichž se lze připojit k Internetu, rychle na popularitě.

Problém ovšem nastává při samotném programování. Při programování pro mobilní telefony narážíme na to, že telefony jsou ve všech ohledech malé.

- Telefony mají velmi malou obrazovku.
- Klávesnice jsou velmi malé (pokud je jimi telefon vybaven).
- Rychlost CPU a velikost paměti je v porovnání s PC malá (v dnešní době už to není takový problém).
- Při psaní je možno použít jakýkoliv jazyk nebo prostředí, ale jenom pokud se to výrobce telefonu rozhodl podporovat.

Kromě toho jsou zde také úskalí s tím, že se jedná o mobilní telefon a tuto funkci musí plnit především. Nesmí se nám tedy stát, že telefon přestane fungovat například z těchto důvodů:

- Aplikace vytíží procesor telefonu takovým způsobem, že telefon není schopný přijmout příchozí hovor.
- Aplikace odmítá tiše zmizet do pozadí, když telefon upozorní uživatele na příchozí hovor nebo uživatel chce někomu zavolat, protože nespolupracuje se zbytkem operačního systému.
- Aplikace způsobí zhroucení operačního systému, například v důsledku masivních úniků paměti.

Z těchto důvodů se tedy vývoj aplikací pro telefon liší od vývoje programů pro stolní počítače, webové stránky nebo obslužné procesy serverů. Liší se vývojové nástroje, jinak se chová aplikační rámec a funkce programů podléhají mnoha různým omezením.

Android se pak snaží o kompromis:

- Umožňuje psát aplikace v běžně dostupném používaném programovacím jazyce (Java), používat některé běžné knihovny (například některé z Apache Commons API) a nástroje, na které jsme zvyklí (Eclipse).
- Nabízí poměrně jasně stanovený a oddělený aplikační rámec, ve kterém programy musí běžet, takže mohou být „dobrými občany“ telefonu a nenarušují funkci ostatních programů nebo samotného telefonu.

2.3.1 Složení aplikací

Když píšeme aplikaci pro stolní počítač, jsme „pány své vlastní domény“. Spustíme její hlavní okno a tolik oken jeho potomků – například dialogů – kolik potřebujeme. Z našeho úhlu pohledu se jedná o náš vlastní svět, ve kterém sice využíváme vlastnosti operačního systému, ale příliš se nestaráme o další programy, které mohou být v počítači spuštěny současně s naším. Pokud s nimi nějak intereagujeme, děláme to obvykle prostřednictvím nějakého API, jako je například Java Database Connectivity (JDBC) nebo na něm založený aplikační rámec, abychom mohli komunikovat s

MySQL nebo jinými databázemi.

Systém Android pak využívá podobné koncepce, ale jinak zabalené a strukturované s ohledem na větší ochranu telefonů proti zhroucení systému. Aplikace pro systém Android sestávají z následujících komponentů:

Aktivity (Activities)

Stavebními kameny pro tvorbu uživatelského rozhraní jsou aktivity. Můžeme je chápat jako analogii oken či dialogů aplikace pro stolní počítač. Ačkoliv je možné, aby aktivity neměly uživatelské rozhraní, většina zdrojového kódu bez uživatelského rozhraní bude zabalená spíše ve formě dodavatelů obsahu anebo služeb.

Dodavatele obsahu (Content Providers)

Dodavatele obsahu zajišťují úroveň abstrakce jakýchkoliv dat uložených v zařízení, která jsou přístupná více různým aplikacím. Vývojový model systému Android nás podporuje v tom, abychom zpřístupnili svá data i jiným aplikacím než pouze své vlastní. Dosáhneme toho pak právě vytvořením dodavatele obsahu, který nám současně poskytuje úplnou kontrolu nad způsobem přístupu k našim datům. Mezi tyto dodavatele patří například prohlížeč souborů nebo události vytvořené v kalendáři, případně sdílené položky kontaktů.

Služby (Services)

Aktivity a dodavatele obsahu jsou entity s krátkou životností a lze je kdykoliv vypnout. Služby jsou oproti tomu navržené tak, aby, pokud je to potřeba, pokračovaly ve své práci nezávisle na jakékoliv aktivitě. Službu můžeme použít k detekci aktualizací RSS zdroje nebo k přehrávání hudby, které pokračuje, dokonce i když byla příslušná ovládací aktivita uzavřena. Tato služba běží na pozadí systému a měla by mít co nejmenší výpočetní nároky, vzhledem k tomu, že se počítá s jejím nepřetržitým chodem. Pokud by byla moc výpočetně náročná mohla by zbytečně mnoho ubírat výdrži baterie.

Záměry (Intents)

Záměry jsou systémové zprávy upozorňující aplikace na výskyt různých událostí, změnami hardwarové konfigurace počínaje (například vložení SD karty), přes události související s příchozími daty (například přijetí SMS zprávy) a událostmi aplikace konče (například její spuštění z hlavního menu zařízení). Na záměry pak můžeme nejenom reagovat, ale můžeme také vytvářet své vlastní záměry a spouštět jejich prostřednictvím jiné aktivity nebo je využívat k detekci specifických situací

(například vyvoláme takový a takový záměr, když se zařízení dostane do vzdálenosti 100 metrů od takové a takové lokace). Nejtypičtějším příkladem je spuštění komunikačních médií (Bluetooth, Wi-Fi atd.), kdy se vytvoří Intent s příslušným médiem.

2.3.2 Vlastnosti a nástroje systému

Systém Android nabízí množství vlastností a nástrojů, které jsou využitelné při vývoji:

Uložiště

Datové soubory, jejichž obsah se nebude nijak měnit (například ikony nebo pomocné soubory), můžeme přibalit přímo k aplikaci. Navíc také můžeme ukořistit trochu volného místa v zařízení a uložit data do databází nebo souborů obsahujících uživatelem zadávaná či načítaná data, která aplikace potřebuje.

Síť

Zařízení využívající operační systém Android jsou obecně vzato připravená pro připojení k Internetu prostřednictvím různých druhů komunikačních médií. Přístup k Internetu můžeme využít na jakékoliv úrovni chceme, surovými Java sockety počínaje a zabudovaným widgetem webového prohlížeče založeným na jádře WebKit konče.

Multimédia

Zařízení využívající OS Android umí přehrávat a zaznamenávat audio–videozáznamy. Ačkoliv konkrétní specifika těchto schopností se liší zařízení od zařízení, můžeme se zařízení zeptat, jakými schopnostmi pro přehrávání a záznam audio a videozáznamů disponuje, a poté tyto schopnosti využít, jak uznáme za vhodné – ať už k přehrávání hudby, pořizování fotografií fotoaparátem nebo záznamu audiopoznámek prostřednictvím mikrofonu.

GPS

Zařízení využívající operační systém Android mají často přístup k dodavatelům údajů o zeměpisné poloze, například ke službě GPS, která může poskytnout informace o tom, kde na zeměkouli se zařízení zrovna nachází. Pak můžeme následně na základě těchto informací zobrazovat mapy nebo je využívat nějak jinak, například ke sledování pohybu zařízení v případě jeho odcizení. Tohoto nástroje bude

využito pro vývoj aplikace. Nakládání s tímto nástrojem bude popsáno v následující kapitole.

Telefonní služby

Protože jsou zařízení využívající operační systém Android obvykle mobilní telefony, může aplikace samozřejmě také provádět telefonická volání, zasílat a přijímat SMS zprávy a provádět jakékoliv další operace, které od moderního telefonu očekáváme.

3 NÁVRH A VÝVOJ TESTOVACÍ APLIKACE

Cílem semestrální práce je vytvořit aplikaci, která by ověřila, zda je možné využít GPS komunikační modul pro vytvoření služby s tímto modulem. Pro otestování této funkčnosti je nutné mít hardware, na kterém lze otestovat funkčnost. Tímto hardwarem je myšlen přístroj s OS Android a komunikačním modulem GPS. V dnešní době již není problémem využít v podstatě jakýkoli chytrý mobilní telefon. Pro účely testování byl využit mobilní telefon HTC One $X+$ a verzí Androidu 4.2.2 Jelly Bean s úpravou HTC Sense 5.0.

Aplikace byla psána v programu Android Studio 1.4 s podporou Java JDK 8.

3.1 Přístupová práva

Pro povolení užívání GPS modulu v telefonu je potřeba získat práva na jeho užívání. Tato práva jsou specifikována v manifestu aplikace a jsou tázána při instalaci. Tata práva jsou:

- *ACCES_FINE_LOCATION* - pro zjišťování polohy pomocí GPS
- *ACCES_COARSE_LOCATION* - pro zjišťování polohy pomocí síťového připojení

V aplikaci je využito obou přístupů k aktualizaci lokace z toho důvodu, že zjištění poslední známé lokace je rychlejší přes síťové připojení k Internetu, než přes GPS. Po zjištění poslední známé polohy probíhá vyhledávání pouze přes GPS.

