



TECHNICKÁ UNIVERZITA V LIBERCI  
Fakulta mechatroniky, informatiky  
a mezioborových studií ■

# ANDROID APLIKACE PRO PŘESNÝ ZÁZNAM POLOHY

## Diplomová práce

*Studijní program:* N2612 – Elektrotechnika a informatika

*Studijní obor:* 1802T007 – Informační technologie

*Autor práce:* **Bc. Radek Luňák**

*Vedoucí práce:* Mgr. Jiří Vraný, Ph.D.



## Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

## **Poděkování**

Tímto bych chtěl poděkovat vedoucímu mé diplomové práce panu Mgr. Jiřímu Vranému za konzultace, užitečné rady k řešení, i za zapůjčení mobilního telefonu v době, kdy jsem neměl ještě vlastní se systémem Android. Firmě Jablocom děkuji za pomoc s vytvořením zadání a konzultace. Také bych chtěl poděkovat své rodině za umožnění studia a za jejich podporu.

## **Abstrakt**

Tato práce se zabývá návržením a vytvořením trackovacího systému, který získává polohu mobilních klientů s platformou Android a shromažďuje je v rámci webové služby. První část práce popisuje dostupné zdroje v systému Androidu, které mohou být použity k získání polohy. Součástí popisu těchto zdrojů je jejich úroveň přesnosti a spotřeby energie. Další část rešerše popisuje framework Grails, který byl použit k vytvoření webové služby. Hlavní část práce se zabývá návrhem a konkrétní realizací systému a odůvodněním, proč senzory jako akcelerometr nejsou vhodné k použití v takovémto druhu systému. V závěru práce je zde porovnání mezi jednotlivými nastaveními aplikace a návrhy, jak by mohl být současný systém vylepšen.

**Klíčová slova:** Trackovací systém, GPS, WPS, Android, Grails

## **Abstract**

This thesis is about design and creation of tracking system, which acquires locations of mobile clients with Android platform and collects them within web service. First part of the thesis describes available sources in system Android, which can be used to obtain location. Part of description of these sources is their level of accuracy and power consumption. Another part of research describes framework Grails, which was used to create web service. Main part of thesis consists of design and actual realization of the system and reason why sensors such as accelerometer is not much suited to use for this kind of system. At the end of thesis there is comparison between individual settings of application and suggestions how current system can be improved.

**Keywords:** Tracking system, GPS, WPS, Android, Grails

# Obsah

Poděkování.....	4
Abstrakt.....	5
Abstract.....	5
Úvod.....	8
1 Senzory.....	9
2 Techniky určení polohy.....	11
2.1 GPS.....	12
2.1.1 Princip .....	12
2.1.2 Techniky zpřesnění GPS.....	14
2.2 WPS.....	15
2.3 Použití senzorů.....	16
3 Použité nástroje a zařízení .....	18
3.1 Grails.....	18
4 Cíle práce a návrh řešení.....	21
4.1 Cíle práce.....	21
4.2 Návrh systému.....	21
4.2.1 Použití akcelerometru.....	23
5 Mobilní aplikace.....	24
5.1 Popis struktury aplikace.....	24
5.1.1 Ukládání uživatelských nastavení.....	25
5.2 Komunikace se serverem.....	26
5.3 Ukládání dat.....	26
5.4 Algoritmus hledání polohy.....	27
5.5 Alarmy.....	29
6 Webová služba.....	31
6.1 Platforma.....	31
6.2 Komunikace .....	31
6.2.1 Formát objektu JSON.....	32
6.3 Databáze.....	32
6.4 Uživatelské prostředí.....	33
7 Srovnání různých nastavení aplikace.....	35
8 Závěr.....	38
Seznam použité literatury.....	40
A Obsah přiloženého CD.....	42

## Seznam ilustrací

Ilustrace 1: Popis určení os u zařízení [1].....	9
Ilustrace 2: Spotřeba služeb v zařízení [4].....	11
Ilustrace 3: Rozložení 24 satelitů [17].....	12
Ilustrace 4: Vícecestné šíření signálu.....	13
Ilustrace 5: Schéma funkce DGPS.....	14
Ilustrace 6: Princip WPS.....	16
Ilustrace 7: Chyba akcelerometru [6].....	17
Ilustrace 8: Spotřeba akcelerometru [4].....	17
Ilustrace 9: HTC One SV.....	18
Ilustrace 10: Ukázka doménové třídy v Grails.....	19
Ilustrace 11: Návrh systému.....	22
Ilustrace 12: Class diagram.....	24
Ilustrace 13: Ukázka logu neodeslaných lokací.....	25
Ilustrace 14: Ukázka hlavní obrazovky mobilní aplikace.....	25
Ilustrace 15: Ukázka aktivity nastavení.....	26
Ilustrace 16: Průběh startu služby.....	27
Ilustrace 17: Průběh testu nové lokace.....	28
Ilustrace 18: Ukázka objektu JSON.....	32
Ilustrace 19: ERD.....	33
Ilustrace 20: Úvodní strana webové služby.....	33
Ilustrace 21: Ukázka seznamu lokací .....	34
Ilustrace 22: Historie spotřeby při různých nastaveních.....	35
Ilustrace 23: Graf porovnání spotřeby různých nastavení.....	36

## Úvod

Mobilní telefony se staly nedílnou součástí našeho života, stávají se čím dál výkonnějšími a poskytují více a více funkcí. Už nějakou dobu neplatí, že telefon se používá pouze k volání, spíše se jedná o takové multifunkční zařízení, které může sloužit k práci i zábavě. Určování pozice se stalo často požadovanou funkcí a můžeme ji řadit do obou kategorií. Většina dnešních mobilních telefonů obsahuje podporu připojení k internetu pomocí Wifi či datového spojení, GPS čip a další senzory udávající informace o okolí. Díky těmto vylepšením získaly telefony více možností, odkud lze získat data o poloze oproti původnímu zdroji z GSM sítě, které spíše určuje pouze oblast. Pokud tato data budeme získávat průběžně, získáme představu o tom, kde se uživatel pohybuje v průběhu dne. Ponecháme-li stranou etičnost tohoto řešení, hlavní motivací je využitelnost této funkce například pro rodiče, kteří mají starost o to, kde se nachází jejich dítě, či pro zaměstnavatele doručovací služby, který chce mít přehled o svých zaměstnancích v terénu. Tato práce pojednává o návrhu a tvorbě takového systému pro platformu Android. Operační systém Android je dnes nejrozšířenějším mobilním systémem a vzniká pro něj mnoho aplikací každým dnem. Zadání vzniklo společně s firmou Jablocom, která měla zájem blíže prozkoumat možnosti tohoto řešení.

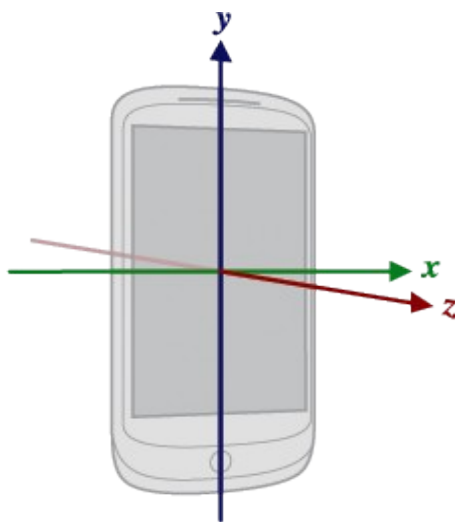
Úkolem práce bylo vytvořit trackovací systém, ve kterém bude mobilní aplikace sledovat svojí polohu pomocí dostupných prostředků a odesílat naměřená data příslušné webové službě. Mobilní aplikace by neměla výrazně ovlivňovat výdrž baterie. Součástí práce bylo také prostudování dostupných prostředků pro zjištění polohy, možnosti jejího zpřesnění a jejich shrnutí. Následně na základě získaných znalostí tyto metody implementovat v mobilní aplikaci. Dalším bodem práce bylo navrhnout a vytvořit webovou službu s komunikačním rozhraním, která bude přijímat data od klientů, archivovat je a následně je přehledným způsobem zobrazovat.

Téma diplomové práce jsem si zvolil z důvodu, že mě problematika určování polohy zajímá i z rekreačních důvodů, jako je například geocaching. Další motivací pro mne bylo, že s vývojem aplikace pro platformu Android jsem se dosud setkal pouze povrchně a tato práce tedy byla příležitostí prohloubit si znalosti ohledně tohoto operačního systému a programování pro něj.

# 1 Senzory

Většina dnešních mobilních telefonů se řadí pod označení smartphone. Smartphone je označení pro telefon, který kombinuje funkce mnoha zařízení, jako například PDA, přehrávače, GPS přijímače či kamery. Používají operační systém a podporují Wifi připojení, webový prohlížeč a jiné funkce. Mimo zmíněných funkcí obsahují tato zařízení také řadu senzorů poskytující informace o okolí a stavu, v němž se zařízení nachází. Tyto senzory dokáží měřit pohyb, orientaci a různé stavy prostředí. To může být užitečné, pokud je třeba sledovat pohyb nebo pozici zařízení ve třech dimenzích, či je nutné reagovat na změny prostředí v okolí zařízení. Senzory lze rozdělit podle [1] do tří skupin na pohybové senzory, senzory prostředí a poziční senzory.

Pohybové senzory měří zrychlení a rotaci ve třech osách. Do této skupiny se řadí akcelerometr, gravitační senzor, gyroskop a senzory vektorů rotace. Další skupina se zaměřuje na parametry prostředí. Tyto senzory dokáží tedy snímat intenzitu okolního osvětlení, teplotu, tlak či vlhkost. Řadí se mezi ně teploměr, tlakoměr a luxmetr. Poslední skupina se nazývá poziční a měří fyzickou pozici zařízení. Do této kategorie spadá magnetometr a orientační senzory.



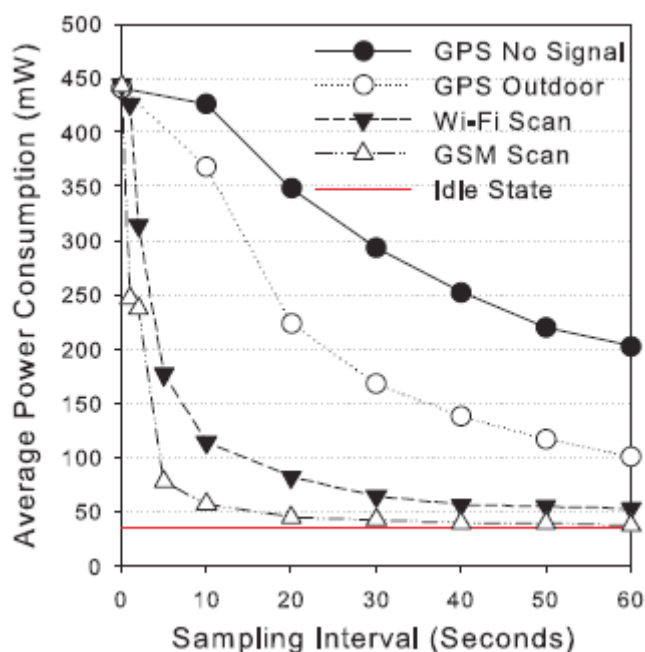
*Ilustrace 1: Popis určení os u zařízení [1]*



System Android poskytuje pro práci se senzory framework, který obsahuje několik tříd a rozhraní pomáhající vykonávat širokou škálu úkolů. Ve většině případů, pokud chci využívat nějaký senzor, dosáhnu toho registrováním naslouchání jeho událostí a poté v jejich obsluze je mohu dále zpracovávat. Před použitím senzoru je doporučeno nejprve otestovat, zda je dostupný. Při používání senzorů je třeba být obezřetný s nastavením intervalu, jak často chci získávat nová měření, jelikož nastavení nízkého intervalu způsobí značné čerpání energie baterie. Nakonec při ukončení práce se senzorem by mělo být naslouchání odregistrováno. V současné době nelze senzory testovat na emulátoru a je proto nutné mít fyzické zařízení nebo simulovat výstup senzoru.

