

# **Aplikace GPS pro mobilní zaměstnance**

A GPS Application for Mobile Workers

Bc. Jan Tichý

---

Diplomová práce  
2012



Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky

---

Univerzita Tomáše Bati ve Zlíně  
Fakulta aplikované informatiky  
akademický rok: 2011/2012

## ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Jan TICHÝ**  
Osobní číslo: **A10714**  
Studijní program: **N 3902 Inženýrská informatika**  
Studijní obor: **Informační technologie**

Téma práce: **Aplikace GPS pro mobilní zaměstnance**

Zásady pro vypracování:

1. Prozkoumejte existující open source mobilní aplikace GPS umožňující tvorbu knihy jízd včetně generování reportů a statistik.
2. Navrhněte architekturu vlastní aplikace pro OS Android.
3. Navrhněte inteligentní algoritmy pro inferenci významných průchozích bodů cesty včetně jejich pojmenování.
4. Implementujte uživatelské rozhraní pro generování reportů z jednotlivých cest.
5. Implementujte generování statistik z jednotlivých cest (např. maximální rychlosti, průměrná doba zastávky, počet soukromých zastávek atd.).

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. L. MURPHY, Mark. **Android 2: průvodce programováním mobilních aplikací**. Brno: Computer Press, 2011. ISBN 978-80-251-3194-7.
2. PECINOVSKÝ, Rudolf. **Myslíme objektivě v jazyku Java: kompletní učebnice pro začátečníky**. 2. Praha: Grada, 2009. ISBN 978-80-247-2653-3.
3. KEOGH, James Edward. **Java bez předchozích znalostí: průvodce pro samouky**. Brno: Computer Press, 2005. ISBN 8025108392.
4. SHARON ZAKHOUR A KOLEKTIV. **Java 6: výukový kurz**. Brno: Computer Press, 2007. ISBN 978-80-251-1575-6.
5. RAPANT, Petr. **Družicové polohové systémy**. Ostrava: VŠB - TU Ostrava, 2002. ISBN 80-248-0124-8.

Vedoucí diplomové práce:

**Ing. Tomáš Dulík**

Ústav informatiky a umělé inteligence

Datum zadání diplomové práce:

**24. února 2012**

Termín odevzdání diplomové práce:

**21. května 2012**

Ve Zlíně dne 24. února 2012

prof. Ing. Vladimír Vašek, CSc.  
*děkan*



doc. Mgr. Roman Jašek, Ph.D.  
*ředitel ústavu*

## **ABSTRAKT**

Tato práce se zabývá problematikou záznamu polohy pomocí GPS s praktickým využitím pro mobilní zaměstnance. Obsahem práce je návrh a implementace nativní aplikace pro chytrá mobilní zařízení s operačním systémem Android a GPS modulem. Výsledný program by měl umožnit tvorbu knihy jízd včetně generování reportů a základních statistik.

V teoretické části práce jsou popsány vlastnosti, komponenty a historie platformy Android. Dále jsou zde představeny principy navigačního systému GPS.

Praktická část popisuje vlastní řešení vzniklé aplikace pojmenované – Mobile Workers. Tato část obsahuje analýzu podobného softwaru, uživatelskou příručku a implementační dokumentaci k aplikaci.

Klíčová slova: GPS, Android, Google Maps API, reverzní geokódování, kniha jízd, záznam jízdy

## **ABSTRACT**

This thesis deals with a recording location via GPS with a practical usage for mobile workers. The content of this thesis is design and implementation of an original application for smart mobile equipment using Android and GPS module. The resulted application should allow creating a route book, generating reports and basic statistics included.

There are described basic characteristics, components and Android platform history in the theoretical part. There are also presented the principles of GPS system in this part.

There is described an original solution of the new application called – Mobile workers. This part includes an analysis of similar software, user manual and implementation documentation for the application.

Keywords: GPS, Android, Google Maps API, reverse geocoding, log book, journey log, book of trips, GPS tracking

Rád bych poděkoval vedoucímu mé diplomové práce Ing. Tomáši Dulíkovi za jeho ochotu, rady a připomínky, které mi pomohly při vypracování diplomové práce.

**Prohlašuji, že**

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

**Prohlašuji,**

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....  
podpis diplomanta

**OBSAH**

<b>ÚVOD</b> .....	<b>9</b>
<b>I TEORETICKÁ ČÁST</b> .....	<b>10</b>
<b>1 OPERAČNÍ SYSTÉM ANDROID</b> .....	<b>11</b>
1.1 HISTORIE.....	11
1.2 VRSTVY OPERAČNÍHO SYSTÉMU.....	12
1.3 HLAVNÍ KOMPONENTY SYSTÉMU .....	14
1.3.1 Activity (aktivity).....	14
1.3.2 Content providers (poskytovatelé obsahu).....	14
1.3.3 Services (služby).....	16
1.3.4 Intents (záměry).....	16
1.4 PŘEHLED VLASTNOSTÍ A NÁSTROJŮ.....	17
1.5 SLUŽBY URČUJÍCÍ ZEMĚPISNOU POLOHU .....	18
<b>2 VÝVOJOVÉ PROSTŘEDKY</b> .....	<b>20</b>
2.1 TECHNOLOGIE JAVA .....	20
2.2 ECLIPSE, SDK A ADT .....	21
<b>3 TECHNOLOGIE GPS</b> .....	<b>23</b>
3.1 POPIS SYSTÉMU.....	23
3.1.1 Kosmický segment.....	23
3.1.2 Řídící segment .....	24
3.1.3 Uživatelský segment.....	25
3.1.4 Signály .....	26
3.1.5 WGS-84.....	27
3.1.6 Time To First Fix a A-GPS .....	27
<b>II PRAKTICKÁ ČÁST</b> .....	<b>28</b>
<b>4 SROVNÁNÍ DOSTUPNÝCH APLIKACÍ</b> .....	<b>29</b>
4.1 TMCZ DRIVEBOOK.....	29
4.2 AUTO MILEAGE LOG.....	31
4.3 TRAVEL LOGBOOK.....	32
4.4 MY CAR TRACKS.....	33
4.5 ANDICAR .....	34
4.6 ZÁVĚR ANALÝZY .....	36
<b>5 UŽIVATELSKÁ DOKUMENTACE</b> .....	<b>38</b>

5.1	POŽADAVKY A INSTALACE APLIKACE .....	38
5.2	PRVNÍ SPUŠTĚNÍ APLIKACE .....	39
5.3	HLAVNÍ OBRAZOVKA .....	40
5.4	MENU .....	42
5.5	NASTAVENÍ .....	43
5.6	SCÉNÁŘ - VYTVOŘENÍ ZÁZNAMU JÍZDY .....	45
<b>6</b>	<b>IMPLEMENTAČNÍ DOKUMENTACE .....</b>	<b>49</b>
6.1	POUŽITÉ TECHNOLOGIE .....	49
6.2	ARCHITEKTURA APLIKACE MOBILE WORKERS .....	50
6.3	ADRESÁŘOVÁ STRUKTURA PROJEKTU .....	52
6.3.1	Layout .....	53
6.3.2	Menu .....	54
6.3.3	Preferences .....	54
6.4	NÁVRH A POPIS TŘÍD .....	55
6.4.1	Třída Map .....	56
6.4.2	Třída GPSManager .....	59
6.4.3	Třídy MyDatabaseManager a MyDbHelper .....	61
6.4.4	Třída Measure .....	62
6.4.5	Třída Tracks a TrackReport .....	62
6.4.6	Třída NewTrackActivity a WebViewActivity .....	64
6.4.7	Třída EditOdometer a OdometerActivity .....	65
6.4.8	Třída EditPreferences .....	65
6.5	DATABÁZE .....	65
6.5.1	Návrh a popis .....	66
6.6	POUŽITÉ KOMPONENTY, KNIHOVNY A SLUŽBY .....	68
6.6.1	Integrace Google Maps .....	68
6.6.2	Reverzní geokódování .....	69
6.7	PROBLÉMY PŘI IMPLEMENTACI .....	70
	<b>ZÁVĚR .....</b>	<b>72</b>
	<b>ZÁVĚR V ANGLIČTINĚ .....</b>	<b>73</b>
	<b>SEZNAM POUŽITÉ LITERATURY .....</b>	<b>74</b>
	<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK .....</b>	<b>76</b>
	<b>SEZNAM OBRÁZKŮ .....</b>	<b>78</b>
	<b>SEZNAM PŘÍLOH .....</b>	<b>80</b>

## ÚVOD

V současné době zaznamenává trh v oblasti chytrých mobilních telefonů obrovský růst. Během několika málo let se tato mobilní platforma dříve určená spíše pro odbornou veřejnost nabízející produkty z high-end sféry transformovala na produkt určený pro masový trh. To bylo podpořeno cenovou dostupností smartphonů pro běžného uživatele, ale podstatnou roli hraje především stále se zvětšující množství aplikací a služeb pro operační systémy těchto telefonů. Velkým předpokladem pro vývoj aplikací se stává skutečnost, že v budoucnu budou lidé stále častěji přistupovat ke službám založených na připojení k Internetu prostřednictvím mobilních zařízení.

Chytré mobilní telefony často obsahují zabudované GPS přijímače, které poskytují prostor pro vznik nových typů aplikací využívajících nějakým způsobem informaci o poloze zařízení, respektive jejich uživatelů. Ačkoli se GPS služby obvykle využívají především pro potřeby navigace a získání informací o cestě k cíli, lze s údaji o zeměpisné poloze naložit i jiným způsobem. Je možné například vytvářet aplikace zobrazující předpověď počasí na základě aktuální zeměpisné polohy zařízení nebo aplikace upozorňující na výskyt přátel v blízkém okolí přihlášených do různých sociálních sítí.

Cílem práce bylo navrhnout a implementovat aplikaci Mobile Workers určenou pro zaměstnance na cestách, která by měla umožňovat pořizování záznamů z jednotlivých služebních či soukromých cest automobilem pomocí GPS. Aplikace by měla inteligentně řešit celý proces od zahájení jízdy, přes identifikaci, pojmenování a zobrazení významných průchozích bodů cesty, až po její vyhodnocení ve formě přehledného reportu.

Práce je rozdělená na teoretickou a praktickou část. První kapitola popisuje historii, strukturu a hlavní komponenty operačního systému Android. Jsou zde představeny základní kameny aplikačního rámce, který umožňuje opakované znovupoužití komponent a přístup k velkému množství služeb systému Android využitelných při vývoji aplikace. Druhá kapitola popisuje vývojové prostředky, nástroje a technologie použité při realizaci aplikace. Třetí kapitola se zabývá principy družicového navigačního systému GPS. Jsou zde uvedeny jednotlivé funkční části systému GPS. Praktickou část otevírá čtvrtá kapitola, ta srovnává vybrané aplikace dostupné zdarma nebo jako open-source software, jež jsou vhodné pro účel tvorby knihy jízd nebo poskytují nějaký záznam z jízdy. Další kapitoly se již zabývají popisem jednotlivých částí vzniklé aplikace. Obsahují uživatelský manuál a implementační dokumentaci.

## **I. TEORETICKÁ ČÁST**

## 1 OPERAČNÍ SYSTÉM ANDROID

V poslední době se velice rychle rozvíjí trend vývoje aplikací pro mobilní zařízení, ať jde o tablety nebo tzv. chytré telefony. Tento rozvoj jde ruku v ruce s nárůstem jejich prodeje, díky dostupnosti služeb využitelných pro běžného uživatele. V současné době existuje několik mobilních platform s různými operačními systémy. Mezi nejvýznamnější patří např.:

- iOS od Apple Inc. určený pro telefon iPhone či tablet iPad.
- Windows Phone 7 společnosti Microsoft pro široké spektrum telefonů.
- Symbian – systém společnosti Nokia, který je na ústupu. Další telefony Nokia budou vybaveny systémem Windows Phone 7.
- BlackBerry společnosti RIM. Velice rozšířená platforma v USA.
- Bada společnosti Samsung. Otevřený systém postavený na linuxovém jádře.
- Android – otevřená platforma, za kterou stojí mnoho výrobců pod záštitou Open Handset Alliance.

Tato práce se zabývá vývojem aplikace využívající systém GPS na platformě Android, proto bude v několika dalších podkapitolách tento operační systém blíže představen. Bude popsána jeho historie, architektura, ale především jeho hlavní komponenty aplikačního frameworku využitelné při vývoji aplikací.

### 1.1 Historie

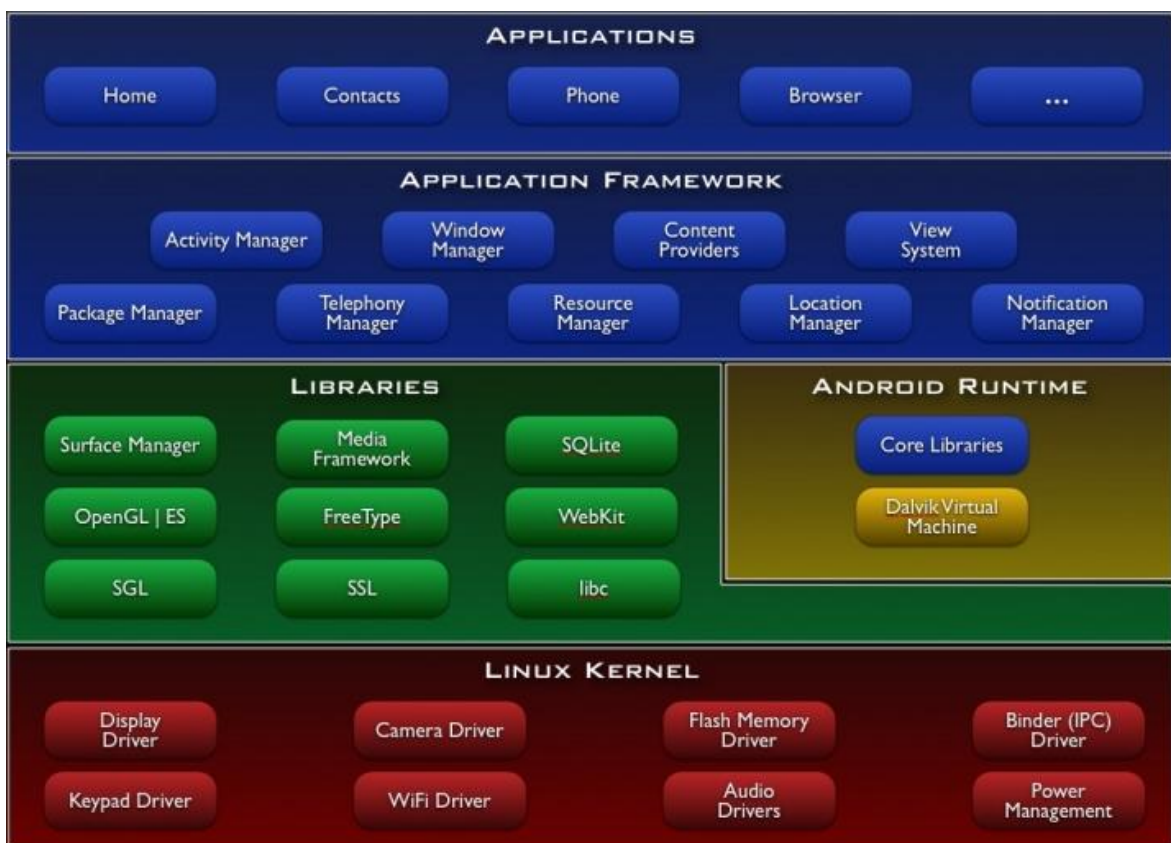
Společnost Android Inc. byla založena v Kalifornii v roce 2003. Google Inc. odkoupil v roce 2005 tento v té době nepříliš známý projekt a udělal s firmy Android Inc. svoji dceřinou společností. [1]

Tým pod vedením tehdejšího spoluzakladatele firmy Android Inc. Andyho Rubina vyvinul platformu založenou na Linuxovém jádře a získal několik patentů v oblasti mobilních technologií. V roce 2007 se k projektu připojilo několik dalších firem. Bylo vytvořeno konsorcium Open Handset Alliance, které zahrnovalo uskupení výrobců mobilních telefonů, mikroprocesorů, telekomunikačních operátorů a technologických firem stojících za vývojem operačního systému Android. Mezi nejvýznamnější společnosti podporující tento systém patří kromě lídra Google např. HTC, Intel, LG, Motorola, T-Mobile, China Mobile, Sony Ericsson,

a další. Cílem bylo vyvinout otevřenou platformu pro mobilní „chytrá“ zařízení postavenou na linuxovém jádře verze 2.6. [2]

## 1.2 Vrstvy operačního systému

Operační systém Android je složen z několika vrstev zobrazených na Obr. 1. Každá z vrstev se specializuje na jiné úlohy a poskytuje své služby vrstvě o úroveň výš přes specifikované rozhraní. Všechny vrstvy společně tvoří architekturu systému.



Obr. 1. Vrstvy operačního systému Android [3]

### Vrstva – Linux Kernel (jádro systému):

Nejnižší vrstva obsahuje jádro operačního systému Linux ve verzi 2.6. V podstatě tvoří abstraktní vrstvu systému mezi hardwarem a dalším softwarem ve vyšších vrstvách. Řešení postavené na Linuxu bylo použito pravděpodobně kvůli velké flexibilitě, otevřenosti tohoto systému a tím snadné přenositelnosti na různá zařízení. Je využito mnoho jeho vlastností, jako jsou mechanismy pro správu operační paměti, procesů, zabudované ovladače, a něco je přizpůsobené přímo pro mobilní platformu. Například zde z principu nemůže existovat

swapování, tak jak ho známe u stolních počítačů. Vše je optimalizované pro malé množství operační paměti, kapacity baterie a slabé CPU. Naopak systém neobsahuje obsáhlé X Window System, ani úplnou sadu GNU knihoven.

#### **Vrstva – Libraries (vrstva knihoven):**

Další vrstvu zahrnují různé knihovny napsané v jazyce C/C++, které zajišťují funkcionalitu systému při práci s médii, databází, 3D OpenGL grafikou, webovým prohlížečem atd. Tyto knihovny zároveň zpřístupňují všechny funkce vývojářům prostřednictvím aplikačního frameworku, který je umístěn nad touto vrstvou knihoven.

#### **Vrstva – Android Runtime (běhové prostředí):**

Na stejné úrovni se nachází vrstva Android Runtime, která zajišťuje běhové prostředí pro kód kompilovaný do Dalvik byte kódu. Vlastní provádění kódu vykonává Dalvik Virtual Machine (DVM). Celé je to založené na podobném principu jako v Javě, jejíž syntaxe a některé knihovny jsou použity k vlastnímu programování Android aplikací. Java je více popsána v kapitole 2.1. Ať už se runtime prostředí více či méně podobá JVM, faktem zůstává to, že Dalvik byte kód je mnohem úspornější v oblasti poměru úspory energie a výkonu. Je optimalizovaný pro běh na mobilních zařízeních.