3.2 Location Strategies

Ke komunikaci s GPS modulem v telefonu se využívá takzvaných Location Strategies, které jsou specifikovány na stránkách developerů[6]. Základní myšlenkou této strategie je využití API android.location, součástí této API jsou třídy LocationManager a LocationProvider. Tato API je dostupná všem zařízením s OS Android, která podporují GPS komunikaci. Třída LocationManager slouží k vytvoření správce spojení, třída LocationProvider zastupuje zvolený modul, ze kterého se budou získávat informace o lokaci. Pokud aplikace získá přístupová práva (uživatel je udělí při instalaci), musí se zjistit, zda je GPS modul, který bude používaný, zapnutý. Na to slouží metoda `turnGPSOn()`, která zjistí zda je zapnutý, a pokud ne, vznese na uživatele dotaz, zda se má zapnout. Od verze Androidu 4.0 se přistupuje k tomuto povolení spuštění tak, že uživatel je přesměrován do nastavení systému, kde musí zaškrtnout pole *Satelity GPS*, aby sloužba byla aktivní. Před touto verzí se vznesl

dotaz, zda se má GPS zapnout a v následujícím pop-up okně se GPS zapla, podobně jako je tomu například u Wi-Fi. Celá syntaxe vypadá následovně:

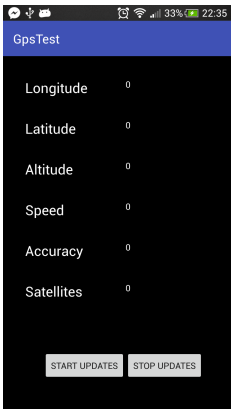
```
private void turnGPSOn() {
        String provider =
           Settings.Secure.getString(getContentResolver(),
                Settings.Secure.LOCATION_PROVIDERS_ALLOWED);
        if (!provider.equals("")) {
            //GPS Enabled
            Toast.makeText(GpsTest.this, "GPS Enabled: "
               + provider,
                    Toast.LENGTH_LONG).show();
        } else {
            Intent intent = new
               Intent(Settings.ACTION_SECURITY_SETTINGS);
            startActivity(intent);
        }
    }
```

Po spuštění GPS se začne aplikace inicializovat. Pro inicializaci jsou napsány dvě metody. Jedna z nich zobrazí pouze hodnoty 0 v textových polích, která jsou viditelná na 3.1. Druhá metoda nastaví jako výchozí hodnotu poslední známou polohu (příklad tohoto výpisu 3.2). Syntaxe je velmi jednoduchá:

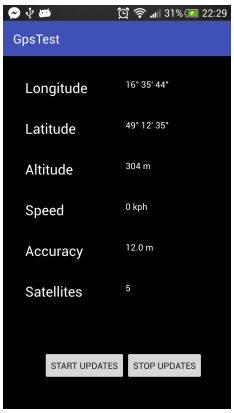
```
private void initializeLastLocation(LocationManager
   locationManager) {
        makeuseOfNewLocation(
        locationManager.getLastKnownLocation(
        locationProviderGPS));
        }
```

Tato metoda využívá stejnou metodu, která se stará o výpis nových souřadnic, a to metodu `makeUseOfNewLocation`. Tato metoda bude specifikována později. Funkce je velmi prostá, vezme stejnou metodu pro aktualizaci a místo nové polohy, zjistí poslední známou.

Důležité zjištění, které vyplynulo z používání této aplikace je takové, že poloha se určuje globálně a pokud se zjistí poslední hodnota v jiné aplikaci anebo službě, bude tato hodnota uložena ve stejné proměnné, kterou využívají všechny aplikace, jenž mají právo k nakládání s polohou. V praxi to znamená, že pokud je telefon připojen do mobilní sítě, tak i z této sítě získává informace o poloze, tzn. že poslední poloha je vždy aktuální a neznamená to, že by to byla poslední hodnota, která byla zjištěna touto testovací aplikací. Jediným způsobem jak využívat pouze této aplikace



Obr. 3.1: Inicializace, hodnoty 0



Obr. 3.2: Inicializace, poslední lokace

k zjišťování polohy by bylo, vypnout všechna ostatní bezdrátová či drátová spojení, což je velmi nepraktické pro uživatele hlavně z toho hlediska, že mobilní telefon by měl v první řadě sloužit ke komunikaci a vypnutí všech komunikačních prostředků je velmi kontraproduktivní.

Po inicializaci se zmáčknutím tlačítka *START UPDATES* začne aktualizovat poloha, která je předávána do textových polí a zobrazena na obrazovce. Tato aktualizace nemusí být hned rozeznatelná, vzhledem k tomu, že poslední známá poloha se může shodovat s nově nalezenou hodnotou. Jediné pole, kde se objeví změna je pole *Sattelites*, kde se zobrazí počet připojených satelitů. Aktualizace je vyvolána pomocí třídy Location Manager. Pro nastavení managera se používá třída Location Provider (GPS, NETWORK). Tím se specifikuje, co má být použito pro aktualizování lokace, jestli GPS modul nebo informace ze sítě.

Podle doporučení vývojářů se má k tomuto problému, zda aktualizovat pomocí sítě nebo GPS přistupovat tak, že první poloha by měla být zjistěna ze sítě. Tento způsob je zvolen kvůli tomu, že tato aktualizace je rychlejší, než aktualizace z GPS. Navíc aktualizace ze sítě není tak systémově náročná, protože se počítá s tím, že se zařízení nachází v blízkosti Wi-Fi nebo má aktivováno datové připojení. Tím pádem se nemusí navazovat počáteční spojení do sítě, tak jako to dělá GPS modul. Sestavení spojení spotřebuje nějaký čas a následná komunikace je také určitě pomalejší, než datové nebo Wi-Fi připojení. Zároveň je čas zjišťování polohy ze sítě, využit k navázání spojení komunikace pomocí GPS. Tato funkce subjektivně zrychluje odezvu připojení. Následná aktualizace polohy je zjišťována pouze pomocí GPS. Četnost aktualizace je pak závislá na parametrech, které se zvolí v kódu. Aktualizace se nastavuje parametry v příkazu:

```
locationManager.requestLocationUpdates(String provider, long minTime,
float minDistance, LocationListener listener)
```

kde `locationManager` je manager, který zajišťuje komunikaci, `provider` je vybraný provider, který má být použit k aktualizaci lokace, `minTime` je minimální čas mezi aktualizacemi, `minDistance` je minimální vzdálenost, po které se se znovu lokace obnoví a `listener` je třida LocationListener, která naslouchá změnám v lokaci a na jejím základě se zvolený `locationManager` dozví, že se poloha změnila. Proměnnými `minTime` a `minDistance` se dá nastavovat četnost obnovy polohy. Tyto proměnné jsou klíčové k tomu, aby se našel kompromis mezi četností a spotřebou energie, protože platí, že čím častěji se poloha aktualizuje, tím rychleji klesá zbytková kapacita baterie. V testovací aplikaci je nastaveno staticky aktualizovat jednou za 5s nebo změní-li se poloha alespoň o 25m.

LocationListener obsahuje několik metod, z nichž nejzajímavější pro testování je metoda `onLocationChanged`, která při každé změně polohy uloží novou polohu do proměnné `location`. Ta je předávána do metody `makeUseOfNewLocation`, která je vytvořena pro aktualizaci uživatelského prostředí. Celý kód pro obnovu dat pak vypadá takto:

```
private String locationProviderGPS; //globalni promenna
   GPS Provider

locationManager = (LocationManager)
   this.getSystemService(Context.LOCATION_SERVICE);
   //lokalni promenna funkce onCreate(Bundle
   savedInstanceState);

locationStatus(locationManager); //spusteni metody
   locationStatus z metody onButtonStartClicked

@TargetApi(Build.VERSION_CODES.M)
    private void locationStatus(LocationManager
       locationManager) {
        LocationListener locationListener = new
           LocationListener() {
            @Override
            public void onLocationChanged(Location
               location) {
                makeUseOfNewLocation(location);
            }
```

```
            public void onStatusChanged(String provider,
                int status, Bundle extras) { }

            public void onProviderEnabled(String
                provider) { }

            public void onProviderDisabled(String
                provider) { }
        };
        locationManager.requestLocationUpdates(
        locationProviderGPS, 5000, 25, locationListener);
        }
```

Metoda `makeUseOfNewLocation` pak obnoví data v textových polích, jak je vidět na obrázku 3.2. Jelikož výpis z proměnné `location` by byl nic neříkající, je využito další metody `convertToDegrees`, která naformátuje surová data do podoby jaká se udává v běžných výpisech (např.: 49°23′56″). Metoda počítá souřadnice odečtením celého čísla od hodnoty lokace a následně zbytek vynásobí 60. Tento postup je zopakován dvakrát po sobě k získání stupňů, minut i sekund. Sekundy se jako poslední hodnota zaokrouhlují na celé číslo. Obě metody pak vypadají takto:

```
private void makeUseOfNewLocation(Location location) {
    editTextLongitude.setText(convertToDegrees(location.
        getLongitude()));
    editTextLatitude.setText(convertToDegrees(location.
        getLatitude()));
    editTextAltitude.setText(String.valueOf(
        location.getAltitude()) + " m");
    editTextSpeed.setText(String.valueOf(location.getSpeed())
      + " kph");
    editTextAccuracy.setText(String.valueOf(location.
        getAccuracy()) + " m");
    editTextSatellite.setText(String.valueOf(location.
        getExtras().getInt("satellites")));
    }

private String convertToDegrees(double decimal) {
    String s;
    int deg, min, sec;
    double c, d;
```

```
deg = (int) decimal;
c = decimal - deg;
d = c*60;
min = (int) d;
c = d - min;
d = c*60;
sec = (int) d;
s = deg + "   " + min + "' " + sec + "\" ";
return s;
}
```

Jelikož textová pole jsou podporována pouze typy `String` nebo `char`, musí být tyto hodnoty přetypovány. Následně se k hodnotě výstupu přidá jednotka, ve které je prezentována. Dále aplikace obsahuje metody `onStart`, `onStop`, `onPause` a `onResume`, které se starají o propojení se systémem.