## 2 Techniky určení polohy

Systém Android nabízí několik možností, kterými lze získat údaje o poloze zařízení. Tato služba je souhrnně označována jako Location API a rozlišuje tři poskytovatele lokace, které nesou označení podle svého zdroje. GPS poskytovatel využívá stejnojmennou službu, síťový poskytovatel získává data z WPS a z GSM sítě od BTS stanic. Poslední poskytovatel, zvaný pasivní, používá data o poloze, které získají ostatní spuštěné aplikace na zařízení.



*Ilustrace 2: Spotřeba služeb v zařízení [4]*

Ilustrace 2 zobrazuje spotřebu jednotlivých služeb v zařízení pro získání polohy. Dle očekávání dosahuje nejvyšší spotřeby služba GPS, hlavně pokud nemá signál. Druhou službou z pohledu spotřeby je skenování Wifi sítí. Nejméně náročné je skenování GSM sítě, které nepřináší navíc žádnou spotřebu energie, jelikož se provádí v zařízení automaticky a je samozřejmostí.

## 2.1 GPS

GPS neboli Global Positioning System je globální navigační systém skládající se z 24 satelitů obíhajících na orbitě Země, které zde umístilo Ministerstvo obrany Spojených států amerických. Původně mělo GPS vojenský účel, ale v roce 1980 umožnila vláda i civilní použití tohoto systému. Tato služba je dostupná za jakýkoliv meteorologických podmínek, kdekoli na světě a to po celý den bez poplatků.

### 2.1.1 Princip

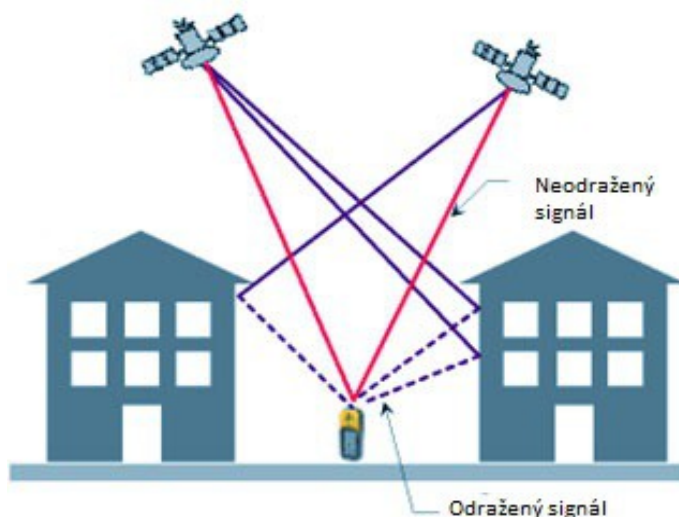
GPS satelity oběhnou Zemi dvakrát za den po velmi přesné dráze a vysílají signál s informacemi na Zemi. Přijímač zpracuje tyto informace a pomocí triangulace určí přesnou pozici uživatele. Výpočet probíhá porovnáním času, kdy byl signál odvysílán s časem, kdy byl přijat. Podle rozdílu časů přijímač určí, jak daleko se satelit nachází. Při získání několika vzdáleností od různých satelitů je přijímač schopen spočítat polohu uživatele. Minimální počet satelitů potřebných pro výpočet je tři, se čtyřmi pak lze získat i nadmořskou výšku. Analogicky platí že, čím je více dostupných satelitů, tím je určení polohy přesnější.



*Ilustrace 3: Rozložení 24 satelitů [17]*

Satelity vysílají dva radiové signály označené jako L1 a L2. Civilní GPS používá L1 na frekvenci 1575.42 Mhz v pásmu UHF. Signál dokáže projít skrz mraky, sklo a plast, ale neprojde skrz většinu pevných objektů jako například budovy či hory. Signál obsahuje pseudonáhodný kód, data efemerid a almanachu. Pseudonáhodný kód je jednoduše identifikační kód, který určuje, jaký satelit vysílá. Termín efemerida se používá pro software, který dokáže generovat pozice planet, satelitů, asteroidů a komet v libovolném čase, o který uživatel zažádá. Dále satelity neustále vysílají svou orbitální

pozici, data o svém stavu, datum a čas. Tato část je důležitá pro určování pozice. Data almanachu říkají, kde by se měl každý satelit nacházet v průběhu dne. Tato data jsou jednotná a poměrně obsahlá, proto jejich přenos zabere déle času. Je možné je stáhnout i z jiného zdroje, nejčastěji z internetu, čehož využívá například A-GPS pro rychlejší zjištění polohy po startu zařízení.



*Ilustrace 4: Vícecestné šíření signálu*

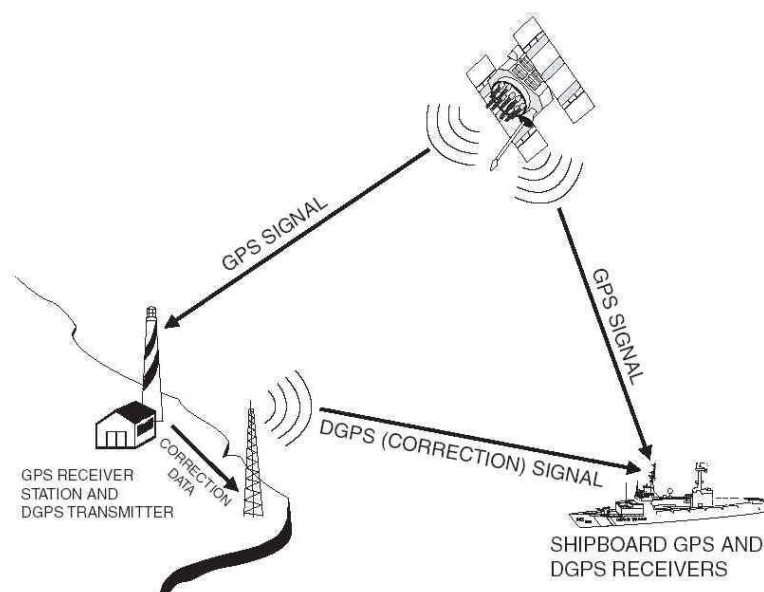
Existuje mnoho faktorů, které snižují kvalitu signálu, a tudíž i přesnost. Nejprve se jedná o zpoždění při průchodu ionosférou a troposférou. Systém používá vestavěný model, který počítá průměrné zpoždění, a ten tuto chybu částečně odstraňuje. Dalším faktorem je vícecestné šíření signálu. To je způsobené odrazem od různých objektů, například budov, což způsobuje prodloužení doby cesty signálu. Další chybu mohou způsobit hodiny přijímače, které nejsou tak přesné jako atomové hodiny na satelitu. Chyba může být způsobena také nedostatkem viditelných satelitů nebo jejich postavením vůči sobě. Do roku 2000 byl také signál úmyslně zhoršován Ministerstvem obrany Spojených států amerických.

Systém GPS není ojedinělé řešení a existuje několik podobných systémů. Příkladem je ruský GLONASS nebo připravované evropské Galileo. Tyto systémy se označují souhrnným termínem globální navigační satelitní systém, ve zkratce GNSS. V rámci použití v systému Android to nehraje roli. V Android frameworku pro lokaci se tyto systémy řadí pod poskytovatele GPS a my se nemusíme starat o to, ze kterého systému získáváme data. Pokud by byla potřeba tuto informaci zjistit, je možné ji získat určenou metodou.

## 2.1.2 Techniky zpřesnění GPS

Pokud by bylo potřeba získat ještě lepší přesnost než dosahuje samostatně GPS, které se pohybuje okolo deseti až patnácti metrů, existuje řada způsobů, jak toho docílit. Většinou se jedná o dopočet polohy za využití údajů z jiných zdrojů. Rozlišujeme dva základní druhy zdrojů, a to lokální GBAS a satelitní SBAS. Do lokálních se řadí zejména DGPS a satelitních například evropský EGNOS či americký WAAS.

Diferenciální GPS se označuje systém, ve kterém jsou korekční data posílána pozemní stanicí, u které je přesně známa poloha. Korekční data jsou počítána na základě odchylky polohy stanice od skutečné. Přijímač musí být do vzdálenosti 1000 km a systém dosahuje přesnost zhruba dva až pět metrů. Korekce lze využívat v reálném čase nebo mohou být dopočítány i zpětně. Tyto služby bývají zpoplatněny.



*Ilustrace 5: Schéma funkce DGPS*

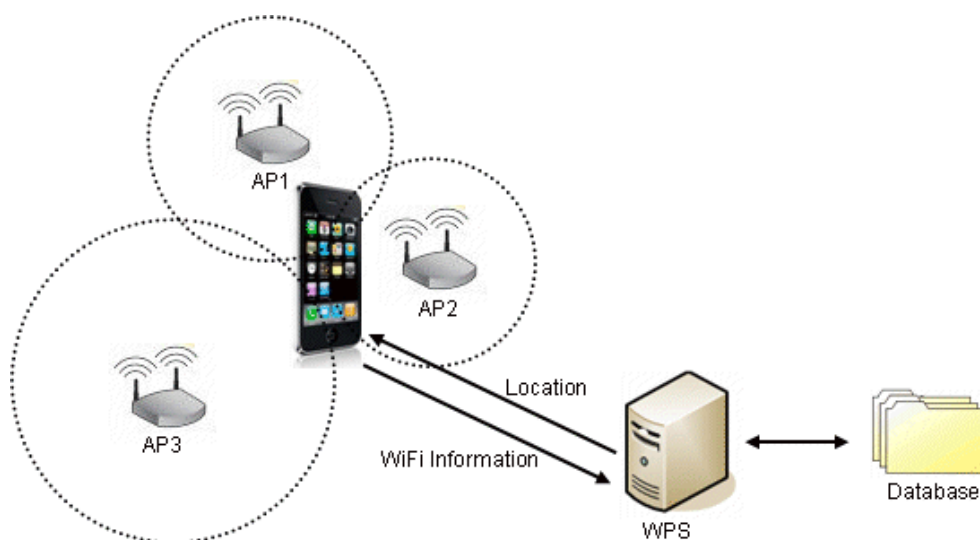
U SBAS jsou diferenciální korekce zasílány ze satelitů, které jsou nasměrovány přímo na oblast, pro kterou je tato služba určena. Hlavními regionálními představiteli jsou WAAS a EGNOS a jsou volně dostupné. EGNOS poskytuje SDK pro Android, ale pro výpočet jsou potřeba neupravená data přímo z čipu GPS, což není v oficiálním API Androidu dostupné, a toto použití vyžaduje externí anténu připojenou přes bluetooth.

Další placenou službou je RTK neboli real-time kinematic. Také jako ostatní využívá jednu základní stanici a je s přesností pod jeden metr nejpřesnější ze všech GPS systémů, ale také zřejmě nejvíce finančně náročný. Tento systém se využívá v topografii, při navigaci na moři a například také při automatickém řízení v zemědělství. Stanice přenáší stejně jako u jiných korekční data, ale navíc ještě přidává informace o nosné fázi, která je k dosažení této přesnosti rozhodující.