V této vrstvě jsou také obsaženy základní knihovny programovacího jazyka Java a svým obsahem se přibližují platformě Java Standard Edition. Jsou vypuštěny některé rozsáhlé knihovny pro tvorbu uživatelského rozhraní, místo kterých je zde vlastní sada tříd pro tvorbu GUI a na druhou stranu jsou přidány vybrané knihovny například knihovna Apache. Překlad aplikace napsané pro Android probíhá zkompileováním zdrojového Java kódu do Dalvik byte kódu pomocí kompilátoru, jako je používán při překladu Java aplikací. Potom je samotný výstup z Java kompilátoru ještě automaticky překompilován do výsledného Dalvik byte kódu, jenž používá vlastní formát kompilovaných souborů DEX (Dalvik EXecutable). V poslední fázi je kód spuštěn na DVM. Každá běžící aplikace je umístěná ve své vlastní instanci DVM jako samostatný proces operačního systému.

#### **Vrstva – Application Framework (aplikační rámeček):**

Nejdůležitější vrstva pro vývojáře Android aplikací se jmenuje Aplikační Framework. Přes jeho jednotné API (aplikační programové rozhraní) je poskytován přístup k nejrůznějším službám, prvkům uživatelského rozhraní, hardwaru zařízení a dalším funkcím. Framework je sadou velkého množství tříd a komponent, které tvoří základní kameny pro vývoj aplikací.

Některé z hlavních komponent, z nichž se aplikace pro systém Android skládají, budou uvedeny dále v této práci.

Na poslední vrstvě se nachází už samotné spustitelné aplikace, které využívají běžní uživatelé. Například může jít o aplikaci na obsluhu telefonních hovorů, SMS program, kontakty, e-mailový klient, a další aplikace.

### 1.3 Hlavní komponenty systému

Základní stavební kameny v aplikacích Android jsou komponenty *Activity* reprezentující obrazovky, *Service* umožňující provádět operace na pozadí, *Content providers* poskytující přístup k datům více různým aplikacím napříč systémem a *Intents* fungující jako systémové zprávy určené k notifikaci různých událostí. Klíčovou součástí projektu aplikace je soubor *AndroidManifest.xml*, který obsahuje „tabulku komponent“ všech použitých součástí a práv. [4]

#### 1.3.1 Activity (aktivity)

Stavebními kameny pro tvorbu uživatelského rozhraní jsou aktivity. Aktivity jsou jakousi analogií oken či dialogů aplikace pro stolní počítač. Aktivity mohou být i bez uživatelského rozhraní, ale většina kódu bez uživatelského rozhraní bude ve formě *Content providers* nebo *Services*. Samotná aktivita reprezentuje jednu uživatelskou obrazovku, jejíž rozložení pomocí kontejnerů (layout) a GUI prvky by měly být definovány v XML zdrojích aplikace. Za celý proces vytváření, rušení a správu životního cyklu aktivity zodpovídá systémový *Activity Manager*. Životní cyklus aktivity se může nacházet v jednom ze stavů zobrazených na Obr. 2.

#### 1.3.2 Content providers (poskytovatelé obsahu)

Dodavatelé obsahu zajišťují úroveň abstrakce jakýchkoliv dat uložených v zařízení, která jsou přístupná více různým aplikacím. Vývojový model systému Android tímto podporuje to, aby vývojáři zpřístupnili svá data i jiným aplikacím než pouze své vlastní. Toho může být dosaženo právě vytvořením dodavatele obsahu, který zároveň poskytuje kontrolu nad způsobem přístupu k datům aplikace.



### 1.3.3 Services (služby)

Aktivity a dodavatelé obsahu jsou komponenty s krátkou dobou životností a lze je kdykoliv vypnout. Služby jsou navrženy tak, aby mohly běžet na pozadí a pokračovat v činnosti nezávisle na spuštěné aktivitě. Službu lze například využít ke stahování dat z internetu, aktualizaci stavů různých klientů, přehrávání hudby, apod.

### 1.3.4 Intents (záměry)

Záměry jsou systémové zprávy upozorňující aplikace na výskyt různých událostí, změnami hardwarové konfigurace (například vložení SD karty) počínaje přes události související s přichozími daty (například přijetí SMS zprávy) až událostmi aplikace konče (například její spuštění z hlavního menu zařízení). V podstatě záměry zapouzdřují požadavky vznesené vůči systému, které vyžadují, aby nějaká aktivita nebo jiný příjemce záměru (*BroadcastReceiver*) nějak reagovali. Teoreticky vzato na záměrech a jejich příjemcích celý systém Android stojí. Proto budou představeny podrobněji. [6]

Nejdůležitějšími dvěma součástmi záměrů jsou požadovaná akce, případně operace, kterou chceme vykonat a nějaká data. Celý systém záměrů se dost podobá principům HTTP protokolu. Akce jsou konstanty například *ACTION\_VIEW* (prohlížení), *ACTION\_EDIT* (editace), apod. Data reprezentuje třída *Uri* například s hodnotou:

```
content://contacts/id_1/
```

Vlastní záměr reprezentuje objekt typu *Intent*, do které je možné uložit krom instance třídy *Uri* a příslušné akce také další kritéria, například následující: [6]

- Kategorie – Hlavní aktivity patří do kategorie *LAUNCHER*, což znamená, že budou umístěny v menu spouštěče aplikací. Ostatní patří do kategorií *DEFAULT* nebo *ALTERNATIVE*
- MIME typ – Specifikuje typ prostředku, nad kterým se bude operovat, pokud nejsou známá vlastní data.
- Komponent - Jedná se o specifikaci třídy nějaké aktivity, která má daný záměr přijmout.
- Doplnky – Představuje objekt typu *Bundle* obsahující další přibalená data, která jsou předána příjemci záměru k možnému využití.

Rozdělení záměrů podle směřování:

- Implicitní – se používá v případě, pokud cílový komponent (aktivity, příjemce záměrů) není specifikován při odesílání Intentu. Android musí sám přijít na to, která komponenta je k přijetí daného záměru vhodná. Aktivita musí splňovat určitá pravidla, aby mohla být klasifikována jako vhodný cíl pro daný záměr. Musí podporovat specifikovanou akci, uvedený MIME typ, a patřit do uvedené kategorie.
- Explicitní – se stává typem směřování v případě, kdy je v záměru specifikován konkrétní cílový komponent, kam se má záměr směřovat. Tento přístup je vhodný pokud se cílový záměr nachází uvnitř vlastní aplikace.

Všechny komponenty systému Android, které se chtějí stát potencionálními cíli záměrů, musí deklarovat tzv. filtry záměrů, podle kterých se systém rozhoduje o případném směřování na tyto komponenty. To je nutné zajistit přidáním elementu *intent-filter* do souboru *AndroidManifest.xml*.

Příklad filtru deklarující hlavní aktivitu je vidět na Obr. 3.

```
20 <activity
21     android:name=".Map"
22     android:label="@string/app_name" >
23     <intent-filter>
24         <action android:name="android.intent.action.MAIN" />
25         <category android:name="android.intent.category.LAUNCHER" />
26     </intent-filter>
27 </activity>
28
```

Obr. 3. Intent filtr aktivity

Výše uvedený příklad filtrů záměrů přivázaných k aktivitám se nemusí hodit ve všech případech použití. Například při spuštění služby zajišťující nějakou činnost, od které je očekáván příjem dat na základě vzniklých událostí. Za tímto účelem Android nabízí přijímač záměrů implementovaný jako rozhraní *BroadcastReceiver*. Přijímače záměrů jsou navrženy tak, aby přijímaly konkrétně směřované záměry a prováděly s nimi nějakou akci.

## 1.4 Přehled vlastností a nástrojů

- Úložiště: Android nabízí možnost přidání datových souborů do aplikace, jejichž obsah se nebude měnit. Může jít například o ikony, pomocné soubory a jiné zdroje. Dále je k dispozici vlastní zabudovaná databáze SQLite, kterou lze využít k uložení uživatelem zadávaných dat či načtení potřebných aplikačních dat.

- **Sít:** Zařízení s Androidem jsou vhodná pro on-line použití s potřebou komunikace přes Internet prostřednictvím různých druhů komunikačních médií. Vývojáři aplikací mohou využít při programování jakékoliv úrovně komunikace od Java socketů, přes webové služby založené na architektuře REST až po vysokoúrovňový přístup s využitím zabudovaného jádra WebKit.
- **Multimédia:** Zařízení využívající systém Android umí přehrávat a zaznamenávat audio – videozáznamy.
- **Global positioning system (GPS):** Zařízení často obsahují integrované GPS čipy a mohou proto získat přístup k dodavatelům údajů o zeměpisné poloze. Díky této funkcionalitě lze vytvářet aplikace využívající tuto informaci ke sledování pohybu zařízení, zobrazení polohy zařízení na mapě a dalším činnostem. Tato práce se zabývá využitím právě systému GPS v zařízeních s operačním systémem Android, a proto bude tato oblast více popsána v následující kapitole. Samotná technologie GPS bude podrobněji rozebrána v kapitole pojednávající o GPS obecně.
- **Telefonní služby:** Zařízení s Androidem jsou převážně telefony, takže lze naprogramovat i software využívající telefonická volání, zasílání a příjem SMS či MMS zpráv a jakékoliv další operace s tím spojené. [6]

## 1.5 Služby určující zeměpisnou polohu

V dalším textu bude stručně popsán princip přístupu ke službám určujícím zeměpisnou polohu v zařízeních se systémem Android 2. [6], [7]

V systému je možné využít několik prostředků k určení zeměpisné polohy. Některé jsou přesnější než jiné, další dokážou poskytnout i jiné informace než pouze specifikaci zeměpisné polohy zařízení, například údaje o nadmořské výšce, aktuální rychlost, přesnost a další charakteristiky. Všechny tyto prostředky jsou reprezentovány množinou objektů typu *LocationProvider*.

Při vývoji aplikace se pro rozhodování o tom, který prostředek bude použit, využívá objekt typu *LocationManager*.

Existují dva způsoby pro určení prostředku k lokalizaci zeměpisné polohy:

- Přímým určením objektu *LocationProvider*, například zapsáním konstanty *GPS\_PROVIDER* přímo do zdrojového kódu aplikace.
- Nalezením nejvhodnějšího kandidáta na základě sady kritérií vytvořené pomocí objektu *Criteria*. Objekt je nutné nastavit voláním setterů. Příkladem může být metoda *setAccuracy()*, jenž nastavuje požadovanou přesnost. Po zaplnění tohoto objektu se zavolá metoda *getBestProvider()* objektu typu *LocationManager*, kterému se předá jako parametr objekt typu *Criteria*. Android projde kritéria a vrátí nejvhodnějšího poskytovatele.

V první řadě musíme získat objekt typu *LocationManager* viz Obr. 4. To zajistíme tím, že z aktivity nebo služby zavoláme metodu *getSystemService(LOCATION\_SERVICE)*. Systém vrátí výsledek, který přetypujeme na typ *LocationManager*. Dále musíme nastavit objekt typu *LocationProvider* viz výše.

Nejvýznamnějším objektem poskytující zeměpisné a další údaje je objekt typu *Location*. Typickým způsobem získávání objektů typu *Location* od objektů typu *LocationProvider* v pravidelných intervalech je možnost zaregistrování aktualizací následujícím postupem.

V prvním kroku zavoláme metodu *requestLocationUpdates()* naší instance třídy *LocationManager*. Metoda přijímá celkem čtyři parametry: Název dodavatele údajů o zeměpisné poloze, časový interval mezi aktualizacemi (v milisekundách), uraženou vzdálenost před zasláním aktualizací (v metrech) a objekt typu *LocationListener*. Poslední zmíněný objekt bude dostávat upozornění na události se změnou zeměpisné polohy.

```
// Acquire a reference to the system Location Manager
LocationManager locationManager = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);

// Define a listener that responds to location updates
LocationListener locationListener = new LocationListener() {
    public void onLocationChanged(Location location) {
        // Called when a new location is found by the network location provider.
        makeUseOfNewLocation(location);
    }

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

    public void onProviderEnabled(String provider) {}

    public void onProviderDisabled(String provider) {}
};

// Register the listener with the Location Manager to receive location updates
locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0, locationListener);
```

Obr. 4. Ukázka implementace objektu *LocationListener* [8]

## 2 VÝVOJOVÉ PROSTŘEDKY

### 2.1 Technologie Java

Za vytvořením Javy stojí společnost Sun Microsystems, která vydala první verzi v roce 1995 pod označením Java 1.0. Cílem bylo vytvořit syntakticky jednoduchý jazyk, ve kterém bude možné vytvářet složité projekty za velmi krátké časové období, ovšem ne na úkor kvality.

Součástí technologie Java je programovací jazyk i platforma. V případě Javy je platforma softwarové prostředí pro spouštění aplikací, které je možné provozovat na jakémkoliv hardwarovém zařízení, na němž je instalovaný virtuální stroj jazyka Java - JVM. Virtuální stroj je program napsaný jako nativní program daného zařízení. Dále platforma obsahuje aplikační programové rozhraní – API, představující rozsáhlou sbírku softwarových komponent členěných do knihoven. Knihovny se v Javě nazývají balíčky (packages) a obsahují sadu souvisejících tříd, které je možné importovat do svých aplikací a využít tak jejich funkcionalitu.

Samotný jazyk je charakterizován jeho syntaxí a vlastnostmi vyšších programovacích jazyků. Jedná se o objektově orientovaný, zabezpečený, dynamický a hlavně přenositelný jazyk nezávislý na architektuře hardwarově závislé platformy. Dalším typickým rysem je to, že správu paměti realizuje pomocí tzv. Garbage collectoru, jenž se stará o automatické uvolnění již nepoužívané části paměti. Díky tomu se programátor nemusí starat o uvolnění paměti jako například v jazycích C/C++.

Celou platformu lze dělit do čtyř edic:

- Java SE – Standard Edition označuje standardní verzi Javy určenou především pro tvorbu desktopových aplikací nad řadou operačních systémů. Velká část knihovny této edice se využívá při programování aplikací pro systém Android.
- Java EE – Enterprise Edition je určená pro vývoj distribuovaných systémů, kde aplikace pracují na více spolupracujících počítačích rozprostřených po rozsáhlé síti.
- Java ME – Micro Edition pro mobilní telefony a různá embedded zařízení navržená pro méně výkonné systémy.
- Java Card – platforma pro vývoj čipových karet.

Vlastní proces překladu a spouštění aplikací vytvořených v Javě funguje následujícím způsobem:

Zdrojový kód se ukládá do textových souborů s příponou *.java*. Kompilátor pak tyto zdrojové soubory kompiluje do souborů *.class*. V souboru *.class* nenalezneme nativní kód příslušného procesoru. Místo toho obsahuje bajtový kód (bytecode), což je strojový jazyk Java Virtual Machine. Nástroj spouštění *java* potom spustí aplikaci s instancí JVM. JVM je k dispozici pro mnoho různých operačních systémů. Stejně soubory *.class* lze proto spustit v různých systémech (Windows, Linux, Mac, Solaris). [9]

## 2.2 Eclipse, SDK a ADT

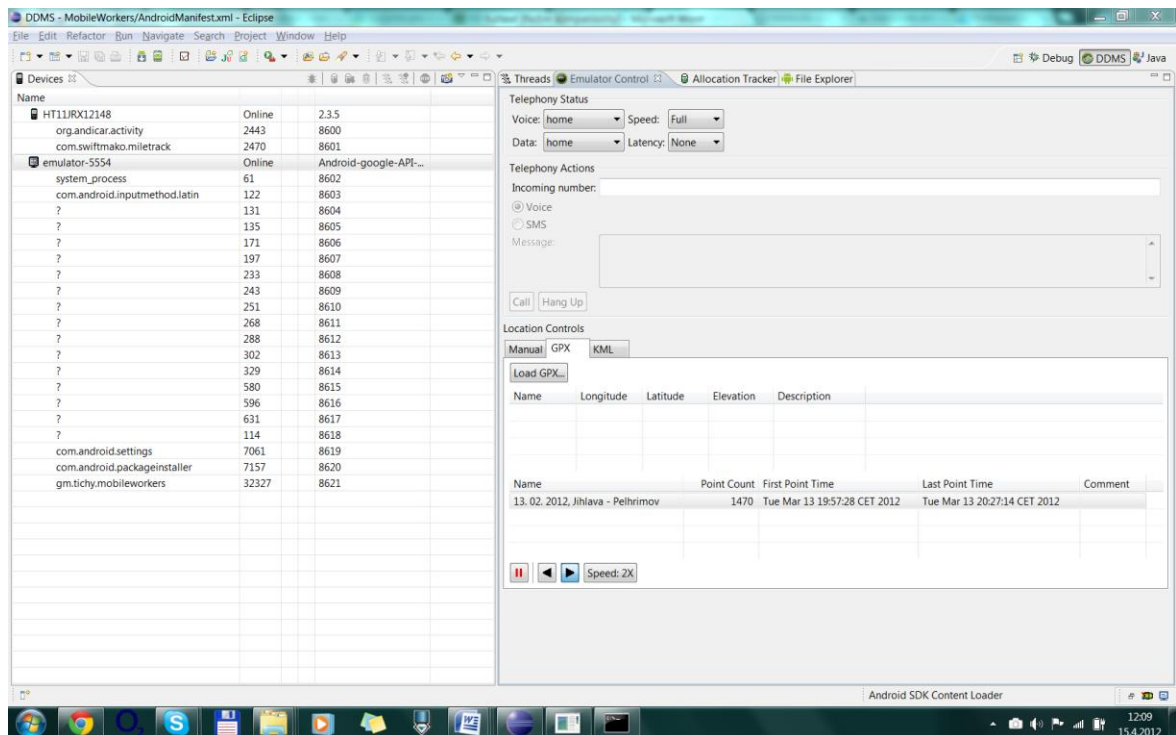
Eclipse je open source vývojové prostředí (IDE) primárně určené pro programování v jazyce Java. V základní verzi obsahuje pouze nezbytné základní části pro vývoj v Javě jako kompilátor nebo debugger. Prostředí je navrženo stylem, který umožňuje základní funkce rozšířit pomocí pluginů a tím zajistit nástroje pro vývoj v dalších programovacích jazycích. Eclipse je také oficiálně podporované prostředí pro vývoj aplikací určených pro Android. Tuto podporu je možné získat po instalaci zásuvného pluginu Android Development Tools (ADT), který výrazně usnadňuje vývoj a ladění aplikací psaných pro Android. ADT je řešení pro pohodlné ovládání nástrojů přístupných v balíku Software Development Kit (SDK) prostřednictvím IDE Eclipse. V tuto chvíli existují i jiná prostředí, ve kterých se pro Android vyvíjí, například IDE NetBeans.

Samotné SDK je v podstatě balík programátorských nástrojů pro systém Android. Celá sada obsahuje mnoho programů, nejdůležitější jsou však emulátor ARM procesorů, knihovny a debugger. Adresář *tools* obsahuje pouze nejnütnější základ balíku SDK. Jde o spouštěč Android SDK, emulátor, AVD a SDK Manager a další knihovny nezávislé na konkrétní verzi systému Android.

Android SDK Manager spravuje jednotlivé komponenty SDK. Vývojář zde má možnost stáhnout různé verze platformy Android a další rozšíření. Každá platforma obsahuje systémové knihovny, systémový obraz, skiny emulátoru typické pro danou verzi. Aby bylo možné začít vyvíjet, je nutné nainstalovat *Platform-tools* a alespoň jednu verzi systému Android. Adresář *platform-tools* zahrnuje další vývojové nástroje, které jsou pravidelně aktualizované za účelem získání nových funkcí.