3.2.1 Test aplikace

Jak již bylo napsáno, aplikace se testovala na mobilním telefonu HTC One $X+$ a verzí Androidu 4.2.2 Jelly Bean s úpravou HTC Sense 5.0. Při testování v klidu, bez většího pohybu, se aplikace zdála být nefunkční. Relativně dlouho trvá, než se výpis změní, je to otázka přibližně 20s k první změně. Nejlépe se dá však funkčnost ověřit při rychlém pohybu přístroje (například v jedoucím automobilu). Aktualizace dat pak probíhá velmi rychle, stěžejní je však nastavení četnosti aktualizace. Při nastavení, které je uvedeno výše se aplikace chová stabilně a při delším spuštění nevybíjí mobil až tak dramaticky. Pokud by se měla tato aplikace dále rozvíjet, tak by nebylo na škodu, aby si uživatel sám zvolil tyto parametry při spuštění, případně v nastavení. Nicméně je-li tato aplikace pouze pro test funkčnosti a ověření proveditelnosti, není žádná potřeba toto vytvářet. Jediné co se v testu nepodařilo ověřit, je vypnutí aktualizace tlačítkem. Nefunkčnost zastavení aktualizace na přání uživatele je sice problém, ale pokud se má z tohoto postupu stát služba, je přistupování k „průběhu života“ aplikace a služby zcela odlišné, není to až tak zásadní problém pro tuto práci.

4 LOKALIZACE JAKO SLUŽBA

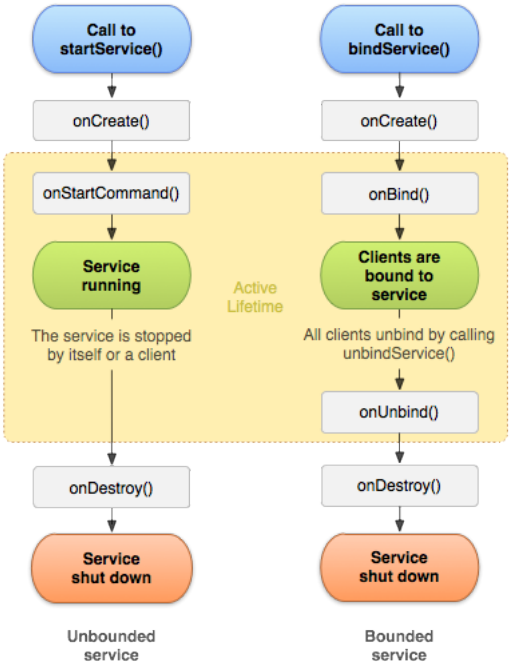
Při návrhu lokalizační služby se musí brát zřetel na to, že tato služba poběží nepřetržitě, pokud poběží activita, která ji ovládá. Jelikož služba využívá systémové prostředky, jako připojení k Internetu a GPS lokátor, je potřeba šetřit s energií přístroje. V souladu se všemi těmito skutečnostmi bylo počítáno při návrhu a tvoření této aplikace, respektive služby. Výsledná služba bude v následujících bodech popsána a bude uvedeno, proč je k těmto věcem přistupováno tak, jak je.

Služba se jmenuje `GPS_Service` a je součástí aplikace `GPS Finder`, která bude popsána později v následující kapitole.

4.1 Koloběh života služby v OS Android

Pro návrh služby je nejprve potřeba, ujasnit si její koloběh života. To znamená, že se musí stanovit, kdy bude služba spuštěna, kdy bude ukončena, případně bude-li možnost ji pozastavit a znovu spustit.

Služba lokalizace, která je zde navržena, je spuštěna při startu aplikace, která ji ovládá a je ukončena pouze tehdy, pokud je ukončena i tato aplikace. Žádné pozastavení není uvažováno. Je navržena jako vázaná na aktivitu, tzv. *Bound Service*. Tento typ služby je spouštěn metodou `bindService()`. Rozdíl mezi vázanou a nevázanou metodou je vidět na obrázku 4.1.



Obr. 4.1: Rozdíl mezi nevázanou (vlevo) a vázanou (vpravo) službou[7]

Při spouštění nevázané služby, dochází k tomu, že se služba dá vypnout sama, pokud dokončí všechny zadané úkoly, nebo ji může vypnout jakákoliv jiná aplikace. V tomto případě služba zaniká i za té podmínky, byla by připojena k více aktivitám (clientům).

Naopak spouštění vázané služby zabezpečí to, že tato služba zanikne až tehdy, jsou-li všechny aktivity, které se k ní vážou, ukončeny. Pro ukončení komunikace, mezi službou a poslední svázanou aktivitou, slouží metoda `onUnbind()`. Hlavním rozdílem je tedy to, že vázaná služba končí svůj život až tehdy, pokud již není využívána, zatímco nevázaná služba může ukončit svůj život kdykoliv.

Metoda `onBind()` slouží ke komunikaci s clientem. Přetížením této metody se vrací `Binder`, což je v podstatě ukazatel na běžící službu. Tento ukazatel je nutný proto, aby se mohl nastavit listener, který naslouchá na komunikačním portu a přeposílá data. `Binder` je typu `IBinder`. Jelikož komunikace se službou je řešena poněkud jednodušeji, může se zdát, že je tento `Binder` nepotřebný. Opak je ale pravdou. Pokud by neexistoval, nedalo by se na službu nijak odkázat z activit, případně fragmentů (fragmenty budou vysvětleny v kap. 5.1.2). `Binder` také slouží k otevření komunikačních portů na pozadí. V kódu pod textem jde vidět, jak `Binder` vypadá.

```
public class GPSBinder extends Binder {
        GPS_Service getService() {
            Log.e(TAG, "getService");
            return GPS_Service.this;
        }
    }
```

Vázaná služba byla zvolena hlavně proto, aby byla umožněna jednodušší komunikace s uživatelským prostředím. Zároveň je vytvořena tak, aby jí mohla používat pouze aplikace `GPS Finder` a byla tak zaručena kontrola toho, že se služba vypíná tak, jak má a že nezůstává spuštěna na pozadí, když to není žádoucí.

Zbytek metod co jdou vidět na obrázku 4.1, jsou pro vytvoření a ukončení služby. V metodě `onCreate()` se provede vše, co je potřeba při vzniku služby. Může zde být například vytvoření žádosti na aktualizaci polohy nebo vytvoření komunikačního socketu. Metoda `onDestroy()` poté slouží k tomu, co se má udát při zániku služby, jako je odregistrování socketu a zánik žádosti na aktualizaci polohy. Tyto metody fungují stejně, jako ty, které jsou součástí aktivit. Služba může mít také metody `onPause()` a `onResume()`, které však nejsou v tomto případě využity, jelikož služba poběží i na pozadí a proto ji není potřeba pozastavovat.

4.2 Komunikace se službou

Komunikaci se službou je možno realizovat v několika variantách. Jednou z nich je použití `Binderu` jak je popsáno výše. Dalším způsobem je využití `Binderu` v kombinaci s Messangerem, který by přeposílal zprávy. Tato varianta je vhodná při vytváření různých messangerů, kteří na pozadí naslouchají na příchozí zprávy. Poslední variantou je využití `BrodcastReceiveru` a poslílání dat využitím záměru, pomocí třídy `Intent`.

Pro tohle řešení byl využit posledně zmíněný způsob, protože je z těch výše popsaných nejjednodušší, nejlépe implementovatelný. Navíc se mohou přeposílat přímo datové bloky, což je velmi výhodné, jelikož se posílá velké množství hodnot. Tento způsob funguje tak, že se vytvoří `Intent`, do kterého se přidají extra položky, jako je například zpracovaná poloha na výstup do uživatelského prostředí. K těmto datům se přiřadí znak, podle kterého lze tento `Intent` rozpoznat a odešle se broadcastem do zařízení. Na straně aktivity (fragmentu) pak stačí vytvořit `BrodcastReceiver`, který naslouchá na příchozí broadcasty a vyfiltrovat zvolený `Intent`, pomocí třídy `IntentFilter` a zvoleného klíče. Výsledný kód na straně služby může vypadat například takto:

```
Intent broadcastLocationIntent = new Intent(LOC_UPDATE);
        broadcastLocationIntent.putExtra(LOC_LONGITUDE,
           getLongitude(location));
        broadcastLocationIntent.putExtra(LOC_LATITUDE,
           getLatitude(location));
        broadcastLocationIntent.putExtra(LOC_ACCURACY,
           getAccuracy(location));
        sendBroadcast(broadcastLocationIntent);
```

Metody typu get (například `getLongitude(location)`) jsou obyčejné get metody pro získání proměnných, které jsou chráněny (private proměnné). Pro odesílání dat slouží funkce `sendBroadcast()`, která tato data zasílá vytvořeným záměrem (`Intent`) do zařízení. Klíč je v tomto případě zvolen jako `LOC_UPDATE`.

Na straně aktivity se přetížením metody `onReceive()` získají všechna potřebná data, která jsou žádána. Součástí každé aktivity nebo fragmentu jsou dvě metody `registerReceiver()` a `unregisterReceiver()`. Ty slouží k registrování respektive odregistrování `BrodcastReceiveru` pro naslouchání.

Výpisem a zobrazením hodnot se bude tato práce zabývat až v následující kapitole 5. Součástí služby je také oznámení o zapnutí této služby. Při startu služby se objeví na dialogovém okně notifikace, že se služba spustila a dává uživateli možnost se pomocí této volby dostat do aplikace, kdyby chtěl například změnit četnost ak-

tualizace pro zmenšení spotřeby energie. Vše je řešeno třídou `Notification`, která je součástí Androidu. Je zde mnoho způsobů jak oznámení vytvořit, třída umožňuje například přiřadit ikonu k oznámení, napsat jakýkoliv text atp.