Můžeme sem řadit i A-GPS, které zejména pomáhá s prvotním určením pozice díky stažení dat almanachu z internetu. Proto je k použití nutné mít prostřednictvím zařízení přístup k internetu. Z hlediska spotřeby by použití těchto metod zvýšilo energetickou náročnost kvůli nutnosti komunikace pro získání korekčních kódů.

## **2.2 WPS**

WPS neboli Wifi positioning system je služba získávající údaje o poloze prostřednictvím Wifi sítě. Motivací pro vznik tohoto systému bylo poskytnout možnost získat polohu zařízením, které poskytují příjem Wifi a neobsahují modul GPS, a také poskytnout případně přesnější pozici uvnitř budov či v dalších místech, kde se GPS potýká s problémy technologického rázu. Princip spočívá v databázi pozic přístupových bodů Wifi. Poloha zařízení je určena na základě SSID sítě a intenzity signálu. Tato metoda se nazývá *fingerprinting*. Tento systém byl původně prosazen firmou Skyhook, nicméně Google, Apple a další výrobci telefonů si vytvořili své vlastní rozsáhlé databáze s pozicemi přístupových bodů. Tato data se získávají ze zařízení, které určí pozici pomocí GPS a odešle seznam SSID. Například [14] uvádí, že pro Google původně získávala data auta vytvářející Street View, ale dnes i společně s ostatními společnostmi jako Apple či Microsoft je získávají anonymně přímo od koncových uživatelů. Stačí povolit Služby zjištění polohy Google na Android zařízení a od tohoto okamžiku je zařízení součástí sběru dat pro WPS. Google dále vytvořil možnost pro přístupové body, aby nebyly zahrnuty do databáze. Toho lze docílit přidáním prefixu `_nomap` před název sítě.

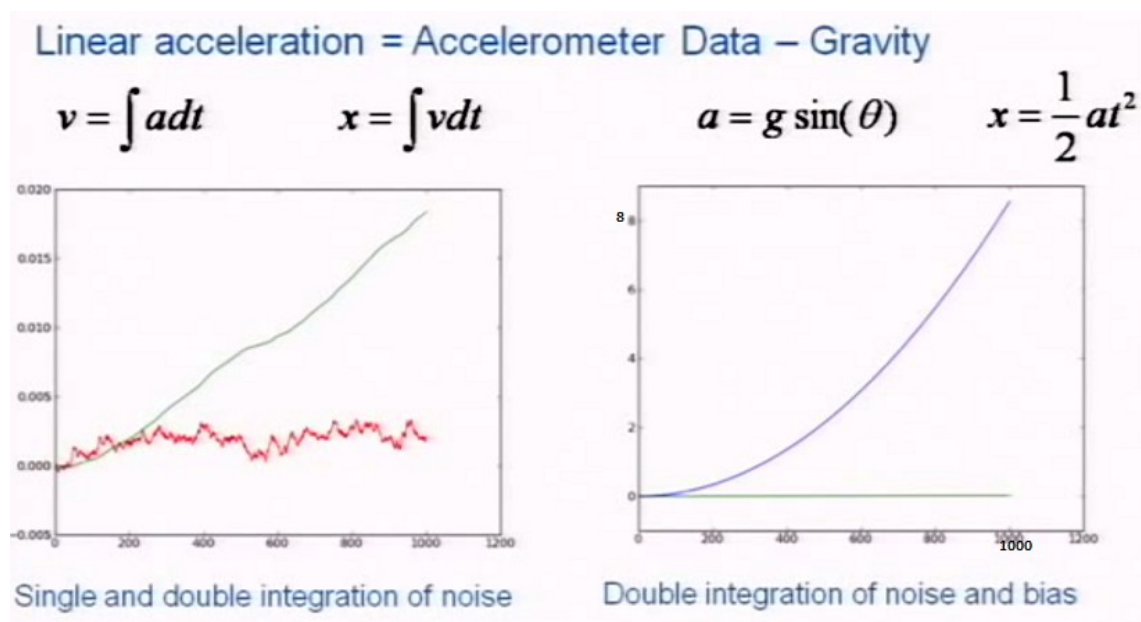


*Ilustrace 6: Princip WPS*

Výhodou WPS je zjištění pozice s přesností pár desítek metrů bez využití GPS. WPS navíc může dosahovat lepší přesnosti než GPS uvnitř budov či v městské oblasti pokryté výškovými budovami, kde se GPS potýká s problémy technologického rázu. Nevýhodou WPS je její závislost na dostupnosti Wifi sítí.

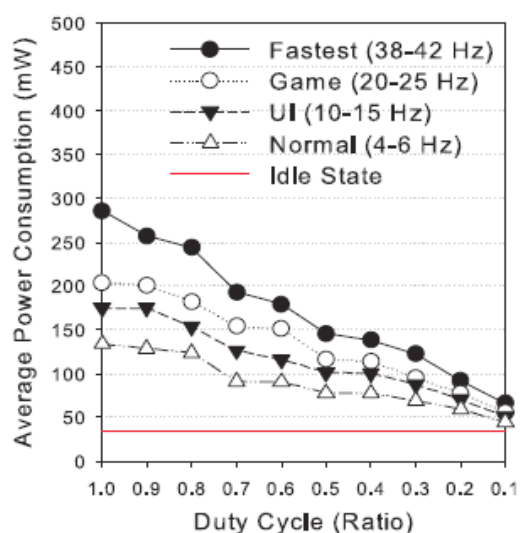
## **2.3 Použití senzorů**

Pro účely práce jsou zajímavé senzory akcelerometr a magnetometr. Když se zkombinují data, získaná z obou senzorů lze určit jakým směrem se zařízení pohybuje. Akcelerometr udává hodnoty zrychlení v metrech za sekundu, a to včetně gravitačního zrychlení. Pokud tedy zařízení leží nehybně na stole, senzor ukazuje hodnotu zrychlení ve směru vertikální osy  $9.81 \text{ m/s}^2$ . Proto pro měření lineárního zrychlení je nutné odfiltrovat přínos gravitační síly. Samotné senzory nemají vysokou spotřebu energie, ale pro jejich používání je nutné, aby procesor neustále pracoval, což způsobí nárůst spotřeby. Tento problém by mohl být vyřešen, pokud by výrobci začali vyrábět navíc procesor určený pouze k obsluze senzorů, pak by nemusel být hlavní výkonný procesor tímto zatěžován. Pro zjištění pozice lze použít dvojitou integraci lineárního zrychlení přes čas  $\iint a dt$ , ale míra chyby je vysoká a roste s časem kvadraticky, nehledě na to, že výpočet z chybně určené pozice chybu dále zvětšuje. Dalším faktorem způsobujícím chybu je kvalita gyroskopu. Například při špatně určeném naklonění nepohyblivého se zařízení o pouhý jeden stupeň vznikne odchylka v určení pozice osm metrů, jak ukazuje ilustrace 7.



Ilustrace 7: Chyba akcelerometru [6]

Ilustrace 7 ukazuje nepřesnost v určení pozice při použití akcelerometru. Horizontální osa udává počet milisekund a vertikální osa nepřesnost v metrech. Na levém grafu je chyba při klidném zařízení v rámci desítek centimetrů. Červená křivka značí rychlost, modrá pozici. Graf vpravo ukazuje určení pozice při chybně určené orientaci o jeden stupeň. Ilustrace také obsahuje vzorce používané pro určení pozice. [6]



Ilustrace 8: Spotřeba akcelerometru [4]

Na ilustraci 8 je graf, který zobrazuje průměrné spotřeby energie při různých nastaveních frekvence a poměru využití senzoru. Na horizontální ose jsou hodnoty poměru využití a na vertikální je průměrná spotřeba energie.



### 3 Použité nástroje a zařízení

Pro vývoj byl použit mobilní telefon HTC One SV, který obsahuje dvoujádrový procesor s frekvencí 1.2 GHz a 1GB paměti RAM. Telefon podporuje GPS a obsahuje řadu senzorů jako akcelerometr, gyroskop, kompas a další. Kapacita baterie je 1800 mAh. Telefon je vybaven systémem Android ve verzi 4.2.2. Jako vývojové prostředí bylo použito Android Studio, které je postaveno na základu prostředí IntelliJ IDEA a poskytuje integrované vývojářské nástroje a možnost ladění. Projekty v prostředí Android Studia jsou překládány pomocí automatického nástroje s názvem Gradle. Pro verzování projektu byl použit software Git, konkrétně služba Github. Pro vývoj webové služby byl použit framework Grails ve verzi 2.3.5 a vývojové prostředí IntelliJ IDEA.



*Ilustrace 9: HTC One SV*

#### 3.1 Grails

Grails [11] je open source framework pro webové aplikace využívající jazyk Groovy. Byl vytvořen za účelem vysoké produktivity s myšlenkou „Convention over configuration“, také známé jako „Coding by convention“, která se snaží snížit počet rozhodnutí, která musí vývojář dělat, a tím získat na jednoduchosti bez ztráty flexibility. Vývojář musí specifikovat pouze to, čím se liší oproti konvencím. Jazyk Groovy je odvozený od jazyka Java s tím, že se snaží být podobný jazyku Ruby. Překladač jej ale překládá do mezikódu Javy, takže výsledná aplikace pak běží pod interpretem Java Virtual Machine.

Grails vychází z frameworku Ruby on Rails a původně nesl název Groovy on Rails, než byl na žádost tvůrce Ruby on Rails přejmenován. Grails měl za cíl poskytnout webový framework pro platformu Java, znovu použít stávající Java technologie Spring a Hibernate pod jedním rozhraním, použít šablon pomocí prezentačního jazyka Groovy Server Pages (GSP), který umožňuje vytvoření dynamického obsahu, dále poskytnout nastavitelnou a rozšiřitelnou podporu Ajaxu a vývojářský mód, zahrnující webový server a automatické načítání zdrojů.

Grails se liší v základu ve třech vlastnostech oproti tradičním javovským webovým frameworkům. Jmenovitě to je žádná konfigurace v XML, vývojové prostředí s webovým serverem připraveným ihned k použití a funkcionality skrze mixiny. Mixin je metoda přidávaná k třídě dynamicky, která umožňuje použití různých operací bez nutnosti implementování rozhraní. Například v Grails mají doménové třídy automaticky přiděleny perzistentní operace jako save, delete nebo find. Vytváření webových aplikací v Javě tradičně zahrnuje konfiguraci prostředí a frameworků, která se často nachází v externím XML souboru. V Grails se používají pravidla či konvence, podle kterých se prochází kód v Grails aplikacích. Například pokud třída nese název končící Controller, je považována za webový kontroler.

```
class Location {
    String latitude
    String longitude
    String provider
    Date time
    String accuracy
    SecUser secUser

    static constraints = {
        longitude blank: false, nullable: false, maxSize: 20
        latitude blank: false, nullable: false, maxSize: 20
        time nullable: false
        provider nullable: true
        accuracy nullable: true
        secUser nullable: true
    }

    static mapping = {
        sort time: "desc"
    }
}
```

*Ilustrace 10: Ukázka doménové třídy v Grails*

Grails používá architekturu model-view-controller, ve zkratce MVC. Kontroler se stará o chování a logiku stránky a zpravidla se umísťuje v projektu do složky controller. To zaručí, že bude rozpoznán v Grails a bude i automaticky přidán do možných URL cest v aplikaci i včetně jeho metod. Část view se stará o zobrazení dat, ke kterému se používá GSP či JSP (Java Server Pages). Tyto soubory se umísťují do složky views a pokud se dodržují zmiňované konvence, Grails zajistí příslušná napojení na kontroler. Grails také poskytuje řadu značkovacích knihoven, ale je možné vytvářet i vlastní. Značky z Grails se označují předponou „g:“. Pro model se v Grails používají doménové třídy, které se umísťují do adresáře domain a pro uložení do databáze se používá GORM neboli Grails Object Relational Mapping. Doménová třída obsahuje pouze definici proměnných a jejich integritní omezení. Grails přidává identifikační klíč automaticky. Metody pro práci s databází jsou automaticky dostupné. Jedná se o metody save, delete, refresh či ident, která vrací identifikační klíč objektu. Dalšími dostupnými metodami je například metoda pro počet záznamů v databázi, test existence objektu nebo mnoho variací na hledání v databázi. Nastavení databáze lze měnit v souboru DataSource.