V AVD Manageru lze vytvářet a spouštět Android Virtual Device, což jsou vývojářem nakonfigurované virtuální zařízení určené k testování aplikací v emulátoru. Při vytváření AVD se musí specifikovat jeho cíl. Cíl indikuje, jakou verzi platformy Android má AVD simulovat, a zároveň je tím také určená úroveň API.

Dalším důležitým programem v SDK je nástroj Dalvik Debug Monitor (DDMS) zobrazen na Obr. 5. Tento nástroj umožňuje komunikovat s běžícími instancemi emulátoru, ale také s fyzickým zařízením připojeným přes USB. DDMS je integrován do prostředí Eclipse díky ADT pluginu. Při vývoji můžeme například v části Emulator Control simulovat některé funkce telefonu například příchozí telefonní hovory, posílání SMS zpráv nebo lokalizaci zařízení dle zeměpisných souřadnic. Dalším velice užitečným nástrojem je LogCat, který umí zobrazovat logy přicházející z celého systému a ladící informace posílané z virtuálního stroje.



Obr. 5. Vývojové prostředí Eclipse – nástroj DDMS

### 3 TECHNOLOGIE GPS

Následující kapitola popisuje principy a části technologie GPS. Při tvorbě textu bylo použito následujících zdrojů: [10], [11]

#### 3.1 Popis systému

Globální družicový polohový systém GPS byl navržen tak, aby umožňoval s omezeným počtem družic pokrýt celý zemský povrch navigačními signály a zajistit tak určování polohy kdekoliv na Zemi. Družice vysílají radiový signál na frekvencích označovaných L1 – L5. Pro navigační účely se využívají frekvence:

L1 = 1575,43 MHz, L2 = 1227,60 MHz. Signály GPS jsou složeny z nosné vlny, navigační zprávy a dálkoměrného kódu. Systém dokáže určit třírozměrnou polohu, rychlost pohybu a přesný čas. GPS lze řadit do skupiny dálkoměrných pasivních družicových navigačních systémů. V podstatě to znamená, že uživatelské přijímače fungují pouze jako příjemci navigačního signálu odeslaného družicí, který následně vyhodnocují. Žádným způsobem se aktivně nezapojují do komunikace s družicovým systémem. Systém GPS provozuje a spravuje ministerstvo obrany USA. Aktuálně by se měla soustava skládat z 32 družic (duben 2012). [12]

Systém GPS je složen ze tří segmentů:

- Kosmický segment
- Řídící segment
- Uživatelský segment

##### 3.1.1 Kosmický segment

Kosmický segment GPS tvoří soustava družic, rozmístěných na oběžných drahách, které vysílají navigační signály. Soustava je tvořena šesti oběžnými drahami se čtyřmi družicemi na každé z nich. Sklon oběžné dráhy vzhledem k rovníku je 55°. Družice oběhne svou dráhu přibližně za 12 hodin. Toto uspořádání je z důvodu zajištění viditelnosti vždy minimálně čtyř družic, z kteréhokoliv místa na Zemi. Tato hodnota je považována jako minimální počet družic, pomocí kterých může přijímač vypočítat svou třírozměrnou polohu určenou zeměpisnou šířkou, délkou a nadmořskou výškou. Ve většině případů je však možné zachytit signál z více družic a tím se výrazně zpřesňuje poloha přijímače.

### 3.1.2 Řídící segment

Řídící segment představuje „mozek“ celého systému. Plní řadu úloh, které zajišťují bezproblémový chod ostatních segmentů. Skládá se ze tří druhů stanic strategicky rozmístěných na velkých amerických vojenských základnách, aby umožňovaly monitorování co největšího počtu družic. Hlavním úkolem řídicí části je přesné nastavení času a pozice jednotlivých družic na oběžné dráze. Správné parametry drah družic a přesný čas jsou základními předpoklady k funkci celého systému. Při poruše některé ze stanic řídicí části GPS je možné použít záložní stanici na Cap Canaveral. Tato stanice jinak slouží ke kompletaci a vypouštění družic do kosmického prostoru.

Stanice jsou rozdělené do čtyř skupin:

- Monitorovací stanice
- Hlavní řídicí stanice
- Komunikační stanice

#### **Monitorovací stanice**

Pozemní monitorující stanice mají za úkol sledovat aktuálně viditelné družice. Přijímají navigační signál s navigačními zprávami, určují zdánlivé vzdálenosti k družicím a přenášejí získaná data do hlavní řídicí stanice ke zpracování.

#### **Hlavní řídicí stanice**

Hlavní řídicí stanice je situována na vojenské základně v Colorado Springs. Zde jsou na základě přijatých výsledků vypočítány přesné údaje oběžných drah (tzv. efemeridy) a korekce hodin jednotlivých družic. Tyto parametry jsou dále předávány komunikačním stanicím, které zajišťují aktualizaci družic. Pozemní GPS přijímače potom vyhodnocují efemeridy oběžných drah družic přenesených pomocí radiových signálů spolu s přesným časem.

#### **Komunikační stanice**

Hlavní činností těchto stanic je v pravidelných intervalech (i několikrát za den) odesílat na družice údaje o jejich oběžných drahách a nastavovat přesný čas.

### 3.1.3 Uživatelský segment

Uživatelský segment obsahuje různá provedení více či méně kvalitních GPS přijímačů. Může jít o automobilové navigace, zabudované GPS čipy v mobilních zařízeních, externí jednoúčelové GPS moduly, speciální GPS přijímače pro oblast geodézie. V této části je velký prostor pro vývoj služeb a aplikací využívající systém GPS. Mezi nejvyužívanější použití běžnými uživateli patří navigace (automobilová, pěší), hra Geocaching, sledovací systémy, apod.

#### Určení polohy

Výpočet polohy GPS přijímače z přijatých signálů získáme řešením soustavy čtyř rovnic 1.1.

$$\begin{aligned}
 r_1 &= \sqrt{(X - x_1)^2 + (Y - y_1)^2 + (Z - z_1)^2} - c \cdot \Delta T \\
 r_2 &= \sqrt{(X - x_2)^2 + (Y - y_2)^2 + (Z - z_2)^2} - c \cdot \Delta T \\
 r_3 &= \sqrt{(X - x_3)^2 + (Y - y_3)^2 + (Z - z_3)^2} - c \cdot \Delta T \\
 r_4 &= \sqrt{(X - x_4)^2 + (Y - y_4)^2 + (Z - z_4)^2} - c \cdot \Delta T
 \end{aligned}
 \tag{1.1}$$

*„Na levé straně rovnic jsou zdánlivé vzdálenosti přijímače k jednotlivým družicím, tak jak byly naměřené.  $X, Y, Z$  jsou souřadnice přijímače, které chceme určit,  $x_i, y_i$  a  $z_i$  jsou souřadnice jednotlivých družic v době měření zdánlivých vzdáleností (získáme je výpočtem z údajů obsažených v navigačních zprávách jednotlivých družic),  $c$  je rychlost světla a  $\Delta T$  je neznámý posun hodin přijímače oproti systémovému času, který chceme rovněž určit. Tyto rovnice musí být řešeny simultánně tak, aby přijímač mohl přímo poskytovat výstup v souřadnicích. Poloha je určována v geocentrických souřadnicích, bývá však zpravidla převáděna do geografických souřadnic. Obecně může být poloha převedena do souřadnic prakticky libovolného kartografického zobrazení.“ [10, str.34]*

### 3.1.4 Signály

K navigačním účelům pomocí GPS se využívají signály přenášené na nosných frekvencích L1 a L2. Obě frekvence jsou odvozeny od tzv. základní frekvence, jejíž hodnota je  $f_0 = 10,23$  MHz. Frekvence L1 může být modulována dvěma kódy. V první řadě jde o přesnější P-kód, který je k dispozici také v zašifrované verzi, potom se označuje jako Y-kód. V druhém případě je to kód C/A určený pro běžné použití. L2 je potom modulována pouze P-kódem.

Kromě těchto dvou tzv. dálkoměrných kódů, jenž jsou vyjádřeny jako pseudonáhodné šumy složené z 0 a 1 (PRN) je přenášen také binární kód, obsahující navigační zprávu. PRN kód zároveň identifikuje jednotlivé družice a musí být tedy pro každou z družic jedinečný.

Navigační zpráva představuje datový rámec dlouhý 1500 bitů, který se dále dělí do pěti podrámců s vlastním obsahem všech údajů z družice. Jak již bylo zmíněno, pro přesné určení polohy přijímače GPS je nezbytně nutné znát polohu vysílající družice v době odeslání dálkoměrného kódu. Ta je vypočítána a odeslána samotnou družicí prostřednictvím právě navigační zprávy, která mimo jiné obsahuje i celou řadu dalších údajů:

- Čas vysílání počátku zprávy
- Přesné keplerovské efemeridy družice
- Údaje umožňující přesně korigovat čas vysílání družice
- Almanach
- Koeficienty ionosférického modelu
- Stav družice

Almanach podává méně přesné informace o parametrech oběžných drah všech družic než přesné efemeridy. Této znalosti se však využívá ke snížení doby pro nastartování GPS přijímače. Pomocí almanachu uloženého v zařízení je přijímač schopný vyhledávat rychleji aktuálně viditelné družice a tím zkrátit dobu pro získání signálu. Pokud od získání prvních údajů uběhla větší zeměpisná či časová vzdálenost aktuální pozice od naposledy zaznamenané, pravděpodobně bude nutné načíst almanach znovu. Almanach má velikost 37500 bitů odesílaný rychlostí 50 bps. Jeho přenos z družice do GPS přijímače bez okolních vlivů tedy trvá minimálně 12,5 minuty.

### 3.1.5 WGS-84

Souřadnicové systémy vychází z fyzikálního modelu tvaru Země tzv. geoidu. Bohužel, tento model není vhodný pro matematický popis, proto se začal využívat model obecného elipsoidu definovaný středem, délkami hlavní a vedlejší poloosy. Pro potřebu zobrazení polohy v souřadnicovém systému jsou aplikovány početní metody, jimiž lze v rovině vytvořit obraz zemského povrchu. Na tomto prostoru je definován souřadnicový systém, ke kterému jsou vztaženy veškeré výpočty.

WGS-84 je Světový geodetický systém vytvořený v roce 1984 definovaný jako souřadnicový systém k určování polohy pro geodézii a navigaci. S tímto typem souřadnicového systému pracuje výstup každého GPS přijímače. Je odvozen ze stejnojmenného elipsoidu. Jeho souřadnice lze převést do jiných souřadnicových systémů, příkladem může být systém S-JTSK používaný v České republice.

### 3.1.6 Time To First Fix a A-GPS

#### **TTFF**

TTFF je doba potřebná pro určení první polohy GPS přijímače po jeho zapnutí. V nejhorším případě je v této fázi provedena kompletní inicializace přijímače, například pokud bylo zařízení resetováno. V ostatních případech doba závisí na aktuálnosti dat almanachu, změně vzdálenosti při vypnutém zařízení, úrovni GPS signálu a době posledního vyhodnocení pořízených dat. Uvádějí se tři typy spuštění, které mohou nastat:

- Teplý start: zařízení předpokládá platnost informací (dat almanachu) uložených ve své paměti, nedošlo k významné změně polohy od posledního spuštění. Závisí i na úrovni signálu – počítá s volným výhledem na oblohu.
- Studený start: odpovídá prvnímu spuštění – inicializaci GPS přijímače.
- Horký start: Nejlepší možná varianta spuštění.

#### **A-GPS**

Většina mobilních zařízení (telefony, notebooky, tablety) mají možnost využít datových přenosů k rychlému získání navigačních dat (almanachu) z asistenčního centra. Technologie A-GPS je nadstavbou pro GPS zařízení, která mají možnost připojení k internetu. Tato cesta je mnohem rychlejší při spuštění, než když musí přijímač stahovat celý almanach pomocí navigačního signálu o velmi pomalé rychlosti.

## **II. PRAKTICKÁ ČÁST**

## 4 SROVNÁNÍ DOSTUPNÝCH APLIKACÍ

Jako hlavní zdroje při hledání vybraných aplikací k analýze a porovnání požadované funkčnosti byly použity zdarma dostupné a open-source aplikace z webových portálů androidmarket.cz, play.google.com, androidzoom.com a code.google.com. Pro platformu systému Android v tuto chvíli existuje více než 400 000 různě kvalitních aplikací z mnoha kategorií. Klíčovou aplikací dodávanou přímo se systémem je Android market (nyní Market Play). Tato aplikace umožňuje uživatelům vyhledávání, výběr, instalaci i následný update již instalovaného softwaru v zařízení, a zároveň slouží jako prostředí, kde mohou vývojáři publikovat své aplikace.

K dispozici je množství automobilových a cestovatelských aplikací využívajících systém GPS k různým účelům. Například jde o aplikace určené:

- k vedení a správě nákladů na servis, údržbu, tankování pohonných hmot, apod.
- záznamu ujetých tras automobilu přes GPS
- lokalizaci automobilu pomocí GPS
- aplikace umožňující tvorbu knihy jízd včetně generování reportů a statistik
- kombinace uvedených funkcí

Bylo vybráno několik aplikací, které jsou vhodné pro účel tvorby knihy jízd nebo záznamu o jízdě a tyto byly následně srovnány z několika úhlů. Jedná se o následující:

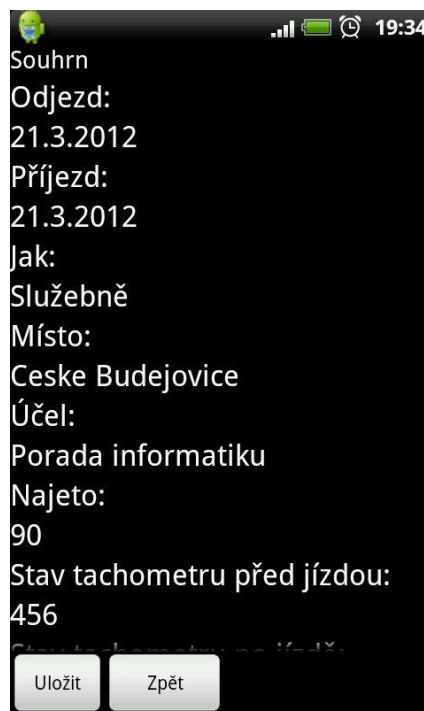
### 4.1 TMCZ Drivebook

Aplikace s názvem TMCZ Drivebook je dostupná pro mobilní platformy Windows mobile, Android, Nokia Symbian a BlackBerry. Pravděpodobně z tohoto důvodu je program napsán pro mobilní platformu J2ME a není dostupný jako nativní aplikace využívající Android API. J2ME – Java Micro Edition je edice určená k vývoji aplikací pro mobilní zařízení s omezeným výpočetním výkonem. Pro správnou funkčnost je tedy nutné instalovat J2ME Runner, až poté vlastní APK instalační balíček.

Instalace a spuštění aplikace může být o něco málo komplikovanější pro méně zkušené uživatele. Nicméně autor programu má na svých webových stránkách podrobný návod k instalaci i obsluze.

Při prvním spuštění aplikace si uživatel vytvoří profil se jménem, vlastním PINem a doplní počáteční stav tachometru. Trochu nešikovně se zde musí zadat umístění ve formátu *file:///adresář*, které slouží k uložení datového souboru se záznamem jízdy na externí paměťovou kartu.

Na dalších obrazovkách se postupně ručně vyplňuje druh jízdy (služebně/soukromě), datum odjezdu a příjezdu, místo, účel, stav tachometru po jízdě nebo celkový počet ujetých kilometrů. Z toho všeho je na konci vytvořen záznam z jízdy a uložen ve formátu souboru CSV na kartu. Záznam z jízdy je zobrazen na Obr. 6.



Obr. 6. TMCZ Drivebook

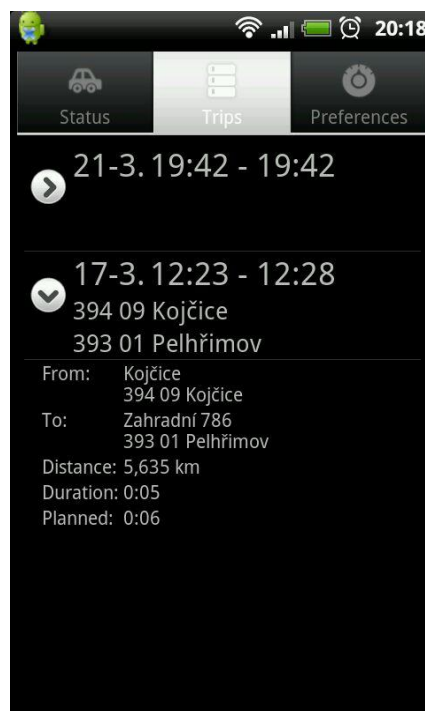
Aplikace je jako jedna z mála v českém jazyce, je přehledná a intuitivní v používání až na některé drobnosti (ruční zadání cesty k úložišti souborů). Tato aplikace však nevyhovuje našim požadavkům, protože žádným způsobem nespolečně pracuje se systémem GPS. Neumožňuje:

- záznam trasy jízdy
- automatické určení místa a času začátku a konce jízdy
- měření ujetých km, doby jízdy a dalších statistik
- určení významných průchozích bodů cesty, např. zastávky apod.

## 4.2 Auto Mileage Log

Auto Mileage Log má velice jednoduché a přehledné uživatelské rozhraní. Ovládání aplikace je řešeno formou tří záložek. Záložka *Status* obsahuje jediné tlačítko, kterým se spouští záznam trasy. Po aktivaci se na obrazovce zobrazí text s upozorněním a časem zahájení jízdy. Během doby záznamu aplikace nic nezobrazuje, pouze v tichosti signalizuje stav GPS signálu. Aktuální jízda se ukončuje stiskem stejného tlačítka, které mezi tím změnilo svůj popis na stop.

V záložce *Trips* je seznam všech uskutečněných cest. Kliknutím na jednotlivé záznamy rozbalíme detailní informace o každé jízdě. Aplikace pojmenovává místo začátku a konce jízdy, měří ujetou vzdálenost a dobu jízdy. Dále umožňuje zobrazit celou trasu v mapových podkladech Google Maps. V poslední záložce *Preferences* jsou dostupná nastavení pro výběr externího bluetooth GPS modulu, zadání e-mailové adresy na posílání reportů a určení metrických jednotek.



Obr. 7. Auto Mileage Log

Tato aplikace umožňuje zaznamenat všechny cesty automaticky. Využívá systém lokalizace zařízení pomocí GPS a služby rozhraní Google Maps. Poskytuje export do souboru CSV na externí kartu a zaslání denního reportu v těle e-mailu. E-mail v těle zprávy obsahuje velké množství zbytečného anglického informativního textu a jen na konci velice krátký záznam o cestě. Ukázka záznamu z jízdy je na Obr. 7.