Příklad vytvoření `BrodcastReceiveru` ve fragmentu `LocationFragment`, pro příjem aktualizace lokace:

```
final IntentFilter intentFilter = new
   IntentFilter(GPS_Service.LOC_UPDATE);
        mBroadcastReceiver = new BroadcastReceiver() {
            @Override
            public void onReceive(Context context,
               Intent intent) {
                Log.e(TAG, "onReceive");

                VYPIS HODNOT

            }
        };
        getContext().registerReceiver(mBroadcastReceiver,
           intentFilter);
```

4.3 Komunikace s okolním světem

Při návrhu komunikace s Internetem nebo venkovní sítí, aby bylo možno přeposílat data o poloze, bylo uvažováno několik modelů. Jedním z nich bylo využití Messangerů, které by posílaly údaje o poloze. Jednalo by se pouze o pozemskou šířku a délku, protože tyto dva údaje stačí na to, aby se dal objekt zobrazit na mapě. Tento způsob by fungoval pouze tehdy, byla by uvažována komunikace pouze na úrovni Androidu.

Jelikož se má jednat i o přeposílání polohy na počítač, nebylo možné tento způsob využít. Při výběru nakonec vyšly nejlépe sockety. Tedy naslouchání na zvoleném portu. Tento způsob je velmi efektivní, jelikož všechna zařízení naslouchají na stejném portu, navíc více zařízení může naslouchat najednou.

Jedná se o komunikaci typu Server-Client, kde server je mobilní přístroj, a client je počítač. Port, na kterém zařízení komunikují byl zvolen na číslo 2001, bude ho ovšem možné změnit v nastavení aplikace. Jedinou nevýhodou je to, že při směrování není možné získat IP adresu serveru, jelikož je v každé síti jiná. Komunikace tedy funguje pouze ve stejné podsíti, kde se dá IP adresa serveru zjistit. Tento neduh je řešitelný několika způsoby.

Prvním způsobem je otevřít tunel do stejné sítě, aby bylo možné komunikovat napřímo. Tento způsob není moc pohodlný, protože se nedá měnit dynamicky, to znamená, že buď telefon nebo počítač by museli být pořád ve stejné síti, což není moc dobře možné.

Dalším řešením, ne-li jedním z nejlepších, by bylo vytvořit webový server, který by přeposílal data na dotazy všem zařízením (clientům). Jednalo by se asi o to, vytvořit databázi zařízení (nejlépe podle účtu google, který má většina uživatelů připojený k telefonu a je jedinečný), kterým by se aktualizovala poloha podle toho jak by se nastavilo v aplikaci. Tento způsob by byl dynamický, jelikož IP adresa serveru by byla veřejná, tím pádem by se na něj dalo dotázat z jakékoliv sítě.

Dalším řešením by bylo otevírat spojení až po navázání komunikace pomocí ozvěn. Takzvané `Echo` by napřed zjistilo, zda dané zařízení na daném portu poslouchá. Pokud by komunikace probíhala a na portu by byla data, která by byla podobná těm přenášeným, otevřel by se komunikační socket na tomto portu. Tento způsob je ovšem náročný na síť a zatěžoval by zbytečně i zařízení. Nicméně pokud by tento způsob bylo možné vyladit tak, aby nezatěžoval zbytečně síť i zařízení, nemusela by se vytvářet databáze zařízení, nemuselo by se komunikovat s Google účtem nebo jiným účtem a vše by probíhalo na pozadí.

Oba posledně jmenované způsoby jsou nad rámec zadaní a proto se jim nebylo věnováno intenzivně, jen se vyzkoušelo, zda by mohly fungovat.

5 APLIKACE GPS FINDER

V následujících řádcích bude popsána aplikace GPS Finder, která je výstupem této práce a obsahuje výše popsanou službu. Aplikace má několik funkcí. Jedná se o výstup dat na obrazovku, dále komunikační obsluhu a v neposlední řadě je možné touto aplikací měnit různá systémová nastavení.

Nejprve bude představeno uživatelské prostředí a také se podíváme, jakým stylem je toto prostředí vytvářeno. Nakonec budou popsány všechny funkce a bude také odhaleno pozadí jednotlivých obrazovek.

5.1 Uživatelské prostředí

Při návrhu uživatelského prostředí (dále jen UI) je potřeba přijmout několik zásad, které je dobré dodržovat, jelikož uživatelé jsou již zvyklí na nějaký styl androidích aplikací. Mezi současný standard v oblasti UI, na který by se vývojář měl především zaměřit, je styl Material Design. Tento styl byl vytvořen Googlem a v současné době je rozšířen na OS Android od verze 5.0 a výše. Dále se s ním můžeme setkat na většině webů, které zabezpečuje Google, např. stránky dokumentace o vývoji na Android[4] se dne 17.2.2016 také začaly stylizovat do Material Designu.

5.1.1 Material Design

Když už se nakousnulo téma Material Design, bylo by dobré si k němu specifikovat něco bližšího. Material Design vychází z pár základních pravidel, hlavně co se týče skladby obrazovek, jejich rozměry nebo ikony. Existuje i soubor ikon, které jsou zdarma dostupny zde[9].

Tento styl vznikl hlavně proto, aby vytvořil řád při vývoji nových aplikací. Vychází z geometrie a vše co je součástí Material Designu je geometrické. Vše je stylyzováno do rozměrů telefonů, případně tabletů, tak, aby bylo možné zobrazit plnohodnotnou informaci a stále zachovat nějaký styl. Jistě si dokážete představit, že není úplně jednoduché na malém displeji zobrazit vše co potřebujete. Material Design vám pak nabízí možnost, jak se s tím vypořádat s co nejmenší náročností.

Základní jednotkou rozměru je 1dp (Density-independent pixels neboli dips). Tato jednotka vyjadřuje vztah mezi hustotou (rozlišením) obrazovky a velikostí požadovanou k zobrazení na obrazovce. V praxi to znamená, že každý prvek, který je zadán v jednotkách dp, má na každé obrazovce stejnou velikost v poměru k rozlišení obrazovky. Matematicky vyjádřeno:

$$dp = (width\ in\ pixels * 160)/screen\ density$$

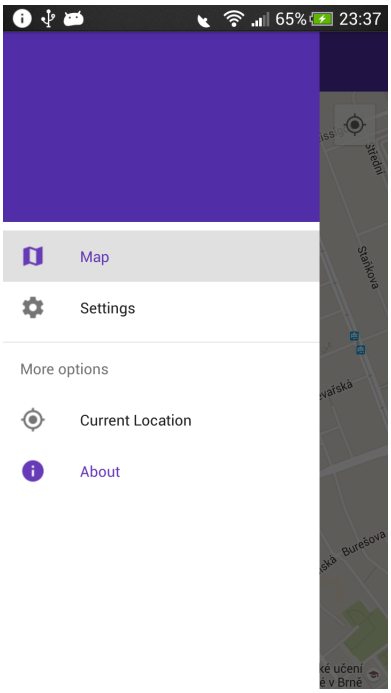
V současné době je většina prvků, které interagují s UI převedena do material designu, takže se vývojář v podstatě nemusí starat o to, zda je něco tak, jako má být.

Stěžejním bodem ve vytváření grafiky aplikace bylo to, navrhnout postranní vyjíždějící menu, tak zvaný `NavigationDrawer`. Tento drawer obsahuje menu, které nabízí přepínání mezi jednotlivými obrazovkami aplikace. Ty budou dopodrobna popsány v následujícím textu.

Návrh draweru je možno uskutečnit pomocí několika stavebních prvků. Jsou tři stěžejní postupy.

Prvním z nich je `NavigationDrawer` pomocí klasického `ListView`, které vychází z Javy. Po kliknutí na položku se vytvoří layout s obrazovkou, která obsadí celý displej. Tento postup je na implementaci nejjednodušší, stačí pouze vytvořit `ListView` typu `String`, které bude součástí draweru, a poté volat jednotlivé položky a nadefinovat jednotlivé obrazovky jako `LinearLayout`.

Druhým způsobem je využití `RecyclerView`, které je v podstatě upravené verze `ListView`. Má několik funkcí navíc, hlavně to, že se do něj dá definovat vlastní typ (třída), takže oproti `ListView` má výhodu v tom, že si jej vývojář může upravit k obrazu svému. Tento způsob navíc může využívat fragmenty na vykreslení obrazovek, není tedy potřeba spouštět novou aktivitu a navíc nepřijdeme o rozdělanou práci v jiném fragmentu.



Obr. 5.1: NavigationDrawer aplikace GPS Finder

Třetím způsobem je ten, který byl využit při konstrukci prostředí. Je to kombinace

widgetu `NavigationView` a fragmentů. Tento widget je již stylizován do Material Designu a není potřeba definovat vlastní prostředí. Navíc umožňuje přidat hlavičku, takže je možné zobrazovat údaje o uživateli, tak jako například v aplikaci Obchod Play. Widget se chová jako menu, tím pádem lze nadefinovat pouze položky menu. Zavoláním funkce `onOptionsItemSelected()`, která je přetižitelná z každé aktivity, se vybere příslušná volba a spustí se fragment, který obsahuje uživatelské prostředí. Výsledný drawer aplikace je na obrázku 5.1. Nezaplněné místo ve vrchní části je hlavička, která není vyplněna. Je zde plánováno propojení s google účtem uživatele, přidáním jeho avataru, e-mailu a jména.

Za zmínku také stojí to, že v současné době je k dispozici velká spousta knihoven třetích stran, které se zabývají touto problematikou.