Dalšími konfiguračními soubory jsou BuildConfig, UrlMappings a Bootstrap. V BuildConfig se nastavují pluginy a závislosti, v UrlMappings lze určit, co se má stát při vstupu prohlížeče na určitou adresu a v Bootstrap lze definovat stav, se kterým se má aplikace spouštět. Například lze předem vytvořit uživatele, aby pak byl aplikaci ihned dostupný.

Pro správu rolí je možné v Grails použít plugin Spring Security [13]. Ten po instalaci přidá do projektu doménové třídy SecUser, SecRole a spojovou třídu SecUserSecRole. Díky němu lze nastavit restriktce na oblasti aplikace v závislosti na roli nebo lze dynamicky zobrazovat obsah dle role.

## **4 Cíle práce a návrh řešení**

### **4.1 Cíle práce**

Cílem práce je vytvořit takzvaný trackovací systém pro platformu Android, který by měl jednak shromažďovat naměřená data od uživatelů na jednom místě a zároveň se zabývat vlastním měřením přímo u uživatelů.

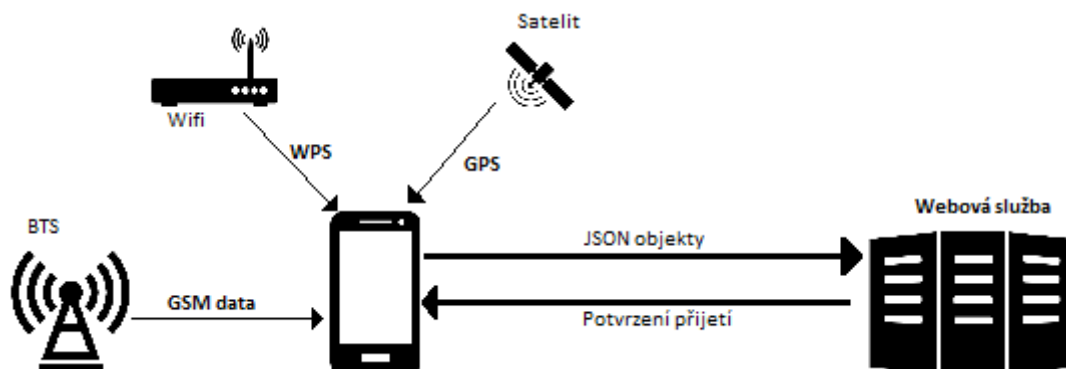
Je proto potřeba navrhnout a vytvořit způsob, jakým má fungovat určování polohy. To znamená navrhnout jak často zjišťovat aktuální polohu, zvážit použití i jiných metod zjištění polohy než GPS a určit jaké polohy jsou vhodné a přínosné k archivaci. Mobilní aplikace by měla mít možnost nastavení různých úrovní přesnosti měření a s tím spojenou spotřebu energie. Dalším požadavkem na mobilní aplikaci je, aby byla schopna běžet neustále i na pozadí bez uživatelského rozhraní. Webová aplikace by měla data od klientů ukládat a dále zobrazovat. Jednotliví uživatelé by měli být nějakým způsobem rozlišení, mít různá práva a mít dostupné pouze svoje záznamy s tím, že by existoval jakýsi účet správce, který má dostupné všechny funkce a záznamy. Dále je potřeba definovat a vytvořit způsob komunikace mezi oběma aplikacemi. Na straně mobilní aplikace je to odesílání záznamů a reakce na neúspěšné odeslání a na straně serveru se jedná o příjem záznamů a jejich následné uložení.

### **4.2 Návrh systému**

Systém bude rozdělen na část webovou a mobilní. Webová část bude vytvořena v jazyce Groovy a bude využívat frameworku Grails. Dále zde bude implementován systém rolí pomocí pluginu Spring Security, jednotlivé záznamy lokací budou zobrazeny v seznamu a budou dostupné pouze určeným uživatelům. Aplikace bude přijímat nové záznamy ve formátu objektu JSON a ukládat je do databáze. Komunikace bude probíhat po protokolu HTTP, a to konkrétně prostřednictvím jeho metody POST.

Aby záznam mohl probíhat neustále, bude mobilní aplikace implementovat službu, která se bude starat o hledání pozice i o komunikaci se serverem. Pro prezentaci informací bude použito jednoduché uživatelské rozhraní, kde budou dostupné informace o stavu zpracovávání záznamů. Aplikaci by nemělo být možné provozovat anonymně, proto bude vyžadováno přihlášení uživatele hned při startu. Přihlášený uživatel bude zapamatován a bude jej možné kdykoli měnit. V aplikaci bude možné nastavovat

využívání služby GPS, schopnost měnit stav Wifi připojení, interval mezi pokusy o zjištění polohy a požadovanou vzdálenost od poslední uložené polohy. Pro hledání polohy bude aplikace využívat všechny dostupné zdroje. Je to GPS, Wifi, GSM síť a pasivně získávat data od ostatních aplikací.



*Ilustrace 11: Návrh systému*

Při maximálním nastavení aplikace bude služba GPS využívána stejně často jako ostatní. Povolení využití GPS služby nelze na platformě Android měnit programově kvůli bezpečnosti, a tak je nutné, aby jej uživatel nastavil ručně. Běžně tato aplikace řeší odkazem přímo do nastavení systému. Dále je také doporučeno, aby měl uživatel zapnuté Wifi připojení. Pokud tomu tak nebude a aplikace bude mít povolenou schopnost měnit stav Wifi připojení, bude aplikace v určitém intervalu povolovat připojení na krátkou dobu za účelem pokusu zjištění polohy pomocí zmiňovaného WPS. I při nízkém nastavení použití GPS bude tato služba přesto používána a to tak, že se bude v určitém větším intervalu spouštět na krátkou dobu, aby se pokusila zjistit polohu. Toto je navrženo za účelem získání přesnější lokace při ucházející spotřebě v místech, kde nejsou dostupné Wifi sítě a záznamy by byly jinak získávány pouze ze sítě GSM. Aby systém neukládal veškeré lokace, bude aplikace implementovat algoritmus, který bude rozhodovat o prospěšnosti nové lokace. Například pokud nová lokace je stále v okruhu nepřesnosti předešlé, není potřeba ji znovu ukládat.

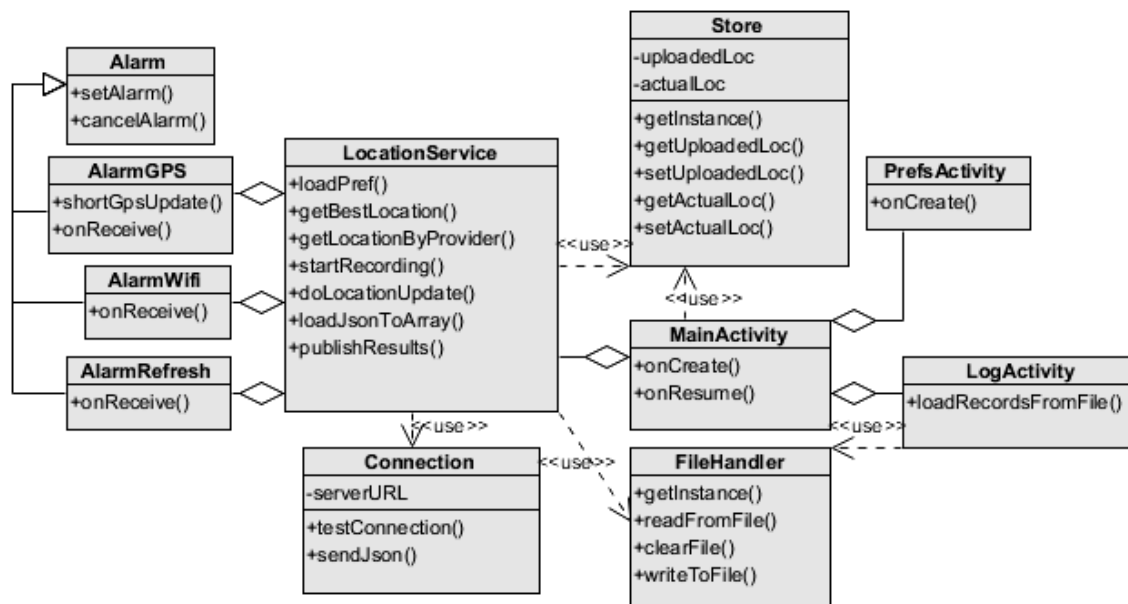
Aplikace bude schválené lokace odesílat na server ve formátu objektu JSON. Pokud server nebude dostupný, budou se data ukládat do souboru, kde vyčkají do dostupnosti serveru. Pokud odeslání proběhne úspěšně bude obsah skladovacího souboru vymazán.

### 4.2.1 Použití akcelerometru

Použití akcelerometru s gyroskopem a kompasem nebude v aplikaci použito z následujících důvodů. Nejprve toto použití vnáší do měření chybu, která by sice mohla být zpočátku zanedbatelná v porovnání s nepřesností polohy z GSM sítě, ale výrazně roste s časem a rychlostí pohybu. Tato chyba vzniká při výpočtu dvojité integrace lineární akcelerace, jak bylo zmíněno v kapitole 2.3. Systémy, které tuto metodu používají, většinou chybu korigují pomocí synchronizace se službou GPS v určitých intervalech. Zjišťování polohy bez použití této metody je možné za předpokladu, že dopředu víme, že bude aplikace používána pouze při stanoveném stavu zařízení. Například zařízení bude umístěno ve stejné poloze v autě či bude mít uživatel připnutý senzor na noze a aplikace bude počítat počet vykonaných kroků. Takto vytvořené aplikace jsou při nedodržení těchto předpokladů nepoužitelné. Hlavním důvodem, který vedl k zavržení této metody bylo také zjištění velké náročnosti používání senzorů na procesor, a tedy spotřebě energie. Aby totiž data z akcelerometru měla nějakou vypovídající hodnotu, musí být čtena velmi často a tedy procesor je stále aktivní a mohou ho použít i jiné aplikace. Při měření po dobu deseti minut a použití nejmenšího počtu vzorků, kde frekvence je 4-6 Hz, je spotřeba větší než zapínání GPS každou minutu po dobu pěti minut, jak je uvedeno v [4]. Dalším důvodem je také to, že u mnoha zařízení výrobci nepodporují získávání dat ze senzorů, když je obrazovka vypnutá. Seznam nepodporovaných zařízení je na [16]. Jelikož při snaze využít tento způsob získávání dat byl kladen velký důraz na snížení spotřeby a pro korekci chyby by bylo stejně nutné využívat GPS, bylo od tohoto důvodu upuštěno a za účelem snížení spotřeby se bude využívat GPS s větším časovým intervalem.