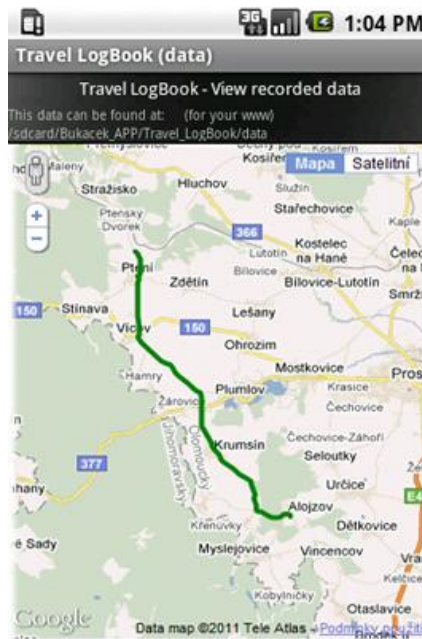
Během testování měla aplikace problém s interním GPS modulem. Nakonec se ukázalo, že aplikace si více rozumí s externími moduly, než s těmi vestavěnými přímo do zařízení. V nastavení mi chyběla možnost určení intervalu nebo vzdálenosti aktualizace GPS polohy, což ve výsledku vedlo při delších trasách k nepřesnostem z důvodu dlouhého intervalu záznamu GPS polohy.

### 4.3 Travel LogBook

Tento program po stisku tlačítka *Run LOG* začne ukládat aktuální GPS údaje v pravidelných intervalech, které lze nastavit v aktivitě *Setup*. Po ukončení je celý záznam rovnou ukládán přímo ve formátu HTML na externí paměťovou kartu v telefonu. V každém kroku měření si uživatel může vložit komentář nebo fotografii pořízenou z fotoaparátu telefonu. Zároveň se zde ukládají následující hodnoty pořízené z GPS: datum, čas, zeměpisná šířka a délka, rychlost a nadmořská výška. Naměřená data lze po uložení zobrazit na mapě kliknutím na tlačítko *Show record data* nebo je můžeme získat z externího úložiště paměti telefonu, kde jsou umístěna ve složce: */sdcard/Bukacek\_APP/Travel\_LogBook/ + název uložených dat*.

Tato složka obsahuje *data\_log.js* soubor s vlastními naměřenými daty, soubor *index.html*, jehož obsahem je HTML element DIV s mapovým podkladem. Vlastní obsluha a vykreslení trasy je realizováno pomocí JavaScriptu a rozhraní Google Maps API.

Program umí vykreslit trajektorii jízdy a zobrazuje základní informace v každé fázi měření jako bod na mapě s příslušnými GPS údaji. Tyto vlastnosti jsou však pro řešení našeho zadání nedostačující, protože aplikace neposkytuje žádné ucelené a uspořádané informace o výsledku cesty, statistické vyhodnocení, chybí možnost označení zastávky a neexistuje zde možnost zaslání reportu na e-mail. Ukázka aplikace je na Obr. 8.



Obr. 8. Travel LogBook

#### 4.4 My Car Tracks

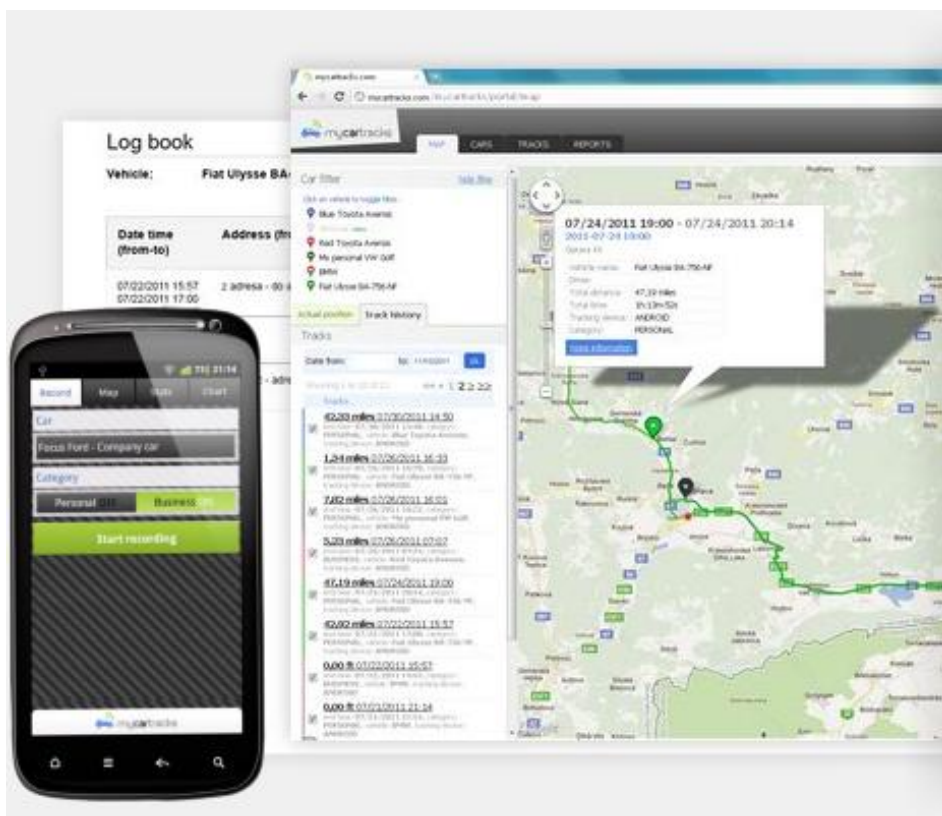
My Car Tracks je velice kvalitně zpracovaný systém pro sledování automobilů. Celý systém je rozdělen na portálovou a mobilní část. Portálová část je určena pro vstup do systému pomocí Google účtu a zobrazení záznamů jednotlivých tras. Podmínkou pro vstup je tedy vlastnictví gmailového účtu. Mobilní zařízení je svázané s portálovým přístupem právě přes tento účet, který je nutné mít zaregistrovaný i v samotném telefonu. V systému Android se vše potřebné pro synchronizaci s Google účtem nachází: hardwarové menu – nastavení – účty a synchronizace.

Mobilní část aplikace je členěna do čtyř záložek. V záložce *Record* můžeme vybrat námi přidaný automobil, případně použít defaultní, již vytvořený profil automobilu. Poté stačí zvolit v kategorii, zda se jedná o služební či soukromou jízdu a v poslední řadě spustit záznam jízdy tlačítkem *Start record*. Během jízdy může řidič resp. uživatel sledovat v záložce *Map* průběh trasy na mapě nebo měnící se statistické údaje v záložkách *Stats* a *Chart*. Po ukončení nahrávání jsou všechny jízdy dostupné přes systémové hardwarové menu telefonu, kde jsou základní informace o dané trase, a také je zde možné synchronizovat data s portálovou aplikací a exportovat trasu do řady formátů.

V portálové aplikaci je možné on-line sledovat aktuální pozici automobilu resp. telefonu s GPS na mapě. Pokud není zařízení v dosahu signálu GPS, je zobrazen pouze datum, čas a

místo poslední pozice. V záložce *Track history* je možné selektovat zaznamenané trasy podle výběru měsíce a dne z kalendáře. Pod odkazem *Graphs* v záložce *Cars* jsou prezentovány naměřené hodnoty a statistická data v podobě přehledných grafů.

My Car Tracks je určen pro malé až střední firmy, které provozují svůj vlastní autopark a sdílí několik vozidel mezi více zaměstnanci. Systém umožňuje vytvářet velice uspokojivé reporty formou tabulky generované ve formátu PDF. Ukázka z aplikace je na Obr. 9.



Obr. 9. My Car Tracks [13]

## 4.5 AndiCar

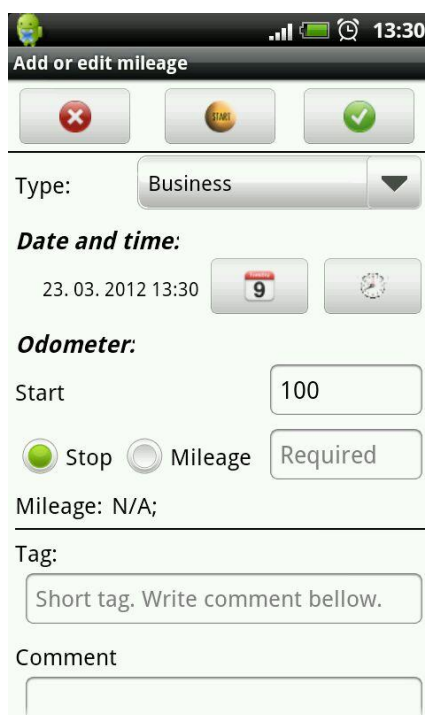
AndiCar ve verzi zdarma je open source dostupný software pod licencí GNU GPL určený pro správu vozidel. V tomto případě se ale nejedná o software z kategorie tzv. Fleet management softwaru jako u předchozího případu. Fleet management software je označení pro aplikace určené ke správě a řízení vozového parku.

Tento program má následující hlavní funkce a vlastnosti:

- možnost nastavení více vozidel, řidičů, měn a měrných jednotek
- záznam služebních / soukromých jízd, tankování pohonných hmot a jiných výdajů

- více výdajových kategorií pro přesný výpočet nákladů za ujeté kilometry
- GPS záznam jízdy s možností exportu do formátů GPX a KML
- přidání úkolů s možností nastavení upomínky na konkrétní datum a čas
- statistika
- odeslání reportu ve formátech souborů CSV, HTML jako přílohu e-mailu, upload do Google Docs nebo Dropboxu

Aplikace je dostupná i v placené verzi Pro, která navíc obsahuje například automatické zálohy databáze, šifrování databáze pro přenos na specifikovanou e-mailovou adresu, uživatelsky nastavitelné šablony nebo možnost připojení bluetooth externího GPS modulu.



The screenshot shows the 'Add or edit mileage' screen of the AndiCar application. At the top, there is a status bar with a globe icon, signal strength, battery level, and the time 13:30. Below the title bar, there are three buttons: a red 'X' for cancel, a yellow 'START' button, and a green checkmark for confirm. The form contains the following fields:

- Type:** A dropdown menu currently set to 'Business'.
- Date and time:** A date and time picker showing '23. 03. 2012 13:30' with a calendar icon and a refresh icon.
- Odometer:** A section with 'Start' set to '100'. Below it are radio buttons for 'Stop' (selected) and 'Mileage', and a 'Required' button.
- Mileage:** A text field containing 'N/A'.
- Tag:** A text input field with the placeholder text 'Short tag. Write comment bellow.'.
- Comment:** A larger text input field for additional notes.

Obr. 10. AndiCar

Pro náš účel byly testovány hlavně funkce určené k záznamu jízdy a generování reportů z jednotlivých cest. V aplikaci existují tři způsoby, pomocí kterých lze docílit záznamu trasy a ujetých kilometrů. Obrazovka pro přidání a editaci jízdy je zobrazena na Obr. 10.

1. Jednoduchý mód – v jednom kroku: Použijeme v případě, když chceme vytvořit záznam až po uskutečněné jízdě. Na hlavní obrazovce klikneme na tlačítko *Add*, které je umístěné v zóně *Last mileage*. Na následující obrazovce nazvané *Add or edit mileage* je položka *Odometer-Start* automaticky doplněna poslední známou

hodnotou tachometru, takže zbývá ručně zadat hodnotu tachometru po jízdě nebo počet ujetých kilometrů, případný komentář a záznam uložit.

2. Mód s nahráváním – ve dvou krocích: Na hlavní obrazovce v *Last mileage* – přidáme novou jízdu tlačítkem *Add*. Dále je potřeba vyplnit položku *Odometer-Start* před zahájením jízdy, poté klikneme na tlačítko *Start* pro zahájení jízdy. Na konci jízdy se vrátíme zpět k této obrazovce, klikneme na tlačítko *Stop*, doplníme konečnou hodnotu tachometru a záznam uložíme.
3. Poslední mód je čistě automatický a podpořen GPS senzorem. Po kliknutí na tlačítko *Add* v zóně *Last GPS track* se objeví obrazovka, na které se tlačítkem *Record* zahájí vlastní záznam jízdy pomocí GPS signálu. Po zastavení záznamu je uživatel přesměrován opět na obrazovku *Add or edit mileage*, kde jsou už tentokrát údaje o stavu tachometru před a po jízdě doplněny samotnou aplikací.

Všechny jízdy je možné zobrazit na mapových podkladech Google Maps nebo si nechat odeslat report vybraných cest jako přílohu e-mailu a to ve formátech CSV nebo HTML.

AndiCar by téměř splňoval naše požadavky, bohužel, ale neumí pojmenovat destinaci začátku a konce jízdy, neřeší možnost zastávek, jeho uživatelské rozhraní působí dost nepřehledně a není vůbec intuitivní. Pro méně zkušené uživatele téměř nepoužitelný bez podrobného prostudování nápovědy.

## 4.6 Závěr analýzy

Po provedení naší analýzy jednotlivých aplikací zabývajících se problematikou záznamu jízdy, tvorby knihy jízd či reportů z cest převážně podpořené systémem GPS, které jsou zároveň volně dostupné na internetu pro systém OS Android, zjišťujeme, že prakticky není aplikace pro českého uživatele, jež by neobsahovala nějaké nedostatky, či by plně vyhovovala našim požadavkům.

Naším cílem je však vyvinout novou aplikaci pro systém Android v českém jazyce s intuitivním ovládáním, která bude určena především pro drobné podnikatele a zaměstnance, podnikající služební i soukromé cesty v rámci svých pracovních aktivit. Aplikace by měla inteligentně řešit celý proces od zahájení jízdy, přes identifikaci a pojmenování významných průchozích bodů cesty, až po její vyhodnocení a podání výsledků ve formě přehledného reportu určeného, například pro nadřízeného nebo pro usnadnění vyhotovení knihy jízd pro daňové účely. Nebude se jednat o komplexní aplikaci na řízení autoprovozu podobně jako je

system My Car Tracks a jemu podobný, protože tyto softwary jsou pro náš účel zbytečně rozsáhlé.

## 5 UŽIVATELSKÁ DOKUMENTACE

### 5.1 Požadavky a instalace aplikace

Aplikace GPS Mobile Workers je určena pro mobilní zařízení s operačním systémem Android, která jsou vybavena vestavěným GPS čipem. Aplikace ke své funkci používá mapové podklady společnosti Google a některé další služby rozhraní Google Maps API jako například funkcionalitu reverzního geokódování, o které se toho více dozvíte v kapitole 6.6.2. Z tohoto důvodu je nutné mít v telefonu pro správný běh aplikace minimálně verzi systému Android s podporou Google Maps. OS Android podporuje mapy od verze 1.6 nebo vyšší.

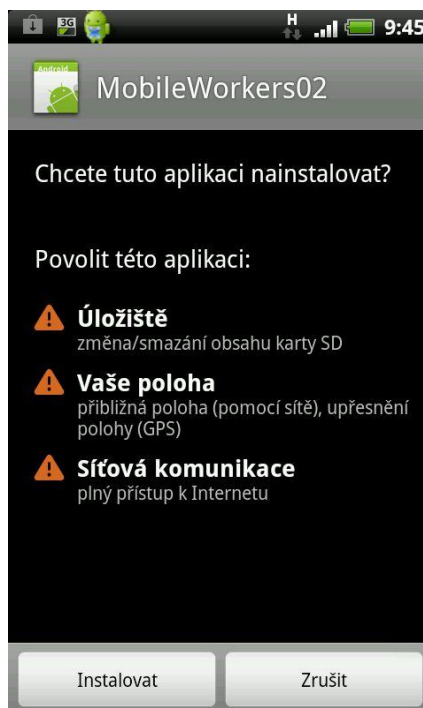
Pro plynulý běh jsou však doporučovány verze Android 2.3.3. nebo 2.3.5. Dále je nutné mít v telefonu povolenou možnost instalace aplikací třetích stran pomocí instalačního balíčku APK z paměťové karty. Potřebné nastavení je k dispozici přes hardwarové menu – Nastavení – Aplikace – zde musí být aktivní zaškrtačací políčko *Neznámé zdroje*, které povoluje instalaci mimo službu Market. Tento požadavek bude odstraněn, jakmile se podaří umístit aplikaci na oficiální Android Market portál (Google Play).

Dalším nezbytným požadavkem ke správné funkčnosti aplikace je dostupnost internetové konektivity. Uživatel musí mít povolen datový přenos od svého mobilního operátora a správně nakonfigurovaný telefon. K fungování postačuje přenosová rychlost na úrovni technologie GPRS/EDGE, nejlepší a nejrychlejší variantou je však dostupnost 3G sítě. Přístup k internetu je důležitý pro rychlejší lokalizaci zařízení díky technologii A-GPS a ke komunikaci se službou Google API.

#### Postup instalace:

1. Připojíme telefon pomocí USB kabelu k počítači, po připojení se na obrazovce telefonu objeví výzva: Vyberte typ připojení, kde se zvolí možnost Disková jednotka.
2. Poté si v počítači otevřeme tuto diskovou jednotku a do kořene zkopírujeme z CD instalační soubor *MobileWorkers.apk* a následně bezpečně odpojíme telefon. Vyčkáme, až bude SD karta opět připravena k použití.
3. V posledním kroku spustíme instalační balíček z telefonu pomocí nějakého správce souborů např. programem *ES Správce Souborů*.

Před zahájením samotné instalace je uživateli nabídnut dialog, kde je informován o tom, která oprávnění aplikace vyžaduje pro svou činnost. Tento dialog musí být uživatelem odsouhlasen, až poté se spustí vlastní proces instalace viz. Obr. 11.



Obr. 11. Souhlas k instalaci

Po nainstalování se přidá ikona s názvem Mobile Workers do hlavní nabídky. Spuštění aplikace probíhá výhradně přes tuto ikonu.

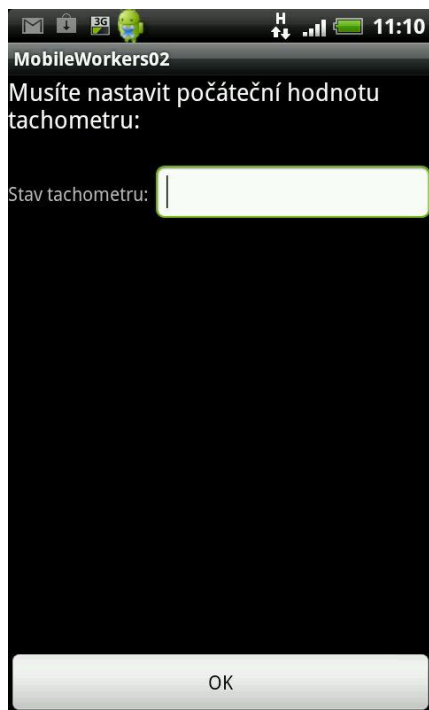
## 5.2 První spuštění aplikace

První spuštění aplikace, hned po instalaci se odlišuje od ostatních tím, že je po uživateli požadováno nastavení aktuální skutečné hodnoty tachometru vozidla Obr. 12. Po tomto počátečním nastavení je hodnota sama aktualizována po každé jízdě o naměřený počet ujetých kilometrů.