V současné době, ale není knihovna, která je využita při konstrukci, vyladěna na 100%. Například při vybrání některé z položek se tato položka vybere, ale předchozí vybraná zůstane vysvícena také. Jsou to ale maličkosti, které nemají na funkčnost draweru vliv. Dobře to lze vidět i na obrázku draweru 5.1, kde je vybrána obrazovka Map (na pozadí jde vidět mapa), ale zároveň je vysvícena i možnost About, která aktivní není.

5.1.2 Fragment

Jak bylo již výše popsáno, pro vykreslení prostředí se využívá fragmentů. Fragment je v podstatě nástroj na vytváření uživatelského obsahu, aniž by se musela spouštět nová aktivita. To má mnoho výhod. Jedna aktivita tak může mít například několik obrazovek a přepínat mezi nimi, dokonce může vykreslit několik fragmentů najednou vedle sebe. To je dobré například pro prohledávání souborů či složek, kde na jedné obrazovce se aktualizuje strom složek a vedle něj se zobrazuje obsah nakliknuté složky.

Fragmenty mají také svůj koloběh života jako aktivity. Jejich koloběh života je velmi podobný tomu z aktivit, jsou zde ale menší rozdíly. Fragmenty narozdíl od aktivit mají více abstraktních metod. Jednou z nich je metoda `onCreateView()`, kde fragment vytváří viditelné uživatelské prostředí. Pokud aplikace vytváří více fragmentů, jako v případě tohoto projektu, je možné využít metod `onPause()` a `onResume()` pro pozastavení činnosti fragmentu a jeho následné obnovení bez ztráty průběhu práce v onom fragmentu.

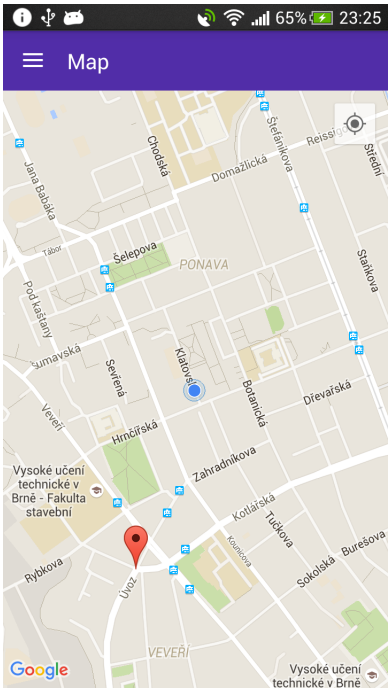
Existují i fragmenty, které jsou spustitelné bez UI. Tyto fragmenty se využívají pro práci na pozadí, kterou není potřeba vytvářet jako službu. Primárně se však fragmenty používají v kombinaci s UI. Vytváření UI je stejné jako v případě aktivity, to znamená pomocí *.xml souboru, s tím rozdílem, že se načítá v metodě `onCreateView()` zmíněné výše.

Spousta knihoven je dnes už dostupná v obou verzích, jak ve verzi aktivity, tak ve verzi fragmentu. Při psaní aplikace bylo využito fragmentů z důvodu použití `NavigationDrawer`, který byl popsán výše.

Následuje popis jednotlivých obrazovek aplikace, kde budou specifikovány hlavní prvky konstrukce kódu.

Map

Na této obrazovce vidí uživatel mapu, která zobrazuje jeho polohu a zároveň, pokud přijímá data od jiného zařízení, zobrazuje polohu druhého zařízení. Mapa je zděděna z typu `SupportMapFragment`, jež je součástí knihovny GoogleMap API, která je dostupná od Googlu zcela zdarma. Tato třída zobrazuje Google mapu, která je totožná s tou, jakou zobrazíte na http://maps.google.com. Označení polohy je řešeno pomocí markeru, který jde vidět na obrázku 5.2. Na stejném obrázku lze i vidět, jak tato obrazovka vypadá.



Obr. 5.2: Obrazovka Map s aktuální polohou zařízení a polohou druhého zařízení

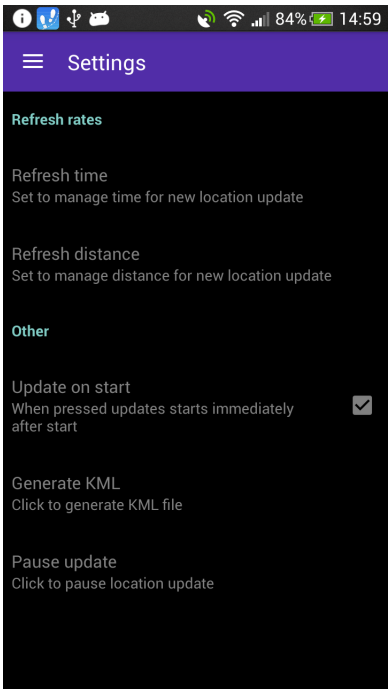
Při stisku ikony v horním pravém rohu, se zobrazí aktuální poloha zařízení, na kterém je spuštěna aplikace. Tento bod je označen modrým kruhem. Knihovna nevyužívá službu GPS_Service, využívá GooglePlayServices jako zjisťovatele polohy. Bohužel v současné době nelze vybírat, co bude tuto mapu aktualizovat. Jediné co se dá ovlivnit je vznik markerů. Ty jsou sdíleny přes síť mezi zařízeními a zobrazovány na jednom z nich. Na mapě se také dají zobrazovat různé informace, které podporuje

i online mapa, například jiné vrstvy apod. Tyto funkce však nejsou úplně dokonale optimalizovány, takže nejsou v projektu využity.

Aby vše fungovalo, tak jak má, na jakékoliv verzi OS, bylo potřeba vše vytvářet z pomocných, support knihoven. Právě u těchto knihoven však není zaručena plná kompatibilita ve všech verzích.

Settings

Obrazovka Settings, jak už název napovídá se stará o nastavení aplikace. Je zde možnost, změnit parametry rychlosti aktualizace. Nabídka Refresh time slouží ke změně času, po kterém se nastaví rychlost aktualizace polohy časovým intervalem (defaultně nastavena na 5s). Refresh distance pak slouží ke stanovení vzdálenosti, za kterou dojde k aktualizaci (defaultně nastaveno na 25m). Možností Update on start se dá nastavit, zda se má začít aktualizovat poloha od startu aplikace nebo až po odškrtnutí této možnosti. Obrazovka vypadá následovně (obrázek 5.3).



Obr. 5.3: Obrazovka Settings

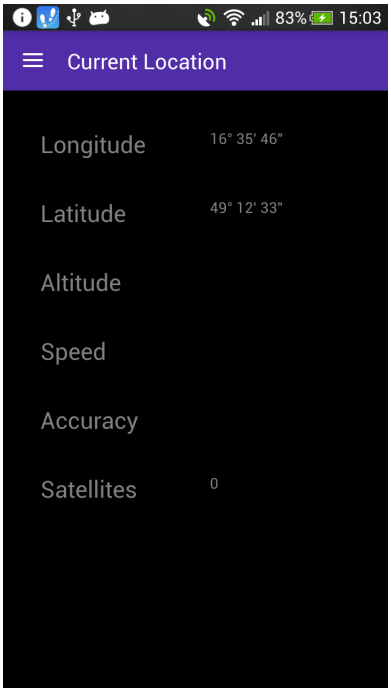
Nastavení je řešeno pomocí dvou tříd. Třída `PreferenceFragmentCompat` se stará o vzhled nastavení, definicí `*.xml` souboru, který zobrazí. Zároveň tato třída dědí z třídy `Fragment`, takže je ji možno volat z rodičovské aktivity.

Druhou, a tou podstatnější třídou, je třída `SharedPreferences` s interfacem `SharedPreferences.OnSharedPreferenceChangeListener`. Tento naslouchač zjišťuje, zda došlo ke změně nějakého z nastavení a pokud je tomu tak, vrátí se v metodě

`onSharedPreferenceChanged()` klíč a hodnota, která se změnila. Klíč je totožný s tím, jaký se vytvoří `*.xml` souboru v třídě `PreferenceFragmentCompat`. Pokud tedy dojde k vyvolání změny z obrazovky, ve vzpomínané metodě se provede následující algoritmus. Otestuje se, zda je na vstupu něco jiného podmínkou `if` a pokud bude hodnota rozdílná, zapíše se do zdrojového souboru. Při dalším spuštění tato hodnota zůstává pořád stejná. Po zjištění, zda je hodnota stejná, se vytvoří editor (`SharedPreferences.Editor`), který funkcí `edit()`, přepíše aktuální hodnotu, za nově získanou hodnotu. Nakonec je potřeba zavolat metodu `commit()`, která změny provede.

Current Location

Obrazovka Current Location, je naprosto stejná jako ta v testovací aplikaci 3.2. Obrazovka byla přidána jako verifikace fungující služby, protože z mapy se nedá přesně určit, zda fungují všechny výpisy. Komunikaci umožňuje `BroadcastRecieveru` jak je popsáno v kapitole 4.2. Pro formu, zde je náhled obrazovky v aplikaci (obrázek 5.4).