## 5 Mobilní aplikace

### 5.1 Popis struktury aplikace



Ilustrace 12: Class diagram

Aplikace obsahuje tři aktivity a jednu službu, která běží neustále. V hlavní aktivitě se zobrazuje aktuální poloha, poslední poloha určená k archivaci, poskytovatel polohy, přesnost a čas měření. Dále je zde tlačítko, které vytvoří aktivitu logu, ve které se zobrazují dosud nedeslané polohy načtené ze souboru. Poslední aktivita je nastavení aplikace dostupné skrze klasické aplikační menu. Nastavuje se zde aktuální uživatel, využití služby GPS, povolení aplikaci měnit stav Wifi, minimální rozdíl v poloze a časový interval mezi jednotlivými hledáními polohy. Veškeré hledání, manipulace a odesílání polohy se odehrává ve službě *LocationService*. Služba se v systému Android používá pro operace dlouho běžící na pozadí, které neposkytují uživatelské rozhraní a běží dále i za stavu, kdy se uživatel přepne do jiné aplikace. Služby také mají přednost před aplikacemi uloženými v mezipaměti, které čekají na obnovení, a tak je méně pravděpodobné, že budou ukončeny při nedostatku prostředků.



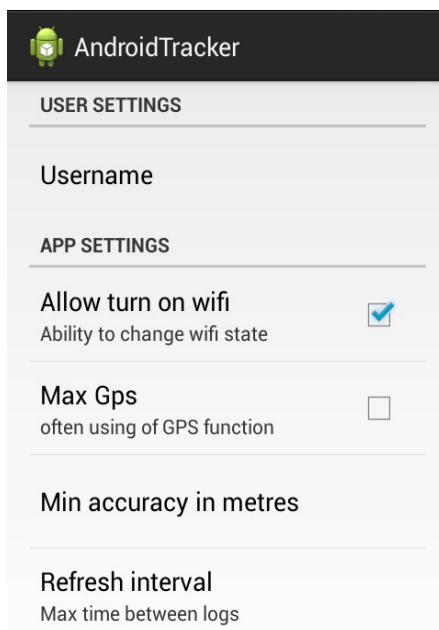
*Ilustrace 14: Ukázka hlavní obrazovky mobilní aplikace*



### 5.1.1 Ukládání uživatelských nastavení

Uživatelská nastavení se ukládají pomocí třídy *SharedPreferences* a jsou uchována i při ukončení aplikace. Pouze odinstalováním aplikace jsou tato nastavení smazána. Jejich definice je obsažena v souboru *preferences.xml* a lze je pro lepší přehlednost seskupovat do různých kategorií. Tato nastavení se editují ve vlastní aktivitě a mají ošetřené vstupy proti nežádoucím parametrům. Nastavení jména uživatele je pro běh aplikace nutné, a tak pokud není zadáno, je při startu aplikace uživatel vyzván k jeho zadání. V případě neúspěšného zadání je aplikace ukončena. Pro editaci nastavení slouží *PrefsActivity*, do níž je vložen *PrefsFragment*, který je podporován od API verze 11, proto jsou minimálním požadavkem na běh aplikace Android ve verzi 3.0. Toto řešení se k použití jeví jako vhodnější a pokud by byl požadavek na funkčnost aplikace nižší verzi, musela by být tato část přepracována.





*Ilustrace 15: Ukázka aktivity nastavení*

## **5.2 Komunikace se serverem**

Veškerá logika komunikace se serverem se nachází ve třídě *Connection* uložená do balíčku *net*. Třída *Connection* obsahuje dvě metody. Jednu pro test spojení se serverem a druhou na odesílání záznamů jako JSON pole. Test spojení se serverem probíhá pokusem vytvoření spojení a pokud pokus selže, tak není server dostupný. Vše probíhá ve vedlejším vlákne, aby nebyla blokována funkce vlákna, ze kterého byla metoda zavolána. Odesílání záznamů využívá třídu *HttpPost* z balíčku *org.apache.http*. Dále podle úspěchu přenosu na server metoda vrací boolean hodnotu, aby mohla aplikace zareagovat v případě neúspěchu.

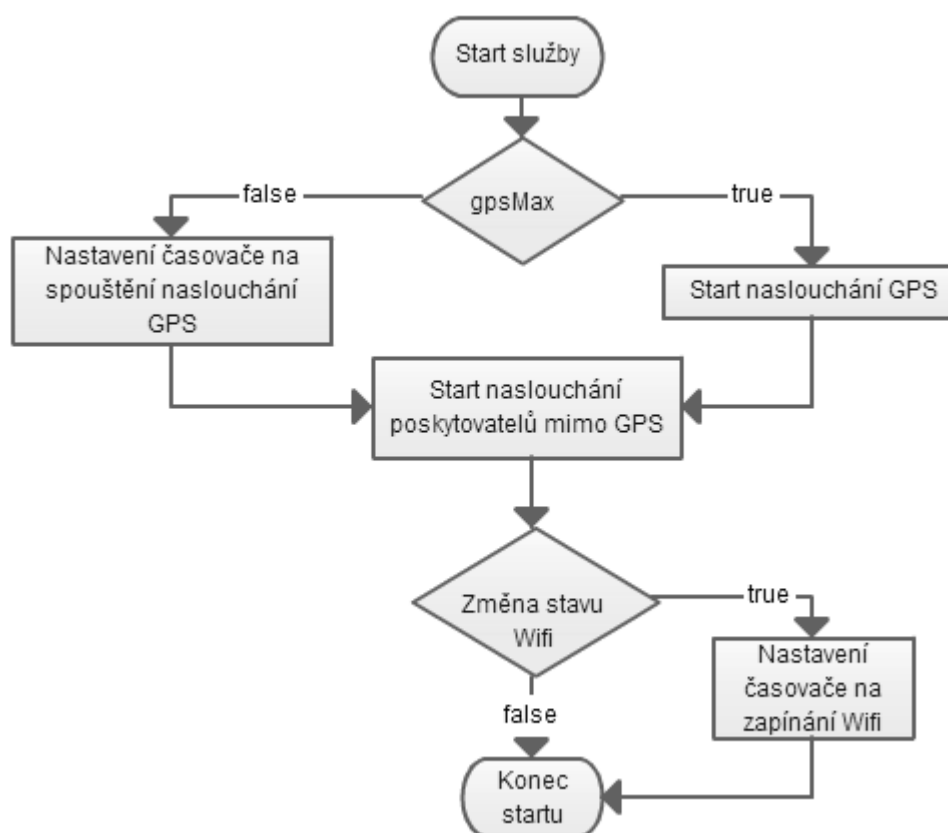
## **5.3 Ukládání dat**

Pokud není server dostupný, záznamy se ukládají do souboru, zde pojmenovaném jako „log.txt“. Pro práci se soubory byla vytvořena třída *FileHandler*, která byla umístěna do balíčku *io*. Třída *FileHandler* používá návrhový vzor Singleton, existovat může tedy pouze jedna instance dané třídy a pro vytváření objektu se využívá statická metoda *getInstance()*. Jsou v ní implementovány metody pro čtení, zápis a smazání obsahu souboru. Konkrétně metoda pro čtení vrací celý obsah souboru jako jeden objekt typu *String* a metoda pro zápis přispisuje obsah parametru na konec souboru.

Třída Store je umístěna v balíčku model, používá také návrhový vzor Singleton a slouží k ukládání dat. Jelikož i při pouhé změně orientace obrazovky ztratí aplikace dynamicky nastavené proměnné v hlavní aktivitě, načítají se z této třídy.

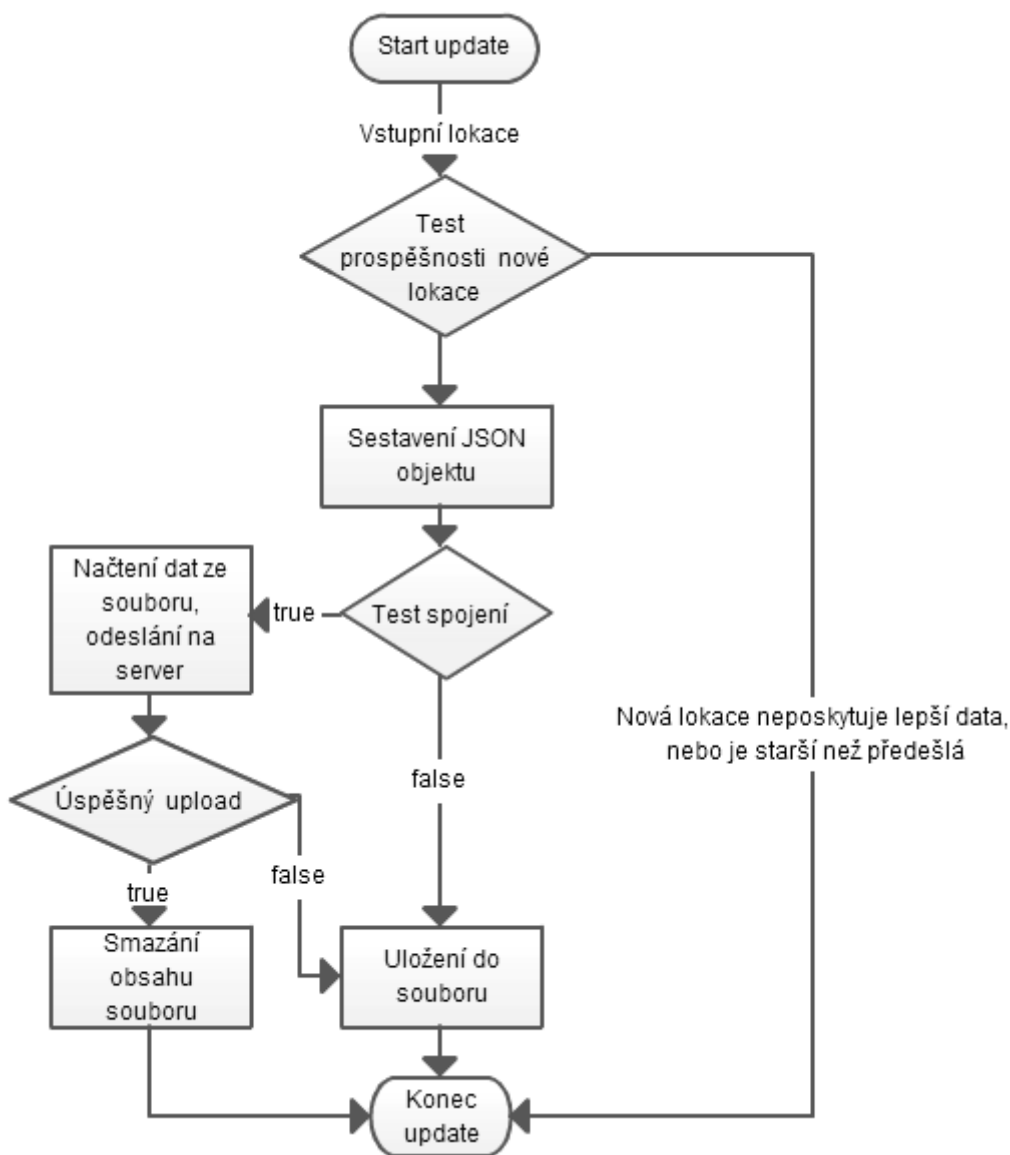
## 5.4 Algoritmus hledání polohy

Algoritmus hledání polohy vychází z ukázky kódu uvedené v příspěvku [5] a má různé podoby v závislosti na uživatelském nastavení. Hlavní faktory, ovlivňující průběh hledání, jsou schopnost aplikace měnit stav Wifi připojení a intenzita využití GPS. Hodnoty těchto nastavení ovlivňují průběh startu služby, kde se spouští časovače, jak je lépe popsáno dále v textu. Při povolení změny stavu Wifi připojení, se spustí časovač, který v určitém intervalu, zde nastaveném na tři minuty, povolí Wifi připojení na patnáct sekund, pokud již nebylo aktivní. Pokud by došlo k ukončení služby v intervalu zapnutí Wifi připojení aplikací, je zde nastaveno, aby bylo opět ukončeno. V rámci tohoto patnáctisekundového okna se nejprve otestuje, zda nějaké záznamy čekají k odeslání, a teprve pak z hlediska úspory se testuje dostupnost serveru k jejich případnému odeslání.