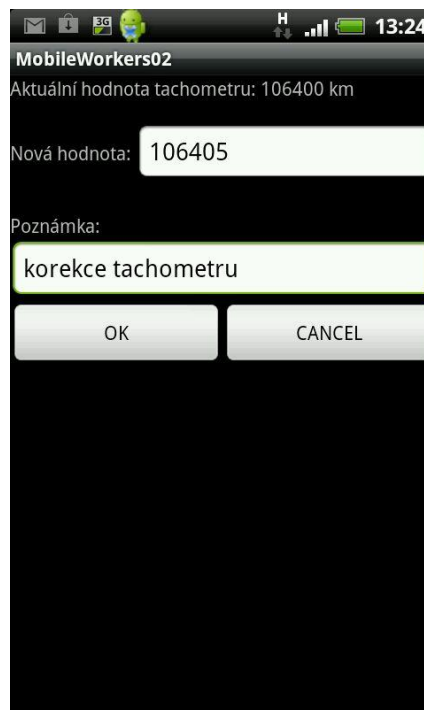
Měření probíhá automaticky pomocí systému GPS. Možnosti ohledně nastavení intervalu mezi aktualizacemi GPS polohy, které ovlivňují přesnost výpočtu lze upravit v nastavení aplikace, o tom více v kapitole 5.5. Podle dosavadních zkušeností s aplikací probíhalo měření velice věrohodně, po každé jízdě odpovídala hodnota tachometru v aplikaci hodnotě ve vozidle.

Pokud dojde k tomu, že se bude skutečná hodnota tachometru vozidla rozcházet s hodnotou, kterou udává aplikace Mobile Workers, je možné udělat korekci tachometru na skutečnou

hodnotu. Toto je možné provést v aktivitě *Tachometr* - Obr. 13: hardwarové menu – Více – Tachometr. Zde uživatel zadá novou hodnotu a do poznámky vyplní důvod změny a uloží tlačítkem OK.



Obr. 12. Nastavení tachometru



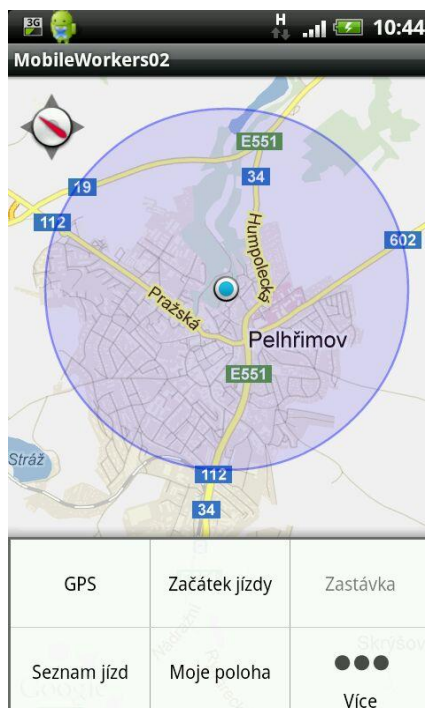
Obr. 13. Korekce tachometru

### 5.3 Hlavní obrazovka

Po spuštění aplikace se zobrazí hlavní obrazovka. Zde je uživateli zobrazena mapa s jeho aktuální zeměpisnou polohou. Poloha je označena malým kulatým bodem, okolo něhož je modře zbarvena oblast o poloměru, jehož hodnota udává velikost kruhu. Plocha kruhu graficky vyjadřuje aktuální přesnost získanou k určení momentální zeměpisné polohy zařízení.

Aplikace se hned po spuštění snaží najít nejvhodnějšího poskytovatele lokalizační služby pro zjištění co nejpřesnější polohy. Jestliže v tu chvíli není aktivována funkce použití GPS satelitů nebo není tento signál zrovna k dispozici, potom je určena alespoň přibližná poloha, a to jedním z alternativních způsobů:

- Triangulací na základě polohy vysílacích věží mobilní sítě, při které se poloha určuje na základě síly signálu přijímaného nejbližšími BTS stanicemi.
- Lokalizace na základě veřejných Wi-Fi sítí, které mají známou zeměpisnou polohu.



Obr. 14. Hlavní obrazovka

Ovládání celé aplikace je řešeno hlavně pomocí hardwarových tlačítek menu a šipky zpět, případně je umožněno na některých obrazovkách vyvolat delším přidržením nad některou z položek seznamu kontextové menu. Vše potřebné k ovládání aplikace je situováno do menu hlavní aktivity. To je přístupné přes hardwarové tlačítko menu telefonu a prakticky se pomocí něho dostaneme do všech částí aplikace. Hlavní obrazovka aplikace je na Obr. 14.

Na hlavní obrazovce je dále v levém horním rohu zobrazen kompas. V případě, že došlo ke spuštění záznamu, je navíc zobrazen symbol určující začátek resp. konec cesty. Během záznamu mohou být přidávány jednotlivé zastávky, které jsou taktéž označeny na mapě příslušným symbolem.

Mapu lze pouze přibližovat, oddalovat, posouvat, není možné vyhledávání adres a podobně. Tlačítka pro přiblížení resp. oddálení se zobrazí v okamžiku, kdy se uživatel dotkne displeje. Jsou umístěná ve spodní části obrazovky.

Vysvětlení jednotlivých symbolů zobrazených na mapě:



Začátek jízdy




Značka symbolizuje zastávku



Konec jízdy

## 5.4 Menu

Po tisknutí hardwarového tlačítka menu na zařízení, pokud se uživatel nachází na hlavní obrazovce s mapovými podklady Google Maps, se vyvolá nabídka, ve které se nachází seznam šesti obrazovek, do nichž se uživatel může přepnout. Poslední nabídka pojmenovaná *Více* rozšiřuje nabídku o další sub-menu. Pro návrat ze zvolené obrazovky je doporučeno používat hardwarové tlačítko zařízení zpět. Tlačítka menu a zpět jsou vybavena všechna zařízení s operačním systémem Android.

GPS	Začátek jízdy	Zastávka
Seznam jízd	Moje poloha	 Více

Obr. 15. Menu aplikace

Pro bližší seznámení s funkcemi menu je níže uveden krátký přehled:

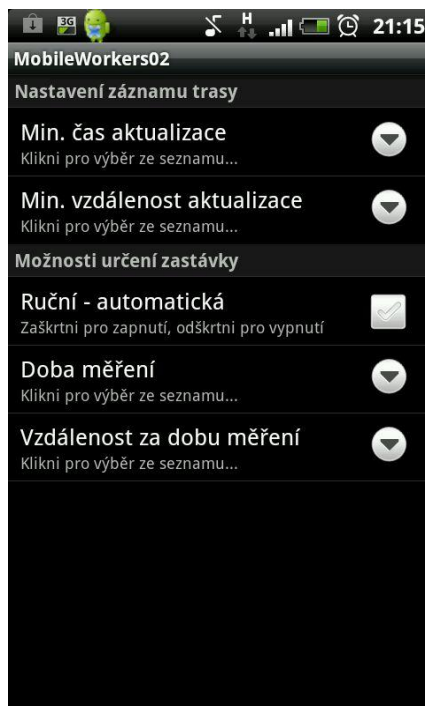
- GPS:** Slouží k vyvolání systémové aktivity (obrazovky) *Umístění* pro zapnutí/vypnutí GPS a zaměření přes bezdrátové sítě. Aktivaci je nutné provést před zahájením záznamu jízdy. Pokud není aktivní GPS přijímač a uživatel se pokusí o zapnutí záznamu, objeví se hlášení o upozornění na vypnutý GPS, a uživatel je následně přesměrován do výše uvedeného nastavení, kde může použití GPS zapnout. Zároveň je doporučeno mít v tuto chvíli povolen přístup k internetu z důvodu rychlejší fixace GPS přijímače k satelitům, díky využití technologie A-GPS. Fix GPS signalizuje ikonka satelitu v oznamovací oblasti hned vedle symbolů baterie a hodin.
- Začátek - konec jízdy:** Začátek i konec záznamu jízdy je iniciován výhradně výběrem této položky menu. Celý proces začíná vyplněním jednoduchého formuláře,

kteřý je zobrazen na Obr. 19. Formulář obsahuje pole pro vyplnění názvu, rozevírací seznam s možností výběru služební či soukromé cesty a více řádkové pole pro případnou poznámku. Po potvrzení formuláře a tím zahájení jízdy se změní popisek tlačítka na *Konec jízdy*, zároveň s tímto se na mapě v místě aktuálně získané GPS zeměpisné pozice zobrazí značka, která symbolizuje místo začátku resp. konce jízdy. Dále je zaznamenán datum, čas, adresa a počáteční resp. konečný stav tachometru.

- **Zastávka:** Jestliže již byla zahájena jízda tlačítkem *Začátek jízdy*, povolí se přístup k položce *Zastávka*, a to pouze pokud je zároveň v nastavení aplikace deaktivována možnost automatické detekce zastávky, o této volbě více v kapitole 5.5 pojednávající o možnostech nastavení aplikace. Stiskem tohoto tlačítka je zobrazen dialog pro přidání zastávky jako na Obr. 17. V dialogu se vyplňuje krátké textové pole udávající účel a druh zastávky (soukromá/služební). Zároveň je zobrazen symbol v místě zastávky a je zaznamenán datum, čas a adresa.
- **Seznam jízd:** Slouží k vyvolání obrazovky, ve které je selektován seznam všech uskutečněných jízd. Každou cestu zde reprezentuje jeden řádek se základními informacemi o jízdě viz. Obr. 21. Řádek se skládá z názvu, data, času odjezdu a příjezdu, celkové doby jízdy a ujetých kilometrů. Po kliknutí na řádek je zobrazen detailní přehled o jízdě spolu se statistickými údaji jako na Obr. 22. Další možnosti lze vyvolat z kontextového menu delším přidržením (alespoň 2 sekundy) nad konkrétním řádkem seznamu. Ukázka kontextové menu je na Obr. 18.
- **Moje poloha:** Funkce *Moje poloha* vrací aktuální GPS zeměpisnou polohu zařízení vycentrovanou na střed mapy a zobrazuje krátkou informační bublinu s adresou aktuálního místa.
- **Více:** Pod touto položkou se skrývá sub-menu aplikace obsahující dvě další položky. Položka *Nastavení* je blíže popsána v kapitole 5.5. Druhá položka *Tachometr* poskytuje možnost upravit hodnotu tachometru, pokud jeho hodnota neodpovídá skutečné hodnotě tachometru v automobilu.

## 5.5 Nastavení

Na obrazovce *Nastavení* uživatel nalezne veškeré možnosti týkající se nastavení parametrů aplikace. Obrazovka je rozdělena do dvou sekcí jak je vidět na Obr. 16.



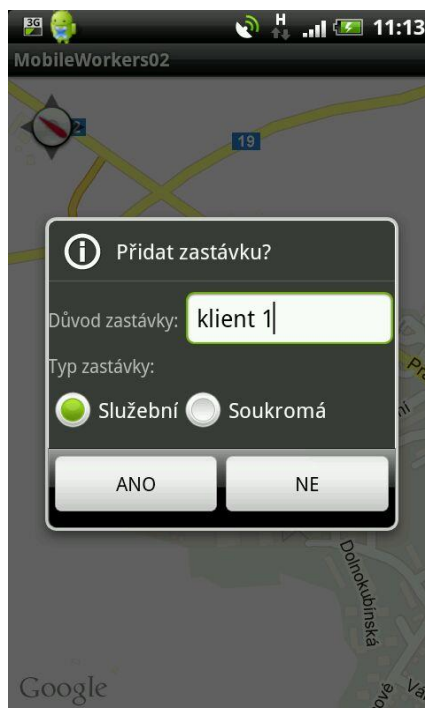
Obr. 16. Nastavení aplikace

V první části nazvané *Nastavení záznamu trasy* se nachází především možnost nastavit intervaly pro aktualizaci GPS pozice. Lze upravit minimální vzdálenost nebo minimální čas mezi aktualizací GPS pozice. Pokud uživatel nastaví u obou nejnižší čas a nejmenší vzdálenost, potom docílí co možná největší rychlosti, za kterou je zařízení schopné dodat aktualizace nových souřadnic. To velice zpřesňuje měření, ale zároveň více zatěžuje hardware telefonu, a tím rychleji vybíjí baterii.

Druhá část pojmenovaná *Možnosti určení zastávky* nastavuje chování aplikace při zahájení zastávky na cestě. Pokud je zaškrťovací pole *Ruční – automatická* aktivní (zaškrtnuto), tak se aplikace sama snaží automaticky identifikovat řidičův úmysl o blížící se zastávce, a sama mu nabídne dialogové okno pro přidání zastávky. Další dvě nastavení *Doba měření* a *Vzdálenost za dobu měření* toto chování blíže upřesňuje.

Příkladem může být řidič, který se blíží ke svému cíli zastávky. Takový řidič pohybující se v cílové oblasti většinou ujede velmi malý počet metrů za krátkou dobu. Právě tuto situaci aplikace vyhodnotí na základě upřesňujících parametrů následujícím způsobem: Pokud automobil neurazí větší než nastavenou vzdálenost za více než nastavenou dobu, je automaticky generováno dialogové okno pro přidání zastávky s tím, že situace typu čekání v koloně nebo na semaforech jsou patřičným způsobem ošetřeny. Dialog může být uživatelem akceptován nebo odmítnut.

V případě, že je zaškrťovací pole *Ruční – automatická* neaktivní (odškrtnuto), potom si přidání zastávky určuje řidič ručně výběrem položky *Zastávka* z hlavního menu aplikace.



Obr. 17. Dialog pro přidání zastávky

## 5.6 Scénář - vytvoření záznamu jízdy

V následujícím textu je krátce popsán scénář, jak použít aplikaci pro záznam jízdy od zahájení až po vytvoření následného reportu z cesty a jeho odeslání na e-mailovou adresu.

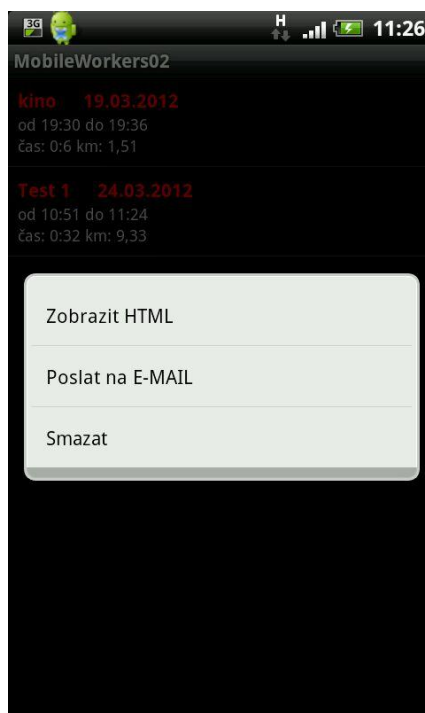
V první řadě spustíme aplikaci. Nyní se zobrazí hlavní obrazovka s mapovými podklady obsahující ovládací prvky a hlavní menu aplikace. V tuto chvíli vybereme a stiskneme v menu tlačítko GPS a na další obrazovce nastavíme příjem GPS signálu. K návratu na hlavní obrazovku je vhodné vždy používat hardwarové tlačítko zpět. Pokud tak neučiníme a rovnou se pokusíme zahájit jízdu tlačítkem *Začátek jízdy*, budeme upozorněni a přesměrováni k výše zmíněnému nastavení. Dále si můžeme nastavit nám vyhovující parametry aplikace dostupné v nastavení: menu – Více – Nastavení. Jinak budou použity výchozí hodnoty.

Nyní musíme vyčkat na první fixaci GPS přijímače k satelitům. Tato fáze trvá na každém zařízení různou dobu. Záleží hlavně na kvalitě GPS přijímače, dostupnosti A-GPS technologie, dobrém výhledu na oblohu a ostatních okolnostech jako například na tom, kde bylo zařízení naposledy lokalizováno apod. Další případný fix trvá znatelně kratší dobu. Vše je signalizováno blikajícím symbolem satelitu v oznamovací oblasti telefonu. Pokud se satelit

ustálil, přestal blikat, je vše v pořádku a můžeme zahájit jízdu stiskem tlačítka *Začátek jízdy*. Vyplníme formulář, který je zobrazen na Obr. 19, a vyrazíme na cestu. Během jízdy můžeme sledovat pohyb vozidla na mapě.

Pokud chceme v průběhu cesty zastavit, stačí stisknout tlačítko *Zastávka* a vyplnit jednoduchý dialog pro určení zastávky. Jestliže máme v nastavení aktivovanou možnost určení automatických zastávek, aplikace se nám tento dialog pokusí nabídnout zcela automaticky.

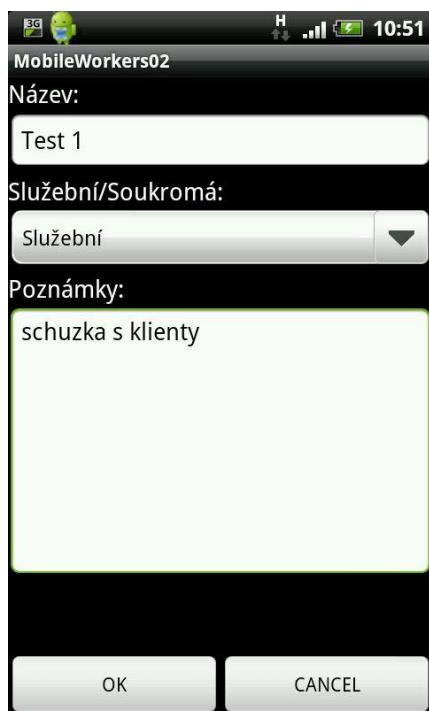
Pro dokončení záznamu jízdy stiskneme tlačítko *Konec jízdy*. Pokud nyní chceme prohlédnout seznam uskutečněných jízd, zvolíme v menu možnost *Seznam jízd*. Na každý řádek můžeme kliknout a prohlížet detailní přehled o jízdě nebo delším přidržením vyvolat kontextové menu a vybrat jednu z nabízených možností.