Obr. 5.4: Obrazovka Current Location

About

Tato obrazovka slouží jako informace o autorovi a aplikaci samotné. Je zde uvedena fakulta a škola, na které projekt vznikal, jméno vedoucího práce, jméno autora práce

a verze aplikace. Její vzhled jde vidět na následujícím obrázku 5.4

Obr. 5.5: Obrazovka About

5.2 Přístupová práva

Aplikace potřebuje několik přístupových práv pro to, aby mohla využít všechny funkce, které nabízí. Zde je seznam a popis těchto práv:

- `INTERNET` - pro přístup k Internetu (vytvoření socketů)
- `ACCESS_NETWORK_STATE` - právo pro informace ohledně síťových stavů (Internet, aktualizace polohy)
- `WRITE_EXTERNAL_STORAGE` - pro zápis na externí zařízení, například SD karta
- `READ_GSERVICES` - právo pro editaci GoogleMap services (aktualizace mapy)
- `MAPS_RECEIVE` - povolení pro zobrazení mapy
- `ACCESS_COARSE_LOCATION` - stejné jako v testovací aplikaci, přístup k GPS modulu
- `ACCESS_FINE_LOCATION` - stejné jako v testovací aplikaci, přístup k lokaci přes síť

Chybí zde popis umístění těchto práv ve třídách, ale všechna tato práva jsou dohledatelná na vývojářském webu Androidu [4], odkud byl i převzat jejich popis.

5.3 Generování GPX a KML souborů

Generace těchto souborů probíhá tak, že v paměti přístroje je uložen základní soubor, který je nastrukturovaný tak, aby vyhovoval syntaxi, která je stanovena pro tento typ xml souborů. Základní struktura, vypadá takto:

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2"
    xmlns:gx="http://www.google.com/kml/ext/2.2">

    <Placemark>
        <name>gx:altitudeMode Example</name>
        <LookAt>
            <longitude></longitude>
            <latitude></latitude>
            <heading>-60</heading>
            <tilt>70</tilt>
            <range>6300</range>
        </LookAt>
        <LineString>
            <extrude>1</extrude>
            <coordinates>
            </coordinates>
        </LineString>
    </Placemark>
</kml>
```

Tento soubor se následně upravuje pomocí Java DOM Parseru. Tento DOM je nástroj pro práci s xml soubory v Javě. Tento nástroj umí jak upravovat, tak vytvářet nebo mazat xml soubory. Jeho fukce jsou navrženy tak, aby byl schopen vyhledávat pomocí tagů, které jsou součástí xml souborů.

Ukládání záznamů probíhá vždy, když dojde k ukočení služby. Tedy ještě před tím, než k tomu dojde. Záznamy o změnách polohy se nejprve ukládají do proměnné typu `ArrayList<String> locations;`. Tento způsob byl zvolen proto, aby se soubor nemusel otevírat a editovat pokaždé, když dojde k aktualizaci polohy. V některých případech by to znamelo i několikrát za minutu, což není žádoucí, protože otevírání, editace a následné ukládání do souboru zabere relativně hodně času. Navíc by systém využíval více, než by bylo vhodné, systémové prostředky.

Při editaci je také brán zřetel na to, zda sekce, která obsahuje data o poloze, již nějaké záznamy neobsahuje. Pokud ano tak se nejprve prohledá sekce

```
<coordinates>
</coordinates>
```

a všechny záznamy se přeskočí. Na konec těchto záznamů se začnou zapisovat záznamy nové. Pokud by se tato sekce nekontrolovala, docházelo by k tomu, že při novém otevření šablony, by se všechna předchozí měření ztratila. Všechno tohle je kontrolováno funkcí:

```
if ("coordinates".equals(eElement.getNodeName())
&& !eElement.getTextContent().isEmpty()) { }
```

První část podmínky znamená toto. Pokud se aktuální část jmenuje `"coordinates"`, pak pro všechny záznamy v poli, vytvoř nový záznam do souboru. Druhá část je zmíněná kontrola, zda se v této části již nenachází text.

Po přidání všech koordinátu se poslední koordinát uloží ještě do pole:

```
<LookAt>
        <longitude></longitude>
        <latitude></latitude>
        <heading>-60</heading>
        <tilt>70</tilt>
        <range>6300</range>
</LookAt>
```

kde se zajistí to, aby se pohled zaostřil na poslední bod, který byl zjištěň před uložením.

Výchozí soubor, který vznikne editací původní šablony, je uživateli v telefonu skrytý. Je to systémová politika Androidu. Do interní paměti se totiž dá zapisovat pouze tak, že se soubor může být zobrazen pouze v aplikaci, která jej vytvořila. Ostatní aplikace k tomuto souboru nemají přístup. Pro odemčení a zviditelnění souboru vedou dvě cesty. Jedna možnost je, že se soubor zapíše na SD kartu, která je v telefonu přítomna. Pokud není karta přítomna je možné soubor získat pouze přes programové prostředí, jako je Eclipse DDMS. Další možností je odblokování uživatele root v telefonu. Po tomto kroku pak jdou skryté soubory najít.

Další možností by bylo přenášet tento soubor přes Internet, zároveň s komunikací. Tento způsob by byl nejvýhodnější pro uživatele. Tuto funkci se bohužel nepodařilo implementovat do aplikace. Otázkou však zůstává, jestli by tento způsob nezpožďoval komunikaci, která by probíhala podstatně častěji.

Nejvýhodnější variantou by nejspíš bylo to, že by se tento soubor vytvářel až na jiném zařízení vyjma telefonu. Pokud by probíhal přenos na webový server a tyto záznamy se ukládaly do databáze, nebyl by problém zpětně z těchto záznamů vygenerovat KML nebo GPX soubory.

Služba v současném stavu dokáže generovat soubory typu KML. Soubory typu GXP se nepodařilo implementovat na telefonu, takže ani nebylo možné ověřit, zda se tyto soubory generují správně.

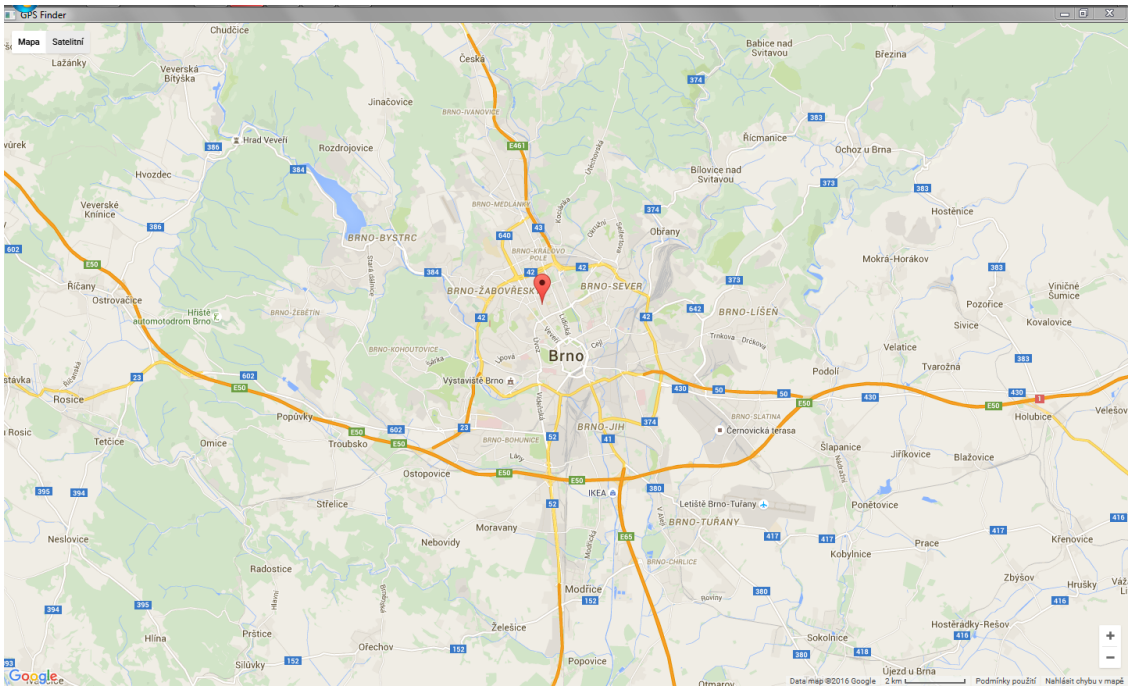
6 APLIKACE GPS FINDER NA PC

Tento PC program je vytvořen k ověření komunikace i na jiné platformy, než je sám Android. Jedná se o velmi jednoduchou aplikaci, která by měla přijímat data z Internetu a zobrazit tak aktuální polohu uživatele, který má u sebe mobilní přístroj. Zároveň by měla aplikace načítat vybraný KML soubor a zobrazit historii pohybu, zachycenou do tohoto souboru.

V této kapitole budou představeny základní prvky stavby programu, představeno uživatelské prostředí a budou popsány některé zajímavé funkce této aplikace.

6.1 Uživatelské prostředí

Uživatelské prostředí je velmi jednoduché v této aplikaci. Žádné v podstatě není. Celé prostředí je jenom Google mapa, která je vytvářena pomocí knihovny GMapsFX. Pro představu jak toto prostředí vypadá, můžete vidět na obrázku 6.1.



Obr. 6.1: Zobrazení mapy po inicializaci a zobrazení protější strany markerem

6.1.1 GMapFX

Knihovna GMapsFX je projekt Roba Terpilowskeho, který je dostupný zde [8]. Jedná se o knihovnu, která vznikla kvůli tomu, že v současné době není od Googlu k dispozici knihovna, která by byla vytvořena pro Javu (JavaFX). K mání je pouze verze

pro JavaScript. Tato knihovna je tedy jakýsi emulátor GoogleMap pro JavaScript na GoogleMapy pro JavaFX. Knihovna plně podpuruje všechny třídy, datové typy a proměnné jako je tomu u GoogleMapy v Androidu. Je zde však rozdíl v incializaci této mapy.

Zatímco mapa v Androidu se aktualizuje pomocí metody `onMapReady()`, která je součástí interface `onMapReadyCallback`, v této knihovně je vytvořen naslouchač, který čeká na inicializování mapy. `MapComponentInitializedListener` se stará o inicializaci metodou `mapInicialized`. V této metodě se dá zobrazovat jak poloha, tak markery nebo třeba celé cesty.

Mapa se vytvoří vznikem nové instance třídy `GoogleMapView`, která vytvoří prostor pro samotnou mapu. Přiřadí se jí listener `MapComponentInitializedListener`, příkazem `addMapInializedListener(listener)`. Vytvořením nové scény, která obsahuje proměnnou typu `GoogleMapView`, a přiřazením této scény do proměnné typu `Stage` se mapa zobrazí. Pokud se má mapa inicializovat, například zde je inicializace vytvořena tak, že se mapa vycentruje po spuštění na centrum Brna, musí se do funkce `mapInicialized` zadat příslušná sekvence. Toto je tvořeno třídou `MapOptions`, která upravuje vzhled mapy podle různých prvků. Těmi mohou být vlastnosti typu zaostření (na určitou vzdálenost), vystředění na určitou pozici a nebo přidáním různých fukcí jako je StreetView. To se dá povolit nebo zakázat. Defaultně je zakázáno.

Po nastavení parametrů se do proměnné typu `GoogleMap`, přiřadí nové okno, které obsahuje mapu, příkazem