Ilustrace 16: Průběh startu služby

Obecně jsou vyžadovány změny polohy od všech poskytovatelů v nastaveném intervalu a vzdálenosti. U poskytovatele GPS záleží na nastavení. Při obdržení nově získané polohy je poloha odeslána jako parametr do metody *doLocationUpdate*, kde se testuje užitečnost příchozí lokace. Nová lokace se porovnává s poslední, která prošla tímto testem. Nejprve se porovnává, zda vzdálenost obou lokací je větší než nastavená minimální. Poté se testuje za předpokladu zhoršené přesnosti, zda-li není minulé lokace stále v dosahu nové přesnosti a nakonec se porovná, jestli je záznam novější než předešlý. Pokud nějaká podmínka není splněna, je lokace zahozena.



*Ilustrace 17: Průběh testu nové lokace*

Na prvních prototypech se ukázalo, že v rámci této metody dochází k redundanci ukládaných záznamů v důsledku několikanásobného volání ve stejný moment. Toto prakticky nelze ovlivnit, jelikož je tato metoda volána jak z časovače, tak z listenerů různých poskytovatelů lokace. Byla tedy nastavena synchronizace na tu část kódu, kde se porovnává nová lokace s minulou uloženou až do části ukládající novou jako minulou. Díky synchronizaci se tato část kódu po vstupu uzamkne a odemkne se pro další vykonávání až při odchodu programu z kritické sekce.

Výraznější změnou je nastavení využití GPS. Pokud je zaškrtnuto maximální nastavení, dotazuje se aplikace na změny podle nastaveného časového intervalu stejně jako u ostatních poskytovatelů. Při nízkém využití se při startu služby spustí časovač, který se jednou za deset minut snaží v rámci třiceti sekund zjistit aktuální polohu. Konkrétně se žádá poskytovatel GPS o změny polohy v intervalu pěti sekund. Dále také nejsou vyžadovány změny polohy podle nastaveného intervalu jako u ostatních poskytovatelů.

## **5.5 Alarmy**

Většina časovačů byla nakonec implementována pomocí alarmů. Týkalo se to především těch, které byly určeny k dlouhému běhu. Při testování prototypů se ukázalo, že systém Android přechází po nějaké době do režimu spánku a tím se přeruší vykonávání programu i ve formě služby. Tomuto lze předejít vytvořením WakeLocku, který zamezí přechodu do tohoto režimu do té doby, než bude zase uvolněn. Existuje několik typů, liší se v tom, co vše probouzí. Zde byl použit částečný "partial", který probouzí pouze procesor. Díky použití WakeLocku přestalo docházet k pozastavení služby, ale zároveň udržování procesoru v neustálém chodu, způsobilo nárůst spotřeby energie. Bylo proto nutné použití alarmů, které jsou registrovány i při režimu spánku. Pro vytvoření alarmu je nutné dědit od třídy *BroadcastReceiver* a dodefinovat metodu *onReceive()*, která je zavolána po uplynutí stanoveného intervalu a určuje co se má dále vykonat. V této metodě je potřeba vzbudit procesor pomocí WakeLocku, provést požadovanou funkcionalitu ve službě a nakonec uvolnit WakeLock, aby zařízení mohlo přejít zpět do režimu spánku.

Vytvořené alarmy v aplikaci nesou název třídy *AlarmWifi*, *AlarmGPS* a *AlarmRefresh*. *AlarmWifi* se spouští při nastavení možnosti měnit stav Wifi, *AlarmGPS* se používá při nízkém nastavení použití GPS a *AlarmRefresh* slouží k periodickému pokusu o nalezení pozice a jejímu příslušnému odeslání. Při registraci více alarmů je potřeba u každého uvést jiné číslo v proměnné *requestCode*, jinak by se navzájem přepisovaly. Alarmy se registrují pomocí třídy *AlarmManager* předáním instance třídy *PendingIntent* a časového intervalu.

Další změnou prošla registrace naslouchání poskytovatelů polohy podle [3]. Původně používaný *LocationListener* je spíše vhodný pro aplikace v popředí a pro aplikace na pozadí je doporučeno použít třídu *PendingIntent*. I zde je nutné pro každý zvlášť nastavit jiný *requestCode*. Díky těmto změnám došlo k získání chtěné funkcionality při zachování spotřeby.

## 6 Webová služba

Druhou částí práce bylo navržení a vytvoření webové služby, která by shromažďovala data naměřená mobilní aplikací a následně zprostředkovala uživateli jejich zobrazení. Bylo tedy nutné vybrat vhodnou platformu pro její vývoj, určit jakým způsobem budou mezi sebou jednotlivé aplikace komunikovat, jaká a v jakém formátu se budou uživateli zobrazovat data a zvolit role přístupu k systému. Služba je dostupná na adrese <http://grailsapp-lunak.rhcloud.com/>

### 6.1 Platforma

Platformou je myšlena technologie, kterou musí podporovat server k běhu aplikace a ukládání dat. Často používaný formát se skládá z aplikace v PHP, ASP či Javě a SQL databáze. Jelikož požadavky platformy na aplikaci nebyly striktně dané a sama aplikace je poměrně jednoduchá, mohl být vybrán zatím nepříliš rozšířený model použití frameworku Grails a jazyka Groovy. Framework Grails umožnil rychlý vývoj, díky implementované správě rolí, generování HTML šablon se styly či automatické práci s databází. Jako databáze byla použita H2, která se ve frameworku Grails základně používá a nebyl žádný důvod přecházet na jiné řešení.

Výběru hostingu měl dvě hlavní kritéria. Jednak aby podporoval framework Grails a dále aby jeho používání nebylo zpoplatněno. Z rešerše vyšel nejlépe hosting Openshift [12], který podporuje navíc i nástroj Jenkins sloužící pro usnadnění nasazování aplikací na server. Openshift poskytuje firma RedHat, volně dostupné jsou tři virtuální stroje, přičemž je možné kdykoli přejít na zpoplatněnou službu a využít jejich cloudu. Na serveru je instalován Apache Tomcat.

### 6.2 Komunikace

Komunikace probíhá na úrovni protokolu HTTP a konkrétně používá objekty typu JSON, které odesílá skrze metodu POST mobilní aplikace. Webová aplikace následně pošle pouze potvrzení o jeho přijetí. Mobilní aplikace ještě využívá klasickou metodu GET pro test dostupnosti serveru. Mimo jiné by zde mohla být i metoda pro ověření uživatele mobilní aplikace, ale ta v současné verzi ještě nebyla implementována.

### 6.2.1 Formát objektu JSON

Každý JSON se posílá jako pole objektů i v případě, že se jedná pouze o jeden objekt. Toto řešení je univerzálnější a umožňuje jednotnou implementaci na straně serveru. Server tedy naslouchá a obsluhuje požadavky, které přijdou na adresu „název serveru“/addlocation. Toho je ve frameworku Grails dosaženo přidáním cesty /addlocation do UrlMappings a nastavením kontroleru a metody, která ji má obsluhovat. V této metodě se nachází již konkrétní implementace, která příchozí pole zpracuje a jednotlivé záznamy, pokud splňují integritní omezení, vloží do databáze.

---

```
[{"time":1398499851000,"longitude":15.076975,"provider":"gps","username":"radek","latitude":50.754544,"accuracy":49}, {"time":1398499932000,"longitude":15.076978,"provider":"gps","username":"radek","latitude":50.75458,"accuracy":49}]
```

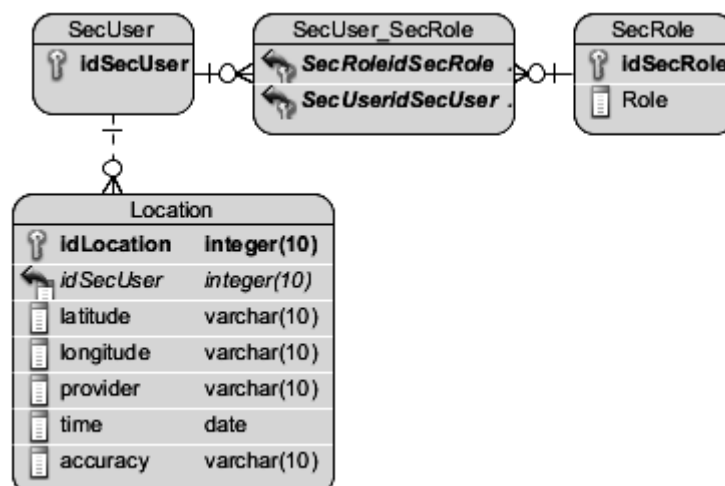
---

*Ilustrace 18: Ukázka objektu JSON*

Z obrázku výše je patrné, že JSON objekt obsahuje parametry jako Uživatelské jméno, zeměpisnou šířku a délku, čas měření, přesnost a poskytovatele polohy. JSON objekt funguje na principu asociativního pole a nezáleží v něm na pořadí dat, proto se může stát, že bude sestaven různými způsoby.

### 6.3 Databáze

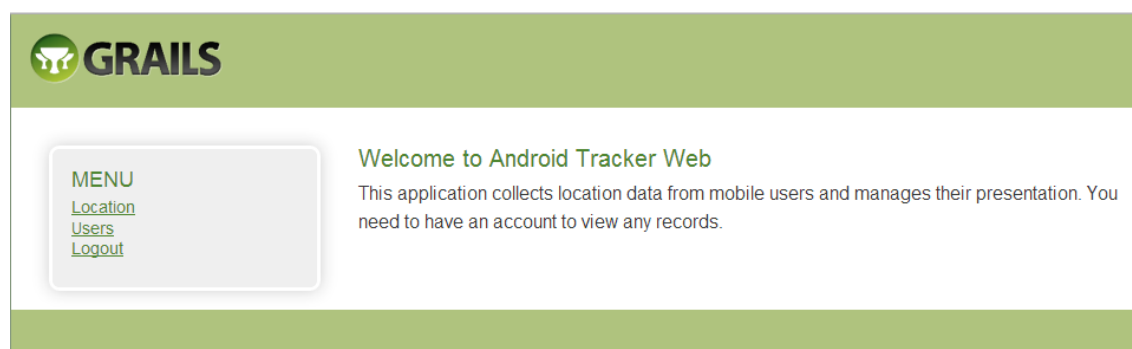
Databázi pohání engine H2 napsaný v Javě. V Grails se tabulky nevytvářejí přímo, nýbrž pomocí doménových objektů. Ty pouze definují, jaké mají obsahovat proměnné a jejich integritní omezení. Tabulky a jejich propojení se vytvoří automaticky na základě těchto doménových objektů. Služba obsahuje čtyři doménové objekty, přičemž tři jsou vygenerované pluginem Spring Security pro práci s rolemi. Zbýlý objekt s názvem Location slouží k ukládání příchozích záznamů. Obsahuje stejné proměnné jako JSON objekt a pro uložení do databáze stačí na instanci tohoto objektu zavolat metodu *save()*. Na ilustraci 19 je ukázka, jak pravděpodobně vypadají vygenerované tabulky z doménových objektů.