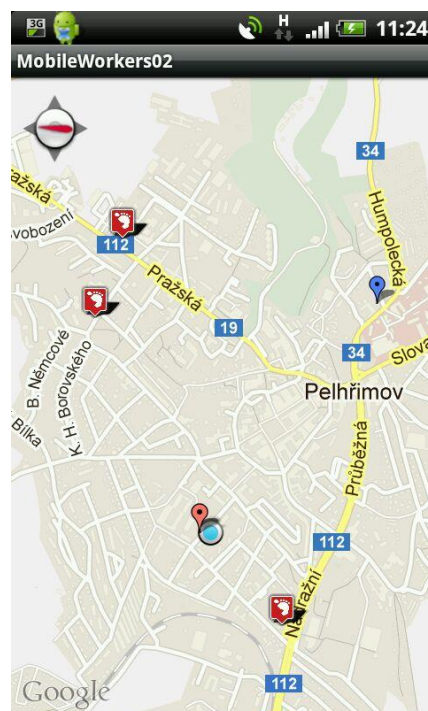
Obr. 18. Kontextové menu

V kontextovém menu nalezneme:

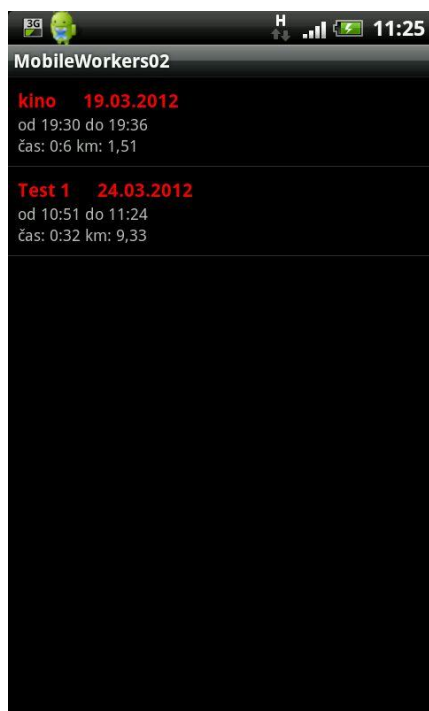
- Položku *Zobrazit HTML report*. Tím docílíme zobrazení souhrnného reportu z cesty formou přehledných tabulek ve webovém prohlížeči mobilního telefonu. Každý telefon s operačním systémem Android obsahuje internetový prohlížeč založený na jádře WebKit.
- Položka *Poslat na E-MAIL* umožňuje tento report odeslat v příloze e-mailové zprávy. Po kliknutí nám Android nabídne seznam aplikací, pomocí kterých lze odeslat e-mail. Na nás je vybrat si ten pravý. V době testování aplikace byl použit mobilní e-mailový klient Gmail, s kterým vše fungovalo bez nejmenších problémů.
- Poslední položkou smažeme označený záznam jízdy z databáze.



Obr. 19. Zahájení jízdy



Obr. 20. Mapa se symboly



Obr. 21. Seznam jízd



Obr. 22. Detail jízdy

## 6 IMPLEMENTAČNÍ DOKUMENTACE

V následujících kapitolách a podkapitolách bude blíže vysvětlena struktura programu. Bude zde uveden popis použitých technologií, jednotlivých tříd, schéma databáze, postupu integrace Google Maps, problémů a jejich řešení. Během studia problematiky operačního systému Android a programovacího jazyka Java bylo čerpáno z těchto zdrojů: [6], [7], [9], [14], [15] a [16]

### 6.1 Použité technologie

Jako programovací jazyk byl zvolen jazyk Java. Jedná se o primární jazyk určený k vývoji aplikací pro systému Android. Při psaní programů se používá syntaxe tohoto jazyka a knihovna tříd, která připomíná podmnožinu edice Java SE a navíc obsahuje také rozšíření specifická pro systém Android. Jde hlavně o doplnění spoustou API „šitých“ na míru přímo pro tento systém. Jako příklad můžeme uvést API pro práci s grafikou, SQLite databázi, lokalizační službou, Apache knihovnou a mnoho dalších. Přehled všech balíčků je k nahlédnutí v referenční dokumentaci. [14]

Běhovým prostředím a mozkem celého Android frameworku je Dalvik VM. Dalvik je Virtual Machine optimalizovaný pro běh na mobilním zařízení. Instrukce jeho bytekódu jsou mnohem úspornější než instrukce skutečné Javy.

Oficiálním vývojovým prostředím pro Android, které je podporováno ze strany Googlu, je Eclipse. Jedná se o open source vývojovou platformu, která obsahuje základní části pro vývoj v Javě, jako je třeba kompilátor nebo debugger. Eclipse je možné velice snadno rozšiřovat pomocí pluginů, a tím zajistit podporu vývoje pro téměř jakýkoli programovací jazyk nebo prostředí.

Dále byla použita nejnovější verze vývojového balíčku Android Software Development Kit (SDK), který v sobě obsahuje emulátor, potřebné knihovny a debugger. Velkou výhodou byla možnost využití pokročilého zásuvného pluginu Android Developer Tools (ADT) pro prostředí Eclipse.

V aplikaci je použita relační databáze SQLite. Android má nativní podporu těchto databází. SQLite je velmi populární zabudovaná databáze, protože obsahuje čisté SQL rozhraní, má velmi malé paměťové nároky a je poměrně rychlá.

Pro zobrazení průběhu jízdy a symbolů byly zvoleny mapové podklady Google Maps. Google poskytuje API knihovny přímo pro Android, díky tomu se integrace map do prostředí telefonu přímo nabízí. Více o tomto řešení v kapitole 6.6.1.

## 6.2 Architektura aplikace Mobile Workers

Zdrojový kód aplikace je logicky rozdělen do tří balíčků, které seskupují související třídy. Jednotlivé balíčky obsahují soubory s příponou *.java*, jejichž jména zároveň odpovídají jménu definované třídy. Název balíčku odpovídá adresáři, kde je daná třída umístěna.

Struktura balíčků je následující:

- **gm.tichy.interfaces**: Obsahuje rozhraní *GPSCallback* s jednou abstraktní metodou *onGPSUpdate()*, která je implementována třídou *Map*.
- **gm.tichy.managers**: Obsahuje třídy *GPSManager*, *MyDatabaseManager* a *MyDbHelper*.
- **gm.tichy.mobileworkers**: Obsahuje zbytek tříd, které převážně reprezentují uživatelské rozhraní a aplikační logiku pomocí aktivit.

Dále bude ve stručnosti demonstrován princip fungování aplikace. Konkrétní popis realizací jednotlivých tříd bude detailněji upřesněn později v dalších kapitolách.

Po spuštění aplikace se zobrazí hlavní okno, které reprezentuje třída *Map*. Tato třída rozšiřuje třídu *MapActivity* a implementuje rozhraní *GPSCallback*. Při prvním spuštění aktivity je volána metoda *onCreate()*, v které jsou vytvořeny instance tříd (objekty) *GPSManager*, *MyDatabaseManager* a jsou zde provedeny další nezbytné inicializace.

*MyDatabaseManager* realizuje databázovou vrstvu pro komunikaci s SQLite databází. Hned v konstruktoru zajišťuje vytvoření potřebných databázových tabulek.

*GPSManager* implementuje potřebnou logiku pro správu, řízení a udržení lokalizační systémové služby k určování a aktualizaci zeměpisné polohy zařízení. Třída se stará hlavně o vytvoření objektu *LocationListener*, získání systémové služby *LOCATION\_SERVICE*, zahájení a ukončení naslouchání změn souřadnic a zaregistrování callback metody *onGPSUpdate()*, do které se po každé změně polohy předávají nové souřadnice spolu s dalšími údaji o rychlosti, nadmořské výšce, času, apod. zabalenými v objektu *Location*. Callback metodu *onGPSUpdate()* implementuje právě třída *Map*, která se tímto způsobem

dozví o každé změně souřadnic. Díky tomuto mechanismu třída *Map* při své činnosti využívá aktuální informace o zeměpisné poloze z objektu *Location*.

Třída *Measure* implementuje stěžejní algoritmy, které zajišťují měření ujeté vzdálenosti pomocí GPS, výpočet statistiky (maximální a průměrné rychlosti) a další podpůrné výpočty. Instance této třídy je vytvořena hned po zahájení jízdy, stisknutím tlačítka *Začátek jízdy*. Tím vlastně dojde k vytvoření a inicializaci objektu *measurer*, který pravidelně aktualizuje naměřenou vzdálenost a další proměnné. Při každé změně souřadnic dochází zároveň k ukládání dat do databáze.

Vykreslení symbolů na mapě zajišťuje vnitřní třída *SitesOverlay*, která rozšiřuje třídu *ItemizedOverlay*. Třída umožňuje vytvoření mapové vrstvy, kterou v podstatě tvoří datová struktura typu *List* (seznam) a její připojení k mapě. Vrstva obsahuje výčet položek, objekty datového typu *OverlayItem*, jenž reprezentují zobrazený symbol.

K ukládání uživatelsky nastavených detailů konfigurace v možnostech nastavení aplikace je využito systému preferencí. Android umožňuje aktivitám ukládat uživatelské preference ve formě párů klíč/hodnota. Načtení preferencí probíhá ve třídě *Map* v metodě *onResume()*, což zajišťuje okamžitou aktualizaci nastavení po každé změně.

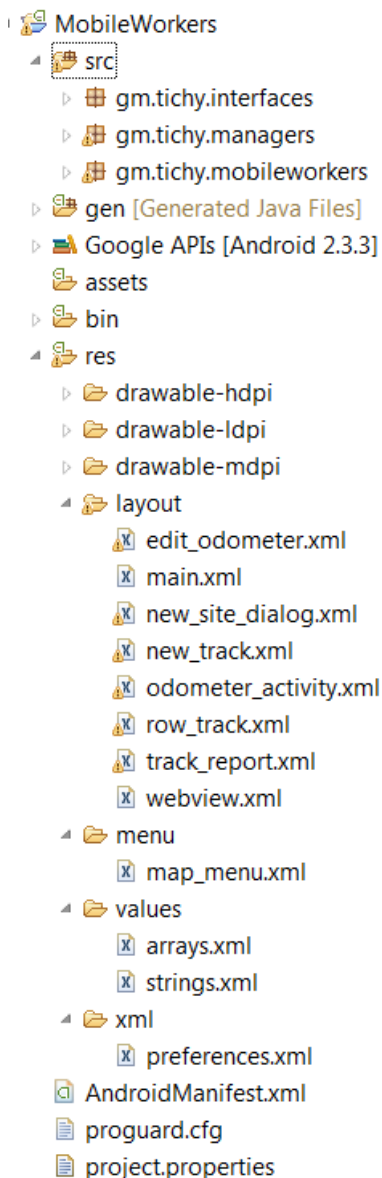
Komunikace mezi aktivitami a jejich vzájemné propojení je řešeno pomocí zasílání *Intentů*. *Intent* v sobě může obsahovat další dodatečná data pro volanou aktivitu a také od ní může vyžadovat odpověď, která musí být dále patřičným způsobem ošetřena v metodě *onActivityResult()*. Příklad vytvoření a odeslání *intentu* je vidět na Obr. 23.

```
113
114     Intent i = new Intent(Tracks.this, TrackReport.class);
115
116     i.putExtra("trackName", trackName);
117     i.putExtra("trackTitle", trackTitle);
118     i.putExtra("trackType", trackType);
119
120     String trackDateTime = trackStartDate + ", " + trackStartTime + " - " + trackEndTime;
121
122     i.putExtra("trackDateTime", trackDateTime);
123     i.putExtra("trackDistance", trackDistance);
124     i.putExtra("trackTotalTime", trackTotalTime);
125     i.putExtra("trackMaxSpeed", trackMaxSpeed);
126     i.putExtra("trackAvgSpeed", trackAvgSpeed);
127     i.putExtra("trackStartAddress", trackStartAddress);
128     i.putExtra("trackEndAddress", trackEndAddress);
129
130     startActivity(i);
131
```

Obr. 23. Ukázka vytvoření a odeslání záměru (*intentu*)

### 6.3 Adresářová struktura projektu

Pro zvýšení přehlednosti a zachování dané struktury Android projektů jsou jednotlivé soubory logicky rozmístěny do adresářů. Adresářová struktura aplikace Mobile Workers je zobrazena na Obr. 24.



Obr. 24. Adresářová struktura projektu

- src/: Složka uchovávající zdrojový kód aplikace v jazyce Java.
- gen/: Složka, do které překladačové nástroje systému Android umístí zdrojový kód, který vygenerují.
- Google APIs: třídy pro práci s Google API.
- assets/: Složka obsahující ostatní statické soubory, které mohou být přibaleny k aplikaci pro použití na zařízení.
- bin/: Uchovává aplikaci po jejím zkompileování.
- res/: Složka uchovávající prostředky – ikony, návrhy grafického uživatelského rozhraní (GUI), zdroje: textové řetězce a pole použité v aplikaci, jiné XML struktury apod.
- AndroidManifest.xml: Důležitý XML soubor popisující vyvíjenou aplikaci a komponenty (aktivity, služby atd.), které aplikace obsahuje.

V níže uvedených kapitolách budou popsány jednotlivé podadresáře prostředků (resources) použité při vývoji aplikace. Prostředky se ukládají jako soubory umístěné v adresáři res/

projektu. Systém Android používá pro definici uživatelského rozhraní (GUI) návrhy založené na XML. Dalšími druhy prostředků jsou například: [6]

- Obrázky (res/drawable): pro ukládání statických ikon a obrázků do uživatelského rozhraní. Zde jsou například uloženy symboly, které se zobrazují v mapových vrstvách Google Maps aplikace Mobile Workers.
- Surové prostředky (res/raw): libovolné soubory, které mají nějaký význam pro aplikaci.
- Řetězce, barvy, pole (res/values): pro přidělení názvů těmto druhům konstant a pro jejich oddělení od zbytku zdrojového kódu. Jsou zde například názvy databázových tabulek, názvy textových popisků, ale i celé *SQL CREATE TABLE* příkazy na vytvoření struktury databáze pro aplikaci Mobile Workers.
- XML (res/xml): pro statické XML soubory obsahující vlastní data a struktury aplikace. Aplikace Mobile Workers zde například ukládá celou strukturu použitou v preferences.

### 6.3.1 Layout

Jak již bylo zmíněno výše, Android používá k definici uživatelského rozhraní (GUI) návrhy XML. Návrh založený na XML je specifikací vzájemných vztahů widgetů každé GUI aktivity a jejich vztahů ke kontejnerům zakódovaných v XML formátu. Každý XML soubor obsahuje strom elementů specifikujících rozložení widgetů a kontejnerů, které tvoří hierarchii náhledů.

[6] Část XML layoutu zobrazuje Obr. 25.

```
1 <?xml version="1.0" encoding="utf-8"?>
2=<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:orientation="vertical" >
6     <TextView
7         android:id="@+id/textView1"
8         android:layout_width="wrap_content"
9         android:layout_height="wrap_content"
10        android:text="@string/nazev_jizdy"
11        android:textAppearance="?android:attr/textAppearanceMedium" />
12=
13     <EditText
14         android:id="@+id/edit_name"
15         android:inputType="text"
16         android:layout_width="match_parent"
17         android:layout_height="wrap_content" >
18         <requestFocus />
19     </EditText>
```

Obr. 25. Ukázka XML návrhu obrazovky

K sestavení uživatelského rozhraní z definovaného XML návrhu stačí použít jediný příkaz v callback metodě *onCreate()* dané aktivity:

```
setContentView(R.layout.new_track);
```

K dalšímu použití jednotlivých *View* (EditText, Button, atd.) z návrhu je nutné získat jejich instance pomocí metody:

```
findViewById();
```

### 6.3.2 Menu

Menu aplikace je přístupné v hlavní aktivitě třídy *Map* stiskem hardwarového tlačítka menu. Menu je vytvořené v okamžiku volání metody *onCreateOptionsMenu()*:

Celá specifikace jeho struktury je umístěna v souboru *res/menu/map\_menu.xml* a jeho naplnění zajišťuje třída *MenuInflater*. Tato třída pomocí metody *inflate()* konvertuje XML specifikaci návrhu na skutečné objekty, které návrh menu reprezentuje a zajistí volání potřebných metod k jeho vytvoření.

### 6.3.3 Preferences

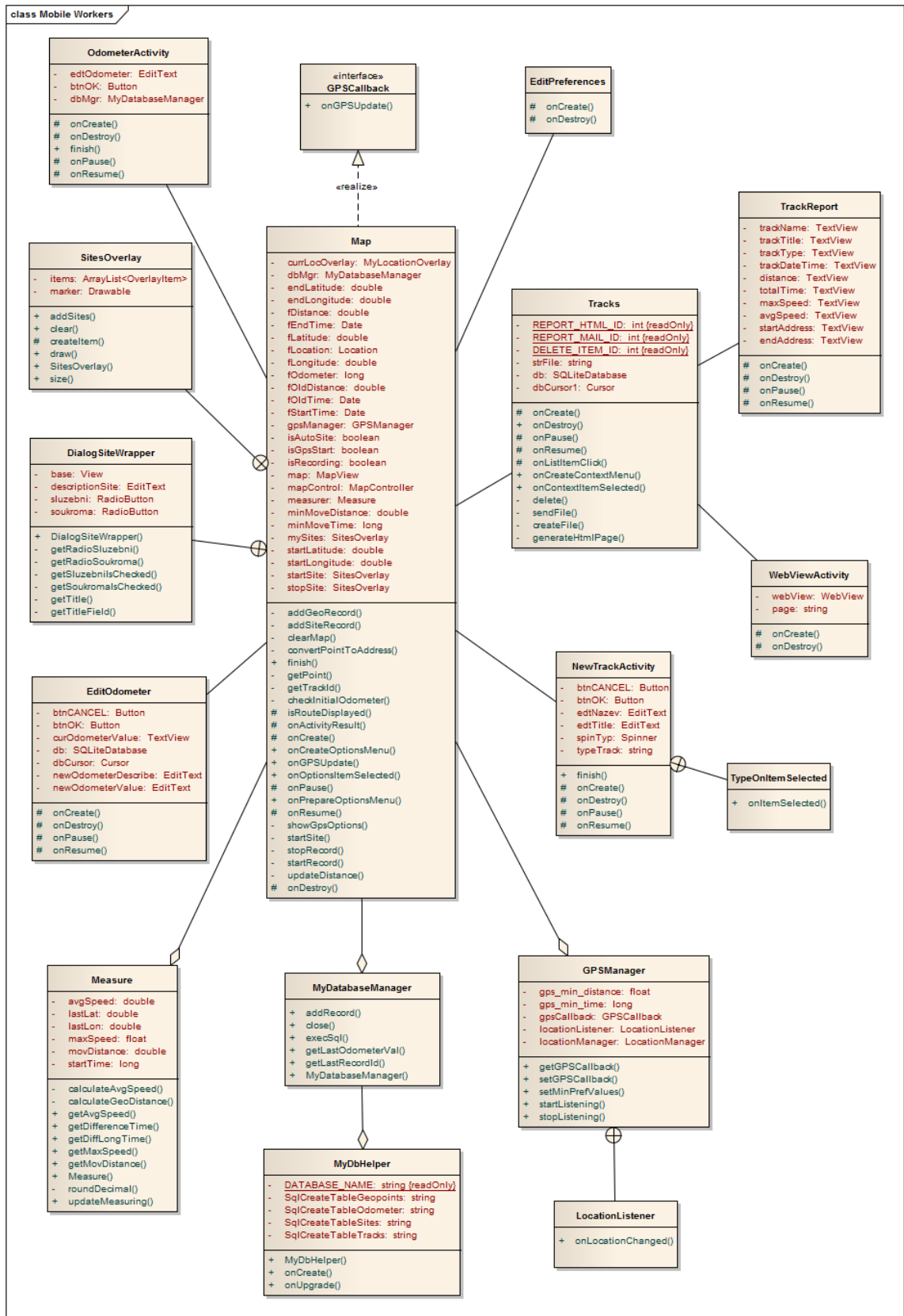
Specifikace preferencí jsou umístěné v souboru *res/xml/preferences.xml*. K vytvoření dochází v metodě *onCreate()* třídy *EditPreferences*:

voláním metody *addPreferencesFromResource()*. Tato třída vznikla rozšířením třídy *PreferenceActivity*. Vlastní přístup do preferencí z třídy *Map* je zajištěn při každém volání metody *onResume()*, jak je vidět na Obr. 26.

```
125 @Override
126 protected void onResume() {
127
128     super.onResume();
129
130     SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);
131
132     long prefMinTime = Long.valueOf(prefs.getString("list_time", "0"));
133     float prefMinDistance = Float.valueOf(prefs.getString("list_distance", "0"));
134
135     if (prefs.getBoolean("checkbox_site", false))
136         isAutoSite = true;
137     else
138         isAutoSite = false;
139
140     minMoveTime = Long.valueOf(prefs.getString("list_time_site", "15000"));
141     minMoveDistance = Double.valueOf(prefs.getString("list_distance_site", "0.03"));
142
143     gpsManager.setMinPrefValues(prefMinTime, prefMinDistance);
144
```

Obr. 26. Ukázka načtení preferencí v metodě *onResume()*

### 6.4 Návrh a popis tříd



Obr. 27. Model tříd

### 6.4.1 Třída Map

Toto je hlavní třída vytvořeného programu. V souboru *AndroidManifest.xml* je definována jako hlavní aktivita, která se zobrazí uživateli hned po spuštění aplikace. Třída je odvozena od třídy *MapActivity* a díky tomu, ošetřuje některé detaily vytvořené aktivity s widgetem typu *MapView* (mapový podklad).