```
googleMap = googleMapView.createMap(mapOptions);
```

Tím se vytvoří nová mapa podle zvolených možností.

Pro přidání různých markerů má knihovna stejné nástroje jako knihovna na Androidu. Pomocí tříd `Marker`, `MarkerOptions` se dá vytvořit marker, který zobrazuje polohu protějšku, jak je vidět na obrázku 6.1. Zde je příklad kódu, jak může vytvoření markeru vypadat:

```
MarkerOptions markerOptions = new MarkerOptions();
markerOptions.position(new LatLong(latitude,
   longitude)).visible(BOOLEAN.true).title("Marker");


Marker marker = new Marker(markerOptions);
googleMap.addMarker(marker);
```

Při zadávání souřadnic je zde zvolena instance třídy `LatLong`, která reprezentuje zemskou délku a šířku v jednom. Marker se dá přidá do mapy pomocí funkce `addMarker(marker)`.

Součástí knihovny je také spousta nástrojů pro vytváření cest, zadávání směrů apod. Těmi se bude práce zabývat v následující části, kapitola 6.3.

6.2 Komunikace s mobilním telefonem

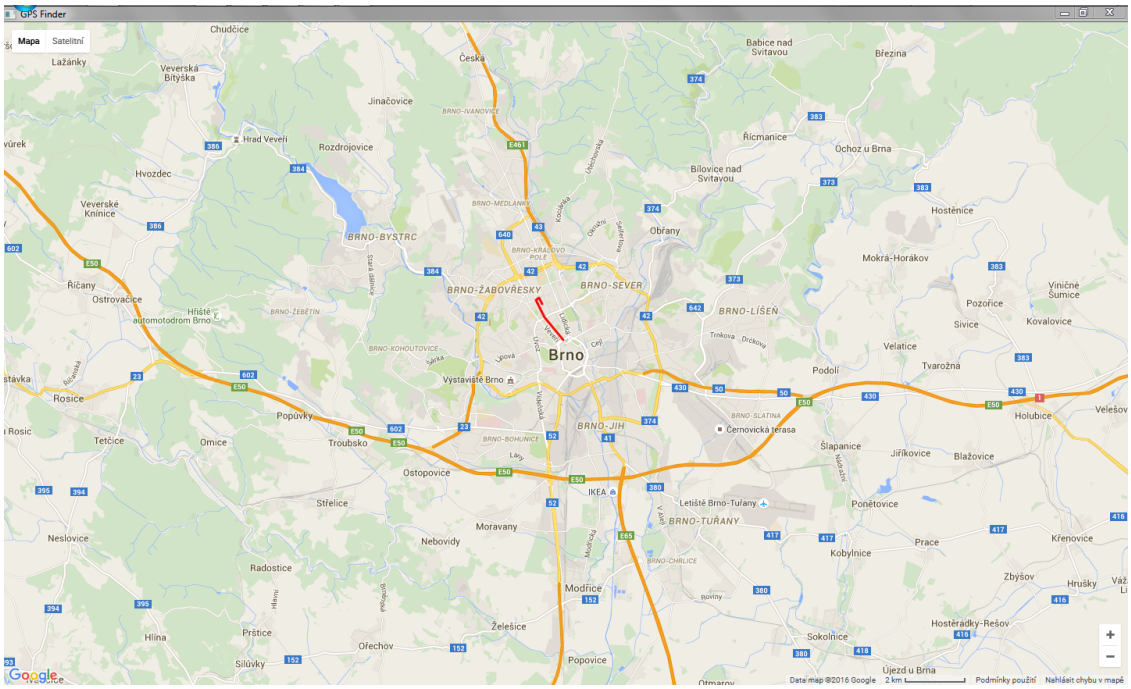
Komunikací se myslí to, že na pozadí služby se přeposílají data o poloze a ty se po přijetí zobrazí na obrazovce počítače. Pro komunikaci je zvolen socket jako na telefonu, hlavně kvůli tomu, že by to jinak ani nemohlo fungovat. Pro příjem je zde třída `ClientSocket`, která dědí ze třídy `Thread` a implementuje interface `Runnable`. Při vytvoření instance třídy se pomocí konstruktoru spustí vlákno a spustí se funkce `run()`. V této funkci je pak komunikace s telefonem. Poloha je posílána pokaždé, když je změněna.

Mobilní telefon pak čeká, dokud mu nepřijde zpráva o přijetí. Po přijetí zprávy se pošle další aktualizace, je-li potřeba. Tento způsob byl hlavně zvolen proto, aby se nestalo, že se nahromadí více aktualizací, než by byl počítač schopen zvládnout. Jak již bylo zmíněno, v současném stavu je vytvořen na mobilním přístroji server na počítači client. Tato varianta byla vybrána hlavně proto, že pro řešení práce, kdy výstup má být hlavně aplikace na Android, se jevil tento postup jako správný. Obrácená varianta nebyla uvažována, tudíž ani zkoušena.

6.3 Zobrazení KML souborů

Pro zobrazování KML souborů je využita stejná knihovna, jako pro jejich zápis. DOM Pharser vyexportuje ze souboru KML všechny potřebné údaje. Syntaxe je velmi podobná. Místo editace dat se však načtou data z tohoto souboru a vytvoří se cesta, která odpovídá bodům specifikovaným v sekci `coordinates`. Ze všech těchto bodů se pak vytvoří polygon, který odpovídá zobrazované cestě. Cesta se zobrazí na mapě jak je vidět na obrázku 6.2.
Jelikož data, které vygeneruje KML soubor jsou typu String, musí se převést na jednotlivé souřadnice. Ve Stringu je řetězec čísel oddělených čárkou a znakem Enter. Tyto znaky se musí odfiltrovat. Byl zvolen způsob převedení Stringu na pole charů. Z tohoto pole se pak cyklem `for` vybírají všechny znaky, které tam nepatří. Zbytek se zapisuje do `ArrayList<Double>`, kde jsou již hotové souřadnice typu Double. Ty jsou řazeny tak, že na každém sudém indexu je zemská délka (včetně nuly) a na každém lichím indexu je zemská šířka. Tento list se pak převede do pole `LatLong[]`, aby bylo možné zobrazit souřadnice na mapě.

Z těchto dat se pak generuje výše zmíněný polygon, pomocí třídy `Polyline`. Konstruktor této třídy vytváří tento tvar pomocí `PolylineOptions`. Do této instance



Obr. 6.2: Zobrazení KML souboru v mapě pomocí polygonu

se také vkládá seznam bodů, kterými má tento tvar projít.

Jednou z možností, nad kterou bylo uvažováno, byla ta, využít knihovnu Java API for KML. Tato knihovna umí sama převést souory KML do typu Object. S těmi se pak dá nakládat jako s jakýmkoliv jiným objektem v Javě.

7 TESTOVÁNÍ

Testování probíhalo, stejně jako u testovací aplikace, na stejném hardwaru a to na HTC One $X+$ s verzí Androidu 4.2.2 Jelly Bean a úpravou HTC Sense 5.0. Testování probíhalo necelý týden. Při testování se hlavně ověřovalo, zda tefefon dokáže být nabitý celý den, a zda aplikace nezatěžuje systém tak moc, až by to bylo neúnosné.

Při testu se neprojevilo závažné zatížení systému jako celku. Služba běžela necelých 6 hodin denně na jedno nabití baterie. Pro srovnání baterie v testovacím telefonu má kapacitu 1800mAh. Na telefonu souběžně probíhalo normální zatížení, to znamená přibližně 30 min hovorů a 5 SMS za zmíněnou dobu. Průběžně bylo zkoušeno, zda aplikace nepadá do nějaké chyby, například přepínáním obrazovek v postraním menu.

Také bylo ověřeno, zda příslušná nastavení ovlivňují službu a zda tyto změny jsou patrné. Při nastavení malých intervalů byla spotřeba vyšší, naopak při nastavení vyšších intervalů aktualizace se spotřeba mírně snížila. Subjektivně se dá tvrdit, že aplikace má přibližně stejnou spotřebu energie, jako navigace od Googlu. Při stejných podmínkách (cesta autem z Brna do Prahy a zpět, jednou s navigací, jednou s aplikací na pozadí) se telefon dostal na kritickou hranici (14% baterie, nastaveno výrobcem) přibližně ve stejném okamžiku. Tato situace nastávala od plného nabití telefonu.

Pro srovnání byl test uskutečněn i na telefonu jiného výrobce i jiné verzi OS. Jedná se o mobilní přístroj značky LG, model F60 D390N s kapacitou baterie 2100mAh a verzí operačního systému Android 5.0 Lollipop. Tento telefon při běžném provozu (volání a SMS) vydrží nabitý necelých 5 dní. Při spuštěné aplikaci ve stejných dobách, tedy současně s prvně jmenovaným přístrojem, vydržel tento telefon 13 hodin provozu. Plné srovnání s navigací nebylo u tohoto telefonu možné, jelikož nemá datové připojení, tudíž srovnání by bylo zkreslené.

Při testování vyvstaly na povrch některé nedokonalosti, které se nepodařilo ošetřit. Například při vybraném NETWORK provideru lokace, se aktualizuje pouze první a druhá položka v obrazovce Current Location (obrázek 5.4). Tento výběr by se dal ovlivnit, ale není nutné jej aplikovat, jelikož veškerá komunikace na pozadí užívá pouze zemskou dělku a šířku k zobrazení informací o zařízení. Ostatní veličiny jsou zanedbatelné pro určování polohy, vyjma ukládání záznamů do KML souborů.

Důležitým testem také bylo to, zda aplikace neruší funkčnost mobilního telefonu. Pokud by například docházelo k tomu, že z nedostatku sytémových prostředků nemohou probíhat hovory, nebyl by tento počin moc vhodný. Toto bylo otestováno pomocí probíhajích hovorů s na pozadí běžící aplikací.

Testování komunikace na jiná zařízení probíhalo na zmíněných dvou telefonech mezi sebou. Test také proběhl při komunikaci na PC. Mezi přístroji nevznikal pro-