Ilustrace 19: ERD

## 6.4 Uživatelské prostředí

Celý tento web využívá styly z Grails, a tak bylo ponecháno i logo s odkazem na tuto technologii. Po zadání příslušné webové adresy se zobrazí úvodní strana aplikace. V levé části se nachází menu aplikace. Obsahuje položky Location records, Login či Logout a pokud uživatel má roli administrátora, tak i položku Users.



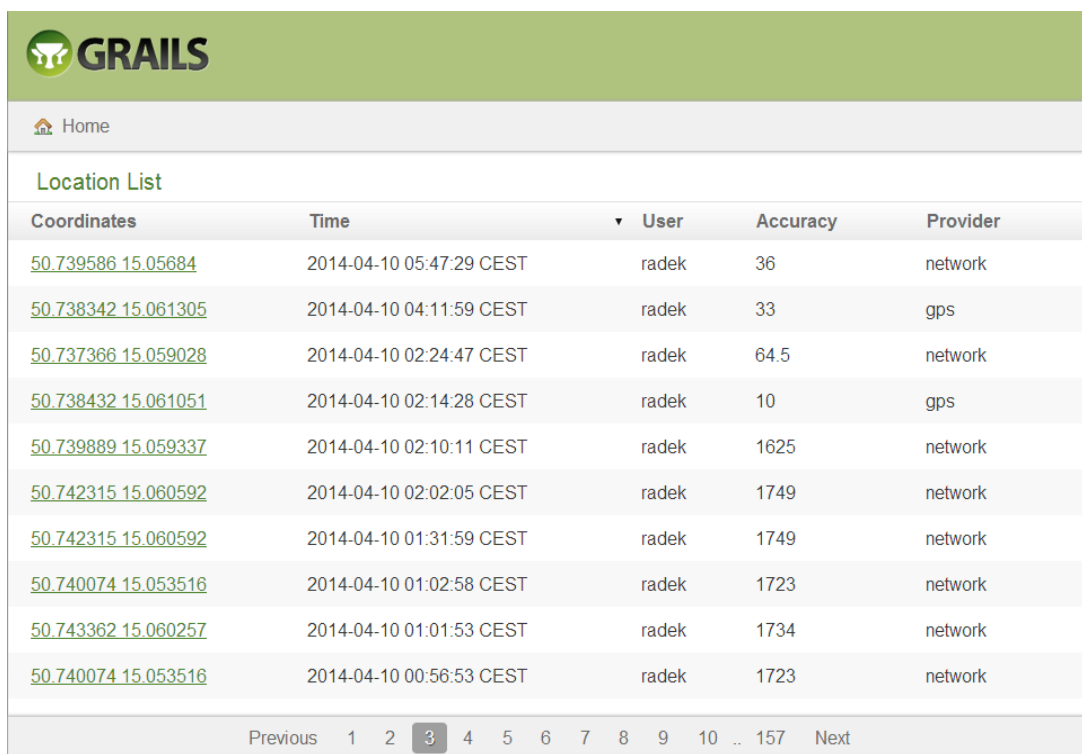
Ilustrace 20: Úvodní strana webové služby

V současné podobě systém obsahuje pouze dvě role, a to administrátora a uživatele. V Grails se o systém rolí stará plugin Spring.Security, a tak je i zde použit. Použití rolí umožní rozlišení typů přístupů k aplikaci a nastavení restriktce na oblasti aplikace. Například bez přihlášení je dostupná pouze úvodní strana a pokud uživatel zadá jinou stránku, ať odkazem či přímo v adresním řádku, je přesměrován na login dialog.



Naměřená data se zobrazují pod položkou menu Location records, kde každý uživatel vidí pouze svoje záznamy. Výjimkou je administrátor, který má dostupné veškeré záznamy všech uživatelů. O toto se stará služba LocationService, která předá stránce pouze data podle přihlášeného uživatele. Jednotlivý záznam obsahuje souřadnice, které jsou zároveň nastaveny jako odkaz na Google mapy, kde je dobře patrné o jaké místo se jedná. Dále záznam obsahuje čas měření, který má v nynější verzi nastaven formát středoevropského času. Hostingový server se totiž pravděpodobně nacházel v Americe a systém pak čas zobrazoval podle lokálního nastavení, což pro zdejší využití nebylo zcela vhodné. Dalšími parametry záznamu jsou jméno uživatele, který jej naměřil, přesnost a z jakého zdroje byla data získána. Kromě souřadnic lze záznamy libovolně řadit, přičemž v základu jsou seřazeny sestupně dle času. Pro přehlednost se záznamy stránkují.

Další položkou menu je Users, která umožňuje manipulaci s uživateli. Tato položka je dostupná pro uživatele s rolí administrátora a je zde možné přidávat nové uživatele, odstranit staré či jen prohlížet aktuální.

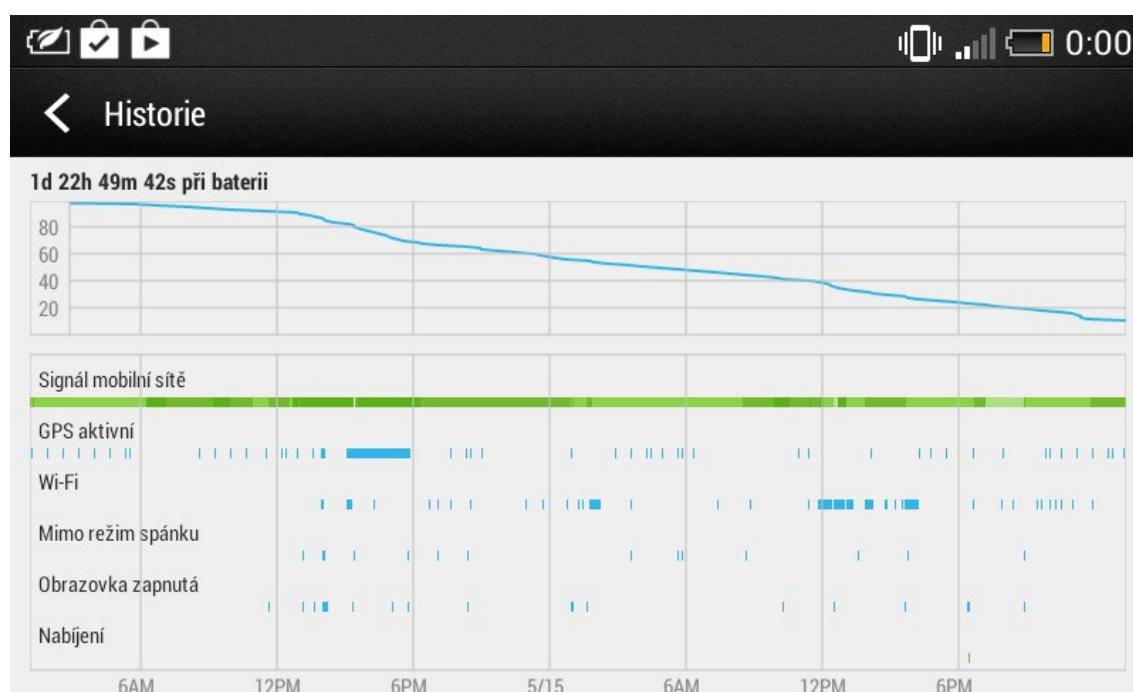


Coordinates	Time	User	Accuracy	Provider
<a href="#">50.739586 15.05684</a>	2014-04-10 05:47:29 CEST	radek	36	network
<a href="#">50.738342 15.061305</a>	2014-04-10 04:11:59 CEST	radek	33	gps
<a href="#">50.737366 15.059028</a>	2014-04-10 02:24:47 CEST	radek	64.5	network
<a href="#">50.738432 15.061051</a>	2014-04-10 02:14:28 CEST	radek	10	gps
<a href="#">50.739889 15.059337</a>	2014-04-10 02:10:11 CEST	radek	1625	network
<a href="#">50.742315 15.060592</a>	2014-04-10 02:02:05 CEST	radek	1749	network
<a href="#">50.742315 15.060592</a>	2014-04-10 01:31:59 CEST	radek	1749	network
<a href="#">50.740074 15.053516</a>	2014-04-10 01:02:58 CEST	radek	1723	network
<a href="#">50.743362 15.060257</a>	2014-04-10 01:01:53 CEST	radek	1734	network
<a href="#">50.740074 15.053516</a>	2014-04-10 00:56:53 CEST	radek	1723	network

*Ilustrace 21: Ukázka seznamu lokací*

## 7 Srovnání různých nastavení aplikace

Již několikrát bylo zmíněno, že mobilní aplikace podporuje různá nastavení a tato část práce obsahuje jejich shrnutí a srovnání. Jelikož aplikace umožňuje zadávání libovolných hodnot pro interval mezi měřeními a pro požadovanou vzdálenost od poslední lokace, neexistuje konečný počet možností nastavení. Lze je tedy rozdělit podle největších odlišností, a to podle intenzity využití služby GPS. První nastavení využívá GPS při každém hledání polohy podle nastaveného intervalu a druhé používá GPS po dobu třiceti sekund každých deset minut. Analýza měření spotřeby probíhala pomocí nástroje Historie obsaženého přímo v operačním systému Android v sekci Napájení.

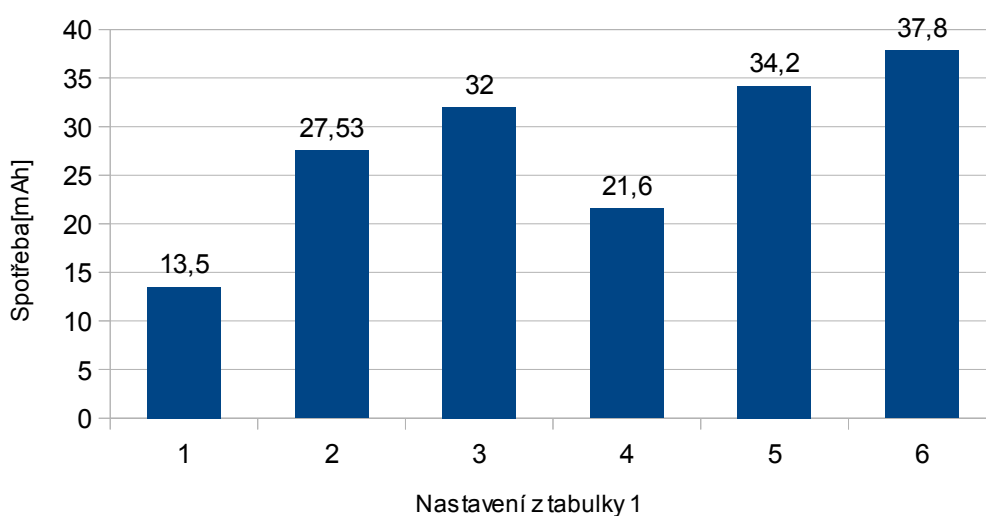


*Ilustrace 22: Historie spotřeby při různých nastaveních*

Na obrázku výše je vidět křivka znázorňující pokles úrovně nabití baterie v průběhu necelých dvou dnů. V této době byla snaha používat telefon co nejméně, aby nebylo výrazně ovlivněno měření. Po celou dobu měření byl nastaven interval mezi hledáním polohy na jednu minutu a minimální vzdálenost od poslední pozice na sto metrů. Během měření bylo nastavení několikrát změněno, což lze vidět i na využití služeb. Záznamy jednotlivých měření jednotlivých nastavení jsou uvedeny dále v tabulce.