Hlavními úkoly třídy jsou:

- zobrazovat aktuální polohou a symboly (začátek, konec, zastávka) na mapě ve spolupráci s vnitřní třídou *SitesOverlay*
- vytvořit strukturu menu a ošetřit události typu kliknutí na položky v menu
- komunikovat s třídou *GPSManager* a získávat aktuální GPS data v callback metodě *onGPSUpdate()*
- komunikovat s databází SQLite pomocí třídy *MyDatabaseManager*
- zajistit měření požadovaných údajů spolu s třídou *Measure*, která například: vypočítává průměrnou rychlost, či počítá ujetou vzdálenost, atd.
- provádí reverzní geokódování pomocí Google Maps API a třídy *Geocoder*
- komunikovat s ostatními aktivitami aplikace

#### Metody:

- *public void onCreate(Bundle)* – Volána vždy, když je aktivita prvně spuštěna, obsahuje inicializaci uživatelského rozhraní, vytvoření instancí *gpsManager*, *dbManager* a jednotlivých mapových vrstev pro zobrazení symbolů.
- *protected void onResume()* – Metoda volána vždy při přechodu aktivity do popředí. Obsahuje potřebné inicializace a načítá uživatelské nastavení aplikace z preferencí.
- *protected void onPause()* – Volána vždy při přechodu aplikace do pozadí, jsou zde příkazy pro odregistrování naslouchání změn GPS souřadnic a metody pro bezpečné uložení stavu aktivity.
- *public void finish()* – Volána při ukončování aktivity, zavírá se zde spojení s databází.

- *protected void onActivityResult(int, int, Intent)* – Pokud třída *Map* spustí jinou aktivitu (potomka), potom bude upozorněna na ukončení jejího provádění prostřednictvím této metody. V metodě je implementována rozhodovací struktura, která vybere podle jedinečného návratového kódu uvedeného jako parametr typu *integer* při volání aktivity (potomka) pomocí metody *startActivityForResult()*. Podle tohoto rozhodnutí jsou provedeny další operace.
- *public void onGPSUpdate(Location)* – Metoda zpracovává každou aktualizaci GPS souřadnic. Ukázka je zobrazena na Obr. 28.
- *public boolean onCreateOptionsMenu(Menu)* – Stará se o vytvoření struktury hlavního menu ze specifikace souboru XML. Naplnění zajišťuje třída *MenuInflater*.
- *public boolean onPrepareOptionsMenu(Menu)* – Je volána při každém stisku tlačítka menu.
- *public boolean onOptionsItemSelected(MenuItem)* – Metoda obsluhuje výběr položky z menu. Podle vyhodnocení konstrukce *switch*, která testuje *id* položky, na níž bylo kliknuto, vykonává konkrétní blok příkazů.
- *private void startSite()* – Metoda vytvoří dialogové okno sloužící k zadání zastávky. Po stisknutí tlačítka *ANO*, dojde k získání adresy místa dle GPS souřadnic, vytvoření nového symbolu a zobrazení na mapě, uložení dat o zastávce do databáze.
- *private void showGPSOptions()* – Zobrazí systémovou aktivitu sloužící k nastavení GPS.
- *private void updateDistance()* – Volá metodu *updateMeasuring()* třídy *Measure measurer.updateMeasuring(lat, lon, speed, time)*, která aktualizuje všechny měřené údaje o cestě.
- *private void addSiteRecord(DialogSiteWrapper, String)* – Přidává záznam popisující zastávku do databáze. Vlastní operaci *INSERT* do tabulky *table\_sites* provádí třída *MyDatabaseManager*.
- *private void addGeoRecord(Location)* – Ukládá každou změnu souřadnic do databázové tabulky *table\_geopoints*.

- *private void startRecord(String[])* – Zahajuje záznam jízdy, dojde k získání adresy místa dle GPS souřadnic, vytvoření nového symbolu a zobrazení na mapě, uložení dat do databáze.
- *private void clearMap()* – Vyčistí mapové vrstvy od symbolů. Výsledkem je prázdná mapa.
- *private int getTrackId()* – Vrací poslední id záznamu v tabulce *table\_tracks*.
- *private boolean checkInitialOdometer()* – Kontroluje hodnotu tachometru. Pokud je poslední hodnota k dispozici, tak ji použije jako aktuální. Pokud neexistuje žádný záznam v tabulce *table\_odometer*, potom se vyvolá aktivita pro zadání počáteční hodnoty tachometru. Typicky při prvním zpuštění aplikace, po instalaci.
- *private void stopRecord()* – Ukončuje záznam jízdy, dojde k získání adresy místa dle GPS souřadnic, vytvoření nového symbolu a zobrazení na mapě, uložení dat do databáze.
- *private GeoPoint getPoint(double, double)* – Metoda slouží k převodu jednoho typu zápisu zeměpisných souřadnic na druhý. Google Maps API používá reprezentaci typu *integer* (celé číslo). Android Location package reprezentuje souřadnice jako datový typ *double* (reálné číslo). Metoda vrací objekt typu *GeoPoint*, jenž obsahuje latitude a longitude jako celočíselné datové typy.
- *private String convertPointToAddress(GeoPoint)* – Zajišťuje převod zeměpisných souřadnic na adresu (reverzní geokódování), pokud je tato adresa v tomto místě k dispozici. Využívá k tomu rozhraní Google Maps API, které tuto funkcionalitu poskytuje pomocí třídy *Geocoder*. Ukázka použití je zobrazena na Obr. 33.

#### Vnitřní třídy:

- *private class SitesOverlay extends ItemizedOverlay<OverlayItem>* – Realizuje a vytváří vrstvy. Vrstva je oddělena od samotné mapy, a to, co je vidět, je kompozice vrstvy symbolů a vrstvy samotné mapy. Připojení překrývající vrstvy k vlastní mapě je řešeno voláním metody *getOverlays()* widgetu *MapView* a prostřednictvím metody *Add()* je vrstva přidána do mapy.
- *private class DialogSiteWrapper* – Tato třída je zde spíše z optimalizačních důvodů. V podstatě obaluje objekty typu *View*, které obsahuje dialogové okno pro zahájení

zastávky. K widgetum na dialogu se potom přistupuje skrze tuto třídu. V zásadě minimalizuje volání metody *findViewById()*, jejíž volání obnáší provedení velkého počtu instrukcí procesoru, a tím rychlejší vybíjení baterie. Více se o této metodě lze dočíst v publikaci. [6]

```

275         if (isRecording) {
276             addGeoRecord(fLocation);
277             updateDistance();
278
279             if (isAutoSite) {
280                 currentDistance = fDistance;
281
282                 currentTime = new Date();
283                 diffTime = measurer.getDiffLongTime(fOldTime, currentTime);
284
285                 if ( diffTime > minMoveTime) {
286                     distance = currentDistance - fOldDistance;
287
288                     if ((distance < minMoveDistance) && (currentSpeed > 0.2)) {
289                         startSite();
290                         distance = 0.0;
291                         fOldDistance = currentDistance;
292                         fOldTime = currentTime;
293                     }
294                     else {
295                         distance = 0.0;
296                         fOldDistance = currentDistance;
297                         fOldTime = currentTime;
298                     }
299                 }

```

Obr. 28. Ukázka části metody onGPSUpdate()

#### 6.4.2 Třída GPSManager

Třída implementuje řídicí logiku spojenou se zahájením, získáváním, ukončením a propojením GPS dat s dalšími třídami aplikace. Hlavním úkolem je tedy vytvořit spojení s GPS nebo jiným poskytovatelem lokalizačních služeb, a to pomocí systémové služby *LocationManager* a objektu typu *LocationProvider*.

##### Metody:

- *public GPSManager()* – Konstruktor vytváří objekt typu *LocationListener*, který ve své metodě *onLocationChanged()* zachytává změny GPS souřadnic a poskytuje je dál přes rozhraní *GPSCallback*.
- *public void startListening(final Activity)* – Metoda zahajuje naslouchání změn zeměpisné polohy zařízení. Vytváří se zde objekt typu *Criteria*, který vyjadřuje požadavky na vlastnosti dodavatele informací o zeměpisné poloze. Pokud nejsou tyto

požadavky splněny (není nalezen nejlepší možný poskytovatel), potom dochází k postupnému procházení všech dostupných poskytovatelů, aby byla získána alespoň přibližná zeměpisná poloha. Celý princip je ukázán na Obr. 29.

- `public void stopListening()` – Ukončuje naslouchání změn zeměpisné polohy, odebírá zaregistrovaný objekt typu `LocationListener`.
- `public void setGPSCallback(final GPSCallback)` – Nastavuje parametrem třídu, která implementuje rozhraní `GPSCallback` pro příjem změn zeměpisných souřadnic. V tomto případě třídu `Map`.
- `public GPSCallback getGPSCallback()` – Vrací aktuální třídu nastavenou jako příjemce v předchozí metodě.
- `public void setMinPrefValues(long, float)` – V metodě se nastavují intervaly minimálního času a minimální vzdálenosti pro aktualizaci GPS polohy.

```
65     final Criteria criteria = new Criteria();
66     criteria.setAccuracy(Criteria.ACCURACY_FINE);
67     criteria.setAltitudeRequired(false);
68     criteria.setBearingRequired(false);
69     criteria.setCostAllowed(true);
70     criteria.setPowerRequirement(Criteria.POWER_LOW);
71
72     final String bestProvider = locationManager.getBestProvider(criteria, true);
73
74     if (bestProvider != null && bestProvider.length() > 0)
75     {
76         locationManager.requestLocationUpdates(bestProvider, gps_min_time,
77             gps_min_distance, locationListener);
78     }
79     else
80     {
81         final List<String> providers = locationManager.getProviders(true);
82
83         for (final String provider : providers)
84         {
85             locationManager.requestLocationUpdates(provider, gps_min_time,
86                 gps_min_distance, locationListener);
87         }
88     }
```

Obr. 29. Ukázka části metody `startListening()`

### 6.4.3 Třídy *MyDatabaseManager* a *MyDbHelper*

Hlavním úkolem třídy *MyDatabaseManager* je vytvořit vrstvu nad samotnou databází a implementovat metody pro manipulaci s daty. Avšak v některých třídách není tento princip přístupu k databázi striktně dodržován.

*MyDbHelper* rozšiřuje třídu *SQLiteOpenHelper*. Hlavním účelem této třídy je vytvořit samotnou databázi a tabulky pro uložení dat.

#### Metody třídy *MyDatabaseManager*:

- *public MyDatabaseManager(Context)* – Konstruktor vytváří instanci třídy *MyDbHelper* a nad ní volá metodu *getWritableDatabase()*, která přiděluje právo zápisu do databáze.
- *public void addRecord(ContentValues, String)* – Metoda umožňuje přidat záznam do databáze. Hodnoty sloupců jsou uloženy v datové struktuře typu *ContentValues*.
- *public int getLastRecordId(String)* – Vrací poslední id záznamu v tabulce, jejíž název je předáván v parametru metody.
- *public void execSql(String)* – Metoda zajišťuje vykonání SQL příkazu.
- *public long getLastOdometerVal(int)* – Vrací poslední hodnotu tachometru.
- *public void close()* – Zavírá databázové spojení.

#### Metody třídy *MyDbHelper*:

- *public MyDbHelper(Context)* – Konstruktor získává ze zdrojů z adresáře: *res/values/string/string.xml* SQL příkazy na vytvoření databáze a potřebných tabulek.
- *public void onCreate(SQLiteDatabase)* – Metoda při prvním spuštění aplikace vytváří vlastní databázové tabulky.
- *public void onUpgrade(SQLiteDatabase, int, int)* – Je volána v případě přechodu na novou verzi databáze.

#### 6.4.4 Třída Measure

Tato třída se stará o výpočet ujeté vzdálenosti, průměrné a maximální rychlosti, času záznamu. Přijímá data od třídy *Map* a zpracovává je.

##### Metody:

- *public Measure()* – Konstruktor inicializuje všechny proměnné na počáteční hodnoty.
- *public void updateMeasuring(double, double, float, long)* – Aktualizuje měřené hodnoty od začátku záznamu jízdy. Je volána třídou *Map* po každé změně zeměpisných souřadnic.
- *public double getMovDistance()* – Metoda vrací hodnotu dosud uražené vzdálenosti.
- *public double getAvgSpeed()* – Metoda vrací hodnotu průměrné rychlosti.
- *public double getMaxSpeed()* – Metoda vrací hodnotu maximální rychlosti.
- *private double calculateAvgSpeed(long, double)* – Vypočítá průměrnou rychlost.
- *private double calculateGeoDistance(float, float, float, float)* – Využívá algoritmus pro výpočet vzdálenosti mezi dvěma body pomocí Haversinovi formule. [17]
- *private double roundDecimal(double, int)* – Zaokrouhluje hodnotu typu *double* na hodnotu s počtem desetinných míst daných parametrem typu *int*.
- *public String getDifferenceTime(Date, Date)* – Metoda vrací rozdíl časů formátovaný v datovém typu *String*.
- *public long getDiffLongTime(Date, Date)* – Metoda vrací rozdíl časů jako *long*.

#### 6.4.5 Třída Tracks a TrackReport

Třída *Tracks* rozšiřuje třídu *ListActivity* a zobrazuje přehled všech uskutečněných jízd. Aktivita obsahuje seznam jízd tvořený pomocí výčtů systému Android a widgetu typu *ListView*, kde každou jízdu reprezentuje jeden řádek. Specifikace každého řádku seznamu je definována v souboru *res/layout/row\_track.xml*. Potřebná data z databáze určená k zobrazení dodává adaptér typu *SimpleCursorAdapter*. Adaptéry systému Android jsou zodpovědné za dodání soupisu dat widgetům nabídek a za konvertování jednotlivých datových položek na specifické náhledy typu *View*, které se ve widgetech nabídek zobrazují jako jednotlivé řádky. Widgety nabídek jsou například již zmíněný widget *ListView*, *Spinner*, apod.

*Tracks* dále implementuje kontextové menu, pomocí kterého mohou být volány různé metody. Zobrazení detailního reportu z cesty v internetovém prohlížeči systému Android. Dále je možné odeslat report z cesty jako přílohu e-mailu ve formátu HTML. Ukázka zdrojového kódu sloužící k odeslání reportu je na Obr. 30.

Třída *TrackReport* je v podstatě aktivita obsahující widgety typu *TextView* určené k zobrazení informací o cestě. Data jsou této aktivitě předána v intentu volající třídou *Tracks*.

```
164         case REPORT_MAIL_ID:
165         {
166             AdapterView.AdapterContextMenuInfo info=
167                 (AdapterView.AdapterContextMenuInfo)item.getContextMenuInfo();
168
169             try {
170
171                 strFile = Environment.getExternalStorageDirectory().getAbsolutePath() + "/temp";
172
173                 File file = new File(strFile);
174
175                 if (!file.exists()) {
176                     file.mkdirs();
177                     Log.i("mkdir", "adresar vytvoren");
178                 }
179
180                 strFile = strFile + "/report.html";
181                 createFile(info.id);
182                 sendFile();
183
184             } catch (Throwable t) {
185                 Log.i("Chyba při přístupu na SD kartu", t.getMessage());
186             }
187
188             sendFile();
189
190             return (true);
191         }
```

Obr. 30. Ukázka odeslání reportu na e-mail

### Metody třídy *Tracks*:

- *protected void onCreate(Bundle)* – Volána vždy, když je aktivita prvně spuštěna. Provádí se zde databázový dotaz, jehož výsledek je databázový kurzor. Ten je předán adaptéru typu *SimpleCursorAdapter*, který zajišťuje naplnění seznamu jízd.
- *protected void onDestroy()* – Metoda volána při vypnutí aktivity.
- *protected void onPause()* – Odpojení databáze a kurzoru.
- *protected void onItemClick(ListView, View, int, long)* – Obsluha kliknutí na řádek seznamu. Je zde vytvořen intent s daty konkrétní jízdy a odeslán třídě *TrackReport*.
- *public void onCreateContextMenu(ContextMenu, View, ContextMenuInfo)* – Metoda vytváří kontextové menu.

- *public boolean onOptionsItemSelected(MenuItem)* – Obsluhuje výběr položky z kontextového menu.
- *private void delete(long)* – Maže záznam jízdy z databáze.
- *private void sendFile()* – Metoda odesílá soubor přes e-mail pomocí intentu.
- *private void createFile(long)* – Vytváří soubor pomocí streamu.
- *private String generateHtmlPage(long)* - Metoda generuje HTML stránku, která obsahuje tabulky (vlastní report z cesty) a vrací jí jako datový typ *String*.

#### Metody třídy **TrackReport**:

- *protected void onCreate(Bundle)* - Volána vždy, když je aktivita prvně spuštěna. Vytvoření instancí *TextView* voláním metody *findViewById()* a načtení dat z intentu.

#### 6.4.6 Třída **NewTrackActivity** a **WebViewActivity**

Třída *NewTrackActivity* reprezentuje aktivitu, která je zobrazena uživateli, když se rozhodne zahájit jízdu stiskem tlačítka v hlavním menu aplikace. Aktivita je volána z metody *onOptionsItemSelected()* třídy *Map* příkazem *startActivityForResult()*. Tento způsob volání podřízené aktivity může být použit k upozornění na ukončení spuštěné podaktivity. Toto upozornění spolu s daty vyplněného formuláře *NewTrackActivity* jsou k dispozici v callback metodě *onActivityResult()* třídy *Map*.

*WebViewActivity* obsahuje widget typu *WebView*, který zajišťuje zobrazení HTML stránky s vygenerovaným reportem.

#### Metody třídy **NewTrackActivity**:

- *protected void onCreate(Bundle)* – Metoda zajišťuje vytvoření formuláře, nastavení návratové odpovědi *RESULT\_OK* nebo *RESULT\_CANCELED* a odeslání dat pomocí intentu zpět volající aktivitě, tedy třídě *Map*.