blém při komunikaci, vše probíhalo tak jak by mělo. Při každé změně polohy jednoho nebo druhého telefonu se odeslala data o poloze na zvolený port a na tomto portu se data přijala. Při kontrole sítě pomocí Wiresharku bylo vidět, že mezi telefony probíhá přenos bez problému. Údaje o poloze nebyly čitelné, ale bylo vidět, že se neodesílají prázdné pakety.

Při komunikaci s počítačem někdy docházelo k výpadkům, hlavně proto, že při pozastavení aplikace (služba byla stále aktivní) docházelo z nevysvětlitelných důvodů k ukončení komunikace. Tento problém se nepodařilo ošetřit, protože se nepodařilo zjistit proč tento problém vůbec vzniká. Možným řešením tohoto problému by mohlo být vytvoření serveru na počítači místo na telefonu.

Dalo by se říct, že testování aplikace dopadlo úspěšně, co se týče části s Androidem. U testování programu na PC se to říct se stoprocentní jistotou říct nedá, už jen kvůli nevysvětlitelnému ukončování komunikace. Veškeré problémy s komunikací by se měli vyřešit přidáním webového serveru, ke kterému by byly připojeny zařízení neustále a nemohlo by dojít k přerušení ani výpadku serveru. Tento postup byl navrhován již v předchozích kapitolách (4.3).

8 ZÁVĚR

Cílem této diplomové práce bylo vytvořit samostatnou službu, která by měla zjišťovat polohu mobilního telefonu s operačním systémem Android. Zároveň bylo zapotřebí, vytvořit spouštěcí aplikaci, která by tuto službu mohla ovládat. Výsledkem práce je služba GPS_Service, která zjišťuje polohu v GPS lokátoru a zároveň ji zjišťuje i ze síťových prostředků. Pokud je jeden ze dvou zprostředkovatelů nedostupný, služba vybere, který z nich to je a na dobu, kdy je nedostupný, jej odebere z aktualizování. Zároveň implementuje síťový socket, který se stará o komunikaci s venkovní sítí. Port socketu je nastaven na 2001. Služba je vytvořena jako vázaná na aktivitu, aby bylo zajištěno správné chování služby na pozadí.

Komunikace je typu Server-Client a je řešena pomocí socketů. Při každém obnovení polohy se odešle záznam do sítě. Při komunikaci s počítačem se občas stávalo, že spojení vypadlo. Nepodařilo se získat původce tohoto jevu, tím pádem je toto nedořešeno. Tato komunikace probíhala pouze na lokální podsíti.

Aplikace, která vytvořenou službu ovládá se nazývá GPS Finder. Tato aplikace je vytvořena pomocí Material Designu a implementuje nejnovější postupy při návrhu vzhledu aplikací. Při vytváření aplikace bylo využito knihovny GoogleMap API, která zobrazuje mapový základ, na kterém jde vidět pohyb uživatele. Při přijímaní dat ze sítě, může zobrazovat i polohu druhého zařízení pomocí markerů. Aplikace také umožňuje změnit některá nastavení služby, jako jsou změna času, za který dojde k aktualizaci, nebo vzdálenost, po které dojde k obnově polohy. Aplikace také umí generovat soubory typu KML, které zaznamenávají historii pohybu mobilního telefonu.

Dalším výstupem práce je jednoduchá aplikace na počítač, která zobrazuje získanou polohu ze sítě od mobilního telefonu. Zároveň slouží jako zobrazovač vygenerovaných KML souborů. Tato aplikace se jmenuje stejně jako aplikace na Android, tedy GPS Finder. Využívá knihovnu GMapFX, která převádí GoogleMap API pro JavaScript na knihovnu využitelnou v Java.

Celý projekt byl otestován na dvou mobilních zařízeních od různých výrobců. Tím byla srovnána náročnost aplikace, jelikož oba mobilní přístroje jsou na opačných koncích výkonových stupnic. Při testování se neprojevila žádná zásadní disfunkce aplikace. Výsledkem testování byla výdrž baterie zařízení se spuštěnou službou. I tento test dopadl s dobrými výsledky. Aplikace na pozadí běžela okolo 6 hodin od plného nabití přístroje, až po kritickou hranici (14% energie) baterie.

Aplikace i služba by se mohly dále rozvíjet. Například, jak již bylo uváděno, vylepšení komunikace mezi zařízeními vytvořením webového serveru. Tento server by mohl obsahovat databázi uživatelů a jejich přístrojů. Zároveň by zaznamenával údaje o poloze a na vyžádání jiných uživatelů, by je jim přeposílal. Současně s tím by

se mohla vylepšit dostupnoust KML souboru. Tento soubor by se nemusel vytvářet na telefonu, ale až na vyžádání na zmíněném serveru.

Zlepšení by si také zasloužila aplikace na počítači. V současné době nenabízí skoro žádné možnosti pro uživatele. Jedná se pouze o stejnou GoogleMapu, která je dostupná také na webu.

LITERATURA

[1] *Americký družicový navigační systém NAVSTAR GPS - Český kosmický portál - Odbor ITS, kosmických aktivit VaVal* [online]. 2010, poslední aktualizace 9. 12. 2015 [cit. 15. 12. 2015]. Dostupné z URL: <http://www.czechspaceportal.cz/š-sekce/gnss-systemy/gnss-mimo-evropu/americky-navstar-gps/>.

[2] *Český kosmický portál - Odbor ITS, kosmických aktivit VaVal* [online]. 2010, poslední aktualizace 9. 12. 2015 [cit. 15. 12. 2015]. Dostupné z URL: <http://www.czechspaceportal.cz>.

[3] *Android* [online]. 2005, poslední aktualizace 15. 12. 2015 [cit. 15. 12. 2015]. Dostupné z URL: <http://www.android.com>.

[4] *Android Developers* [online]. 2008, poslední aktualizace 15. 12. 2015 [cit. 15. 12. 2015]. Dostupné z URL: <http://developer.android.com/index.html>.

[5] MURPHY, Mark L. *Android 2: Průvodce programováním mobilních aplikací*. 1. vyd. Brno, Computer Press, 2011. 375 s. ISBN 80-2513-194-7

[6] *Location Strategies | Android Developers* [online]. 2008, poslední aktualizace 15. 12. 2015 [cit. 15. 12. 2015]. Dostupné z URL: <http://developer.android.com/guide/topics/location/strategies.html>.

[7] *Services | Android Developers* [online]. 2008, poslední aktualizace 20. 03. 2016 [cit. 15. 05. 2016]. Dostupné z URL: <http://developer.android.com/guide/components/services.html>.

[8] *GMapsFX* [online]. 2016, poslední aktualizace 17. 05. 2016 [cit. 20. 05. 2016]. Dostupné z URL: <https://github.com/rterp/GMapsFX>.

[9] *Material Icons - Google Design* [online]. 2012, poslední aktualizace 17. 05. 2016 [cit. 20. 05. 2016]. Dostupné z URL: <https://design.google.com/icons/>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

API	Application programming interface
dp	Density-independent pixels
DVM	Dalvik Virtual Machine
GLONASS	Globalnaya navigatsionnaya sputnikovaya sistema
GPS	Glogal Positioning System
GPX	GPS Exchange Format
HDML	Handheld Device Markup Language
JDBC	Java Database Connectivity
KML	Keyhole Markup Language
LORAN	Long Range Navigation
NDK	Native Development Kit
OS	Operační systém
SMS	Short Message Service
SDK	Software Development Kit
UI	User Interface (Uživatelské prostředí)

SEZNAM PŘÍLOH

A Zdrojové kódy aplikací **57**

A ZDROJOVÉ KÓDY APLIKACÍ

Zdrojové kódy aplikací jsou dostupny v systému Appolo a na přiloženém CD.