Tabulka 1: Měření spotřeby dle nastavení

Nastavení	Doba měření / Spotřeba	Spotřeba [mAh]
1 Nízké GPS bez Wifi	12h / 9%	13,5
2 Nízké GPS se změnou Wifi	8,5h / 13%	27,53
3 Nízké GPS, zapnutá Wifi	4,5h / 8%	32
4 Plně GPS bez Wifi	10h / 12%	21,6
5 Plně GPS se změnou Wifi	10h / 19%	34,2
6 Plně GPS, zapnutá Wifi	10h / 21%	37,8



Ilustrace 23: Graf porovnání spotřeby různých nastavení

Z výsledků měření vyplývá, že aplikace s minimálním nastavením při spotřebě 13,5 mAh, by na testovaném telefonu vydržela běžet kolem 133 hodin. Při maximálním testovaném nastavení lze odvodit stejným způsobem výdrž na necelé dva dny. Nutné je podotknout, že čísla jsou odhadnuta jako maximální výdrž na základě použitého telefonu s předpokladem nevyužívání ostatních aplikací. Ve výsledcích měření mělo nastavení zapínající Wifi horší spotřebu než nastavení používající plně GPS bez Wifi. Tento výsledek je pravděpodobně způsoben komunikací aplikace s webovou službou, jelikož druhé nastavení ukládalo při měření data pouze do souboru. Další možností je, že při povoleném připojení Wifi jej mohou využít i jiné aplikace. Ve prospěch nastavení s Wifi připojením je stav, kdy je uživatel zvyklý jej mít často zapnuté a výdrž je pak ovlivněna méně.

Pokud uživatel zadá menší interval mezi hledáními či menší vzdálenost mezi ukládanými pozicemi se samozřejmě výdrž sníží. Další případ snížení výdrže by mohl nastat, pokud by zařízení měnilo neustále svojí polohu a ta by byla neustále odesílána na server. Takováto častá komunikace se serverem má také vliv na spotřebu. Dále se spotřeba odvíjí i od ostatních spuštěných aplikací, které by normálně byly neaktivní. Díky funkci aplikace, která probouzí procesoru v určených intervalech na nějakou dobu. V této době mohou aplikace vykonat například naplánované funkce, které neprobouzí procesor a čekají na jeho zaktivnění.

Maximální nastavení využití GPS může být používáno více způsoby. Hlavní roli u něho hraje zvolený interval mezi jednotlivými hledáními polohy, který je základně nastaven na jednu minutu. Pokud chceme získat přesnější data, je logické ho nastavit na nižší hodnotu a musíme počítat s nárůstem spotřeby. Druhou možností je tento interval zvýšit, a tak snížit spotřebu energie. Toto řešení je vhodné do oblastí, kde nejsou dostupné žádné Wifi připojení. Druhé nastavení, které se spoléhá na tato data a využívá GPS jen málo, by poskytovalo v takovýchto oblastech horší výsledky.

## 8 Závěr

Na základě získaných znalostí byl vytvořen systém, který dokáže sledovat pohyb více uživatelů najednou. Webová služba podporuje systém rolí a podle něj zobrazuje naměřená data, to znamená, že každý má dostupné pouze svoje záznamy. Dále obsahuje roli správce, který má možnost spravovat uživatele a může zobrazovat veškeré záznamy. Webová služba je vždy připravena zpracovávat požadavky ve formátu objektu JSON od mobilní aplikace. Naměřená data jsou ukládána v databázi.

Mobilní aplikace je schopna periodicky zjišťovat polohu a tu následně odesílat webové službě, nebo při nedostupnosti serveru ji ukládat lokálně. Aplikace dokáže běžet neustále, neboť je implementována ve formě služby. Pro lepší přehlednost a možnost měnit nastavení aplikace bylo implementováno grafické uživatelské rozhraní. Aplikace disponuje různými možnostmi nastavení jako například nastavení intenzity využití GPS, Wifi a nastavení intervalu mezi měřeními a vzdálenosti od poslední známé lokace, přičemž vzdálenost spíše ovlivňuje, jaká data se budou archivovat. Aplikace používá pro zjištění polohy GPS, Wifi připojení, síť GSM a případně data ostatních běžících aplikací. Od původně zamýšleného použití senzorů bylo odstoupeno z důvodu nevhodnosti použití v tomto systému, jak je popsáno podrobněji v kapitole 4.2.1.

Aplikace obsahuje navržený algoritmus hledání polohy, který se snaží v závislosti na nastavení být úsporný vůči spotřebě energie a zároveň ještě poskytovat relevantní informace. Dále je obsažen test prospěšnosti nové pozice, který rozhoduje o provedení archivace na server, jelikož například není potřeba ukládat data ze stejné pozice vícekrát. Pokud tedy archivované záznamy vykazují větší časový odstup mezi jednotlivými záznamy, znamená to, že se uživatel po tuto dobu nacházel na udaném místě.

Současná verze aplikace nepodporuje ověření uživatele vůči serveru a podmínkou pro úspěšný běh je přesné zadání stejného uživatelského jména jaké je definováno ve webové službě. Proto je také podmínkou, aby byl uživatel vytvořen před použitím mobilní aplikace. Toto by bylo možné do výsledné verze doplnit, ale z hlediska tématu práce tato část nebyla až tak podstatná. Aplikace není finální a obsahuje chyby, které se zatím nepodařilo opravit. Příkladem je neúmyslné ponechání zapnutého Wifi připojení či uložení redundantních dat, takže by se dalo na aplikaci dále pracovat.

Důkazem toho jsou možné funkce, o které lze aplikaci rozšířit. Kromě výše uvedeného by aplikace mohla poskytovat práci s nastavením reakce na vstup do či výstup z určité oblasti, což se v této problematice označuje jako geofence. Dále by aplikace mohla upozorňovat uživatele například na překročení rychlosti nebo ztrátu komunikace se serverem. Webová služba by naopak mohla implementovat mapy a zobrazovat na nich trasu pohybu uživatelů. Toto jsou pouze návrhy a jistě by bylo možné nalézt spoustu dalších užitečných funkcí a vylepšení. Například algoritmus hledání polohy by se nejspíše mohl vylepšovat neustále.

Podle měření dokáže aplikace při nízkém nastavení vydržet teoreticky až přes pět dní fungovat. Tudíž nijak výrazně neovlivňuje výdrž samotného zařízení, což bylo jednou z požadovaných vlastností.

## Seznam použité literatury

- [1] Android - API Guides. Developer Android [online]. 2014 [cit. 2014-05-15 ].  
Dostupné z: <http://developer.android.com/guide/components/index.html>.
- [2] VOGEL, Lars. VOGELLA.COM. *Android Development* [online]. [cit. 2014-05-15].  
Dostupné z: <http://www.vogella.com/tutorials/android.html>
- [3] MEIER, Reto. A Deep Dive Into Location. In: *Android Developers Blog* [online]. 2011 [cit. 2014-05-15]. Dostupné z: <http://android-developers.blogspot.cz/2011/06/deep-dive-into-location.html>
- [4] CHON, Yohan, Elmurod TALIPOV, Hyojeong SHIN a Hojung CHA. Mobility prediction-based smartphone energy optimization for everyday location monitoring. *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems - SenSys '11*. New York, New York, USA: ACM Press, 2011, s. 82-95.  
DOI: 10.1145/2070942.2070952. Dostupné z: <http://dl.acm.org/citation.cfm?doid=2070942.2070952>
- [5] Good way of getting the user's location in Android. In: *Stackoverflow* [online]. [cit. 2014-05-15]. Dostupné z: <http://stackoverflow.com/questions/6181704/good-way-of-getting-the-users-location-in-android>
- [6] SACHS, David. Sensor Fusion on Android Devices: A Revolution in Motion Processing. [video]. GoogleTechTalks. [online]. [cit. 2014-05-15].  
Dostupné z: <http://www.youtube.com/watch?v=C7JQ7Rpwn2k>
- [7] Techniques to improve the GPS precision. *International Journal of Advanced Computer Science and Applications* [online]. 2013, roč. 3, č. 8 [cit. 2014-05-15].  
Dostupné z: [http://thesai.org/Downloads/Volume3No8/Paper\\_20-Techniques\\_to\\_improve\\_the\\_GPS\\_precision.pdf](http://thesai.org/Downloads/Volume3No8/Paper_20-Techniques_to_improve_the_GPS_precision.pdf)
- [8] GPS.GOV [online]. [cit. 2014-05-15]. Dostupné z: <http://www.gps.gov/>
- [9] MCNAMARA, Joel. GPS for dummies. 2nd ed. Hoboken, N.J.: Wiley, 2008.  
ISBN 04-704-5785-6.
- [10] MITCHELL, Katarina. Background Location Updates On Android. In: *UNCORKED STUDIOS* [online]. [cit. 2014-05-15].  
Dostupné z: <http://uncorkedstudios.com/blog/background-location-updates-on-android>
- [11] *Grails* [online]. [cit. 2014-05-15]. Dostupné z: <https://grails.org/>
- [12] *OpenShift* [online]. [cit. 2014-05-15]. Dostupné z: <https://www.openshift.com/>
- [13] Spring Security Core Plugin. [online]. [cit. 2014-05-15]. Dostupné z: <http://grails-plugins.github.io/grails-spring-security-core/docs/manual/>

- [14] VAUGHAN-NICHOLS, Steven J. How Google--and everyone else--gets Wi-Fi location data. *ZDNet* [online]. 2011 [cit. 2014-05-15].  
Dostupné z: <http://www.zdnet.com/blog/networking/how-google-and-everyone-else-gets-wi-fi-location-data/1664>
- [15] ZAHRADNIK, Fred. Wi-Fi Positioning System. *About.com* [online]. [cit. 2014-05-15].  
Dostupné z: [http://gps.about.com/od/glossary/g/wifi\\_position.htm](http://gps.about.com/od/glossary/g/wifi_position.htm)
- [16] Android Smartphones with Accelerometers Which Work with the Screen Off [online]. 2012 [cit. 2014-05-15]. Dostupné z: <http://www.saltwebsites.com/2012/android-accelerometers-screen-off>
- [17] Garmin. *What is GPS* [online]. [cit. 2014-05-15].  
Dostupné z: <http://www8.garmin.com/aboutGPS/>



## **A Obsah přiloženého CD**

Přiložené CD obsahuje čtyři adresáře.

- AndroidTracker
- Grailsapp
- Instalace Android
- Docs

Android Tracker obsahuje projekt mobilní aplikace vytvořený v Android Studiu. Grailsapp je projekt webové služby vytvořený v prostředí IntelliJ IDEA připravený pro nasazení na openshift. Instalace Android obsahuje instalaci mobilní aplikace v souboru .apk. Mobilní aplikace je určena pro Android ve verzi 3.0 a vyšší. Poslední adresář s názvem Docs obsahuje vlastní text práce a zadání ve formátu PDF.