*WebViewActivity* přijímá data „přibalená“ k odeslanému intentu volající třídou *Tracks* a zobrazuje je jako report z cesty v prohlížeči. Android umožňuje do aktivity vložit zabudovaný webový prohlížeč jako widget, který je založený na jádře *WebKit*. Jeho specifikace je uložena v adresáři *res/layout/webview.xml*.

### Metody třídy `WebViewActivity`:

- `public void onCreate(Bundle)` – Vytvoří zabudovaný webový prohlížeč a zobrazí v něm informace o cestě.

#### 6.4.7 Třída `EditOdometer` a `OdometerActivity`

První třída umožňuje editaci hodnoty tachometru pomocí jednoduchého formuláře. Druhá je vyvolána pouze při prvním spuštění aplikace, když ještě není známa počáteční hodnota tachometru. Třída obsahuje pouze základní metody, které obsahuje každá aktivita, o nichž už byla řeč dříve. Není tedy nutné se jimi blíže zabývat.

#### 6.4.8 Třída `EditPreferences`

`EditPreferences` rozšiřuje třídu `PreferenceActivity` systému Android, která zprostředkovává uživatelské rozhraní pro nastavení možností aplikace. Specifikace preferencí jsou v adresáři `res/xml/preferences.xml`. Jejich načtení a zobrazení je zajištěno voláním metody `addPreferencesFromResource()` při spuštění aktivity.

### 6.5 Databáze

Pro ukládání veškerých informací o zaznamenaných jízdách byl zvolen nativní databázový systém SQLite. Při práci s databází byl nejčastěji využíván program SQLite Manager, který je dostupný jako zásuvný plugin do webového prohlížeče Mozilla Firefox. Databázový soubor je umístěn v adresáři:

```
data/data/gm.tichy.mobileworkers/databases/dbmobileworkers
```

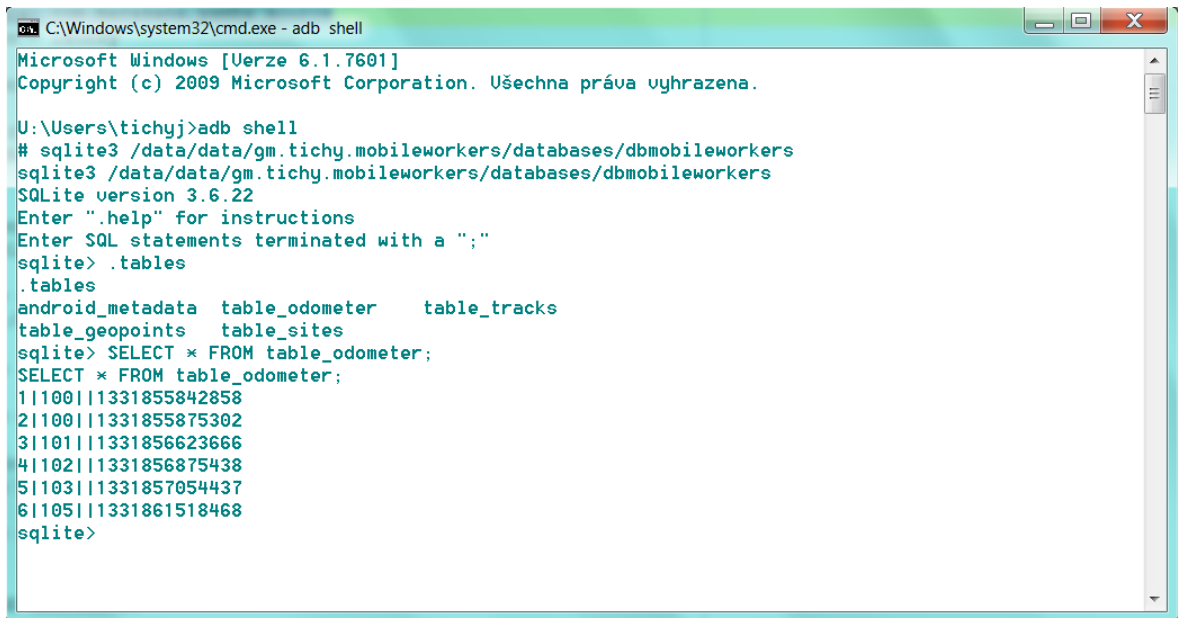
Existují dvě cesty, kterými lze kopírovat soubor do počítače pro potřeby testování a manipulaci s obsahem databáze.

1. Využití DDMS v prostředí Eclipse, který obsahuje jednoduchý vizuální nástroj *File Explorer* ke kopírování souborů z/do zařízení nebo emulátoru.
2. Pomocí příkazů `adb pull` a `adb push` z příkazové řádky.

Celý příkaz pro vykopírování souboru může vypadat takto:

```
U:\Users\tichyj>adb pull /data/data/gm.tichy.mobileworkers/databases/dbmobileworkers
```

Rychlejší, ale méně přívětivý přístup k obsahu databáze je umožněn pomocí příkazu *adb shell*, který spouští v emulátoru vzdálený interpret (shell) k vykonávání příkazů. Emulátor v sobě zahrnuje konzolový program *sqlite3* pro práci s databází. Ukázka použití tohoto nástroje je na Obr. 31.



```
C:\Windows\system32\cmd.exe - adb shell
Microsoft Windows [Verze 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Všechna práva vyhrazena.

U:\Users\tichyj>adb shell
# sqlite3 /data/data/gm.tichy.mobileworkers/databases/dbmobileworkers
sqlite3 /data/data/gm.tichy.mobileworkers/databases/dbmobileworkers
SQLite version 3.6.22
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> .tables
.tables
android_metadata  table_odometer  table_tracks
table_geopoints  table_sites
sqlite> SELECT * FROM table_odometer;
SELECT * FROM table_odometer;
11100||1331855842858
21100||1331855875302
31101||1331856623666
41102||1331856875438
51103||1331857054437
61105||1331861518468
sqlite>
```

Obr. 31. Ukázka připojení k databázi pomocí příkazové řádky

### 6.5.1 Návrh a popis

Navržená struktura databáze obsahuje 4 tabulky, které jsou mezi sebou propojeny relačními vztahy.

#### Tabulka *table\_tracks*:

V této tabulce jsou uloženy všechny údaje z jednotlivých jízd. Jsou zde obsaženy základní údaje vyplněné uživatelem před zahájením jízdy (název, typ, účel jízdy), údaje automaticky doplněné aplikací (datum, adresy, stavy tachometru) a vypočítané hodnoty během záznamu jízdy (ujeté kilometry, max. a prům. rychlosti, celkový čas). Primárním klíčem byl zvolen sloupec *\_id*.

#### Tabulka *table\_sites*:

Tabulka přiřazuje jednotlivé zastávky, které vznikly během jízdy. Cizím klíčem je *track\_fk\_id* a primárním klíčem identifikátor zastávky *site\_pk\_id*. Vztah mezi tabulkou *table\_tracks* a touto tabulkou je 1:N (za každou jízdu může být uskutečněno 0 až N

zastávek). U každého záznamu je uloženo datum, čas, typ, adresa místa, zeměpisné souřadnice.

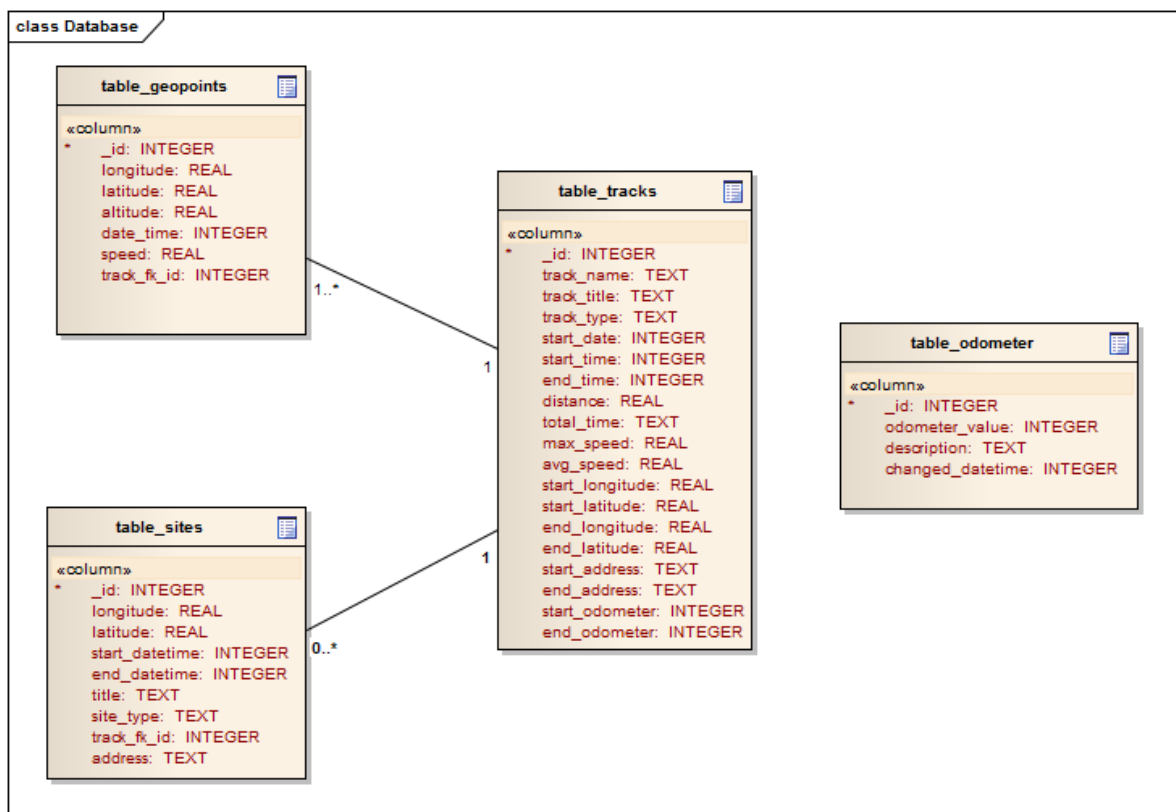
### Tabulka *table\_geopoints*:

Jednotlivé záznamy polohy, určené pomocí zeměpisné šířky a délky, jsou uloženy v tabulce *table\_geopoints*. Jako primární klíč bylo zvoleno pořadové číslo záznamu *geo\_pk\_id*. Vztah mezi tabulkami *table\_tracks* a *table\_geopoints* je 1:N, jedna jízda může mít několik až N záznamů.

### Tabulka *table\_odometer*:

Tato tabulka eviduje všechny změny hodnot tachometru od prvního spuštění aplikace až po poslední uskutečněnou jízdu. Tabulka je samostatná bez vazby na ostatní tabulky. Jako primární klíč používá sloupec s názvem *\_id*.

Výsledný ER-diagram databáze je uveden níže na Obr. 32.



Obr. 32. Datový model

## 6.6 Použité komponenty, knihovny a služby

Při vývoji aplikace byla použita služba Google Maps společnosti Google. Hlavní důvody pro výběr bylo to, že má výborně zpracovanou dokumentaci a její propojení se systémem Android je na skvělé úrovni. Mapové podklady jsou dodávány spolu s Google Maps API. API poskytuje funkcionalitu dostupnou prostřednictvím knihovny tříd *com.google.android.maps*. Licenční podmínky umožňují libovolné využití mapových podkladů pro nekomerční účely. Komerční využití je povoleno pouze držitelům *Google enterprise licence*. Jinou alternativou integrace map do aplikací založené na jiných zdrojích je například projekt OpenStreetMap.

Od verze systému Android 1.5 není služba součástí prostředí SDK. Místo toho je součástí přídatného modulu, které rozšiřuje dodávané SDK. Při vývoji aplikace je nutné učinit následující kroky:

1. Vytvořit projekt s vhodným cílem, aby bylo zajištěno, že API služby Google Maps budou k dispozici. Aplikace Mobile Workers byla testována na konfiguraci AVD s cílem API Level 10 (odpovídá verzi Android 2.3.3).
2. Integrovat mapy do aplikace vytvořením vlastní podtřídy třídy *MapActivity* a XML návrhu s widgetem *MapView*.

### 6.6.1 Integrace Google Maps

Základem je odvozená podtřída třídy *MapActivity*. Dále musí být vytvořen XML návrh aktivity, v kterém je specifikován element *MapView*. Element zahrnuje atribut *android:apiKey*, který musí odkazovat na API klíč služby Google Maps. Onen klíč je nutné získat pro zpřístupnění služby následujícím postupem:

V první řadě se musí vytvořit hašovacím algoritmem MD5 otisk certifikátu, který je používán při podepisování aplikací digitálním podpisem vygenerovaným z certifikátu. Tento certifikát je součástí instalace SDK. Vlastní podepisovací proces vyžaduje použití Java utility *keytool*.

MD5 otisk vygeneruje tento příkaz:

```
keytool -list -alias androiddebugkey -keystore <cesta> -storepass android -keypass android
```

hodnota přepínače *-keystore* musí být nahrazena příslušnou lokací platnou pro danou platformu.

Ve Windows Vista/7 se nachází v umístění: *C:\Users\<uživatel>\.android\debug.keystore*.

Výstupem výše uvedeného příkazu je otisk vyjádřený sérií párů hexadecimálních čísel oddělených dvojtečkami. Ten je potřeba vložit na stránce pro přihlášení API klíče a odeslat formulář. Výsledná stránka vrátí potřebný klíč, který již stačí vložit jako hodnotu atributu *android:apiKey* do widgetu *MapView* v XML návrhu aktivity. Adresa stránky pro vygenerování API klíče byla v době registrace:

*<http://code.google.com/intl/cs-CZ/android/add-ons/google-apis/index.html>*

Kromě tohoto je nutné v souboru *AndroidManifest.xml* nakonfigurovat další dvě věci:

1. Přidat oprávnění *INTERNET*, *ACCESS\_COARSE\_LOCATION* a *ACCESS\_FINE\_LOCATION*
2. Do elementu *<application>* přidat element *<uses-library>* s atributem *android:name = "com.google.android.maps"*, čímž aplikaci sdělíme, že používáme jedno z volitelných API systému Android.

### 6.6.2 Reverzní geokódování

Geokódování je proces identifikace zeměpisné polohy, při které se na základě umístění daného adresou, městem, apod. vyhledají zeměpisné souřadnice. Pro účely aplikace Mobile Workers ke zjištění adresy a města začátku či konce jízdy bylo využito obráceného postupu, který se nazývá reverzní geokódování. Google Maps API podporuje vyhledávání pro území České republiky do úrovně ulic. Ukázka zdrojového kódu použití reverzního geokódování je na Obr. 33.

```
708     Geocoder geocoder = new Geocoder(getBaseContext(), Locale.getDefault());
709     try {
710         List<Address> address = geocoder.getFromLocation(point.getLatitudeE6() / 1E6,
711             point.getLongitudeE6() / 1E6, 1);
712
713         if (address.size() > 0) {
714
715             if (address.get(0).getThoroughfare() != null)
716                 resultAddress += address.get(0).getThoroughfare() + " ";
717             if (address.get(0).getFeatureName() != null)
718                 resultAddress += address.get(0).getFeatureName() + ", ";
719             if (address.get(0).getSubAdminArea() != null)
720                 resultAddress += address.get(0).getSubAdminArea();
721         }
722     }
723     catch (IOException e) {
724         e.printStackTrace();
725         Toast.makeText(getBaseContext(), "chyba", Toast.LENGTH_LONG).show();
726     }
727     finally {
```

Obr. 33. Ukázka třídy Geocoder

## 6.7 Problémy při implementaci

Největším problémem, který se při implementaci vyskytl, byla nedostupnost služby umožňující reverzní geokódování pomocí třídy *Geocoder* v emulátoru. V telefonu takový problém nenastal. Toto omezení bylo velkou překážkou hlavně v počátcích vývoje aplikace, kdy se převážně testovalo v emulátoru, nikoli ve fyzickém zařízení. Bylo vyzkoušeno několik konfigurací s různými cíli API, ale všechny selhávaly stejným způsobem. V okamžiku dotazu na aktuální souřadnice, dojde k vyvolání výjimky typu *IOException: Service not Available*. Při hledání řešení bylo zjištěno, že se jedná o známý problém rozhraní geocoding API, který je pravděpodobně na straně serveru společnosti Google.

Dalším méně závažným problémem by mohlo být to, že Google Geocoding API na své serverové straně limituje počet požadavků za den. Je uváděno číslo 2.500 dotazů za den, poté jsou dotazy blokovány. Tuto hodnotu se nám však při testování nepodařilo překročit.

Komplikace se objevily i na fyzickém zařízení. Testovacím telefonem se stal HTC Desire HD. Už od začátku byl problém s dlouhou fixací GPS, i když byly zapnuty všechny dostupné prostředky k urychlení získání signálu. Signál se podařilo zachytit v rozmezí 15 až 30 minut nebo vůbec. Trvalo to déle, než „studený start“, při kterém se stahují informace o poloze satelitů přímo z družic. Ani přesnost a kvalita signálu nebyla zrovna uspokojivá. Během jízdy došlo mnohokrát k opakované ztrátě signálu. Problém se podařilo vyřešit až po instalaci utility *GPS Status*, která poskytuje nástroje k ladění GPS. Pomocí tohoto programu byla

vymazána mezipaměť GPS dat a následně stažena nová A-GPS data přes internet z asistenčního centra. Poté byl start GPS mnohem rychlejší.

## ZÁVĚR

Cílem práce bylo vytvořit aplikaci pro mobilní zaměstnance, kteří podnikají soukromé nebo služební cesty firemním automobilem a musejí evidovat knihu jízd. Aplikace, pro kterou byl zvolen název Mobile Workers, by měla umožňovat sledovat trasu mobilního telefonu podle GPS a zaznamenávat ji do databáze v telefonu. V programu byly implementovány algoritmy pro inferenci významných průchozích bodů cesty včetně jejich pojmenování. Nasbíraná data jsou po skončení jízdy zpracována a reprezentována v podobě přehledných reportů a statistik z jednotlivých cest s možností jejich odeslání na e-mail. Aplikace byla vytvořena jako nativní pro mobilní platformu Android podle specifikace v zadání. Součástí práce byla analýza dostupného software zabývající se touto oblastí použití. Byla zjištěna skutečnost taková, že prakticky neexistuje zdarma dostupná aplikace pro českého uživatele s dostatečně intuitivním ovládáním, která by poskytovala uspokojivý výstup.

Ve fázi programování vznikly problémy hlavně při simulování polohy zařízení v emulátoru vývojového prostředí. Pro potřeby testování bylo zaznamenáno několik různě dlouhých tras ve formátu souboru GPX prostřednictvím programu GPS tracker. Emulátor umožňuje automaticky načítat jednotlivé souřadnice zeměpisné polohy ze souboru a posílat je do zařízení. Tímto způsobem lze dosáhnout simulovaného pohybu telefonu na mapě. Ostatní zpřesňující informace jako rychlost, přesný čas a nadmořská výška však simulovat nelze. Další komplikace byly způsobeny nedostupností služby reverzního geokódování přes Google Maps API. Z důvodů těchto nepříjemností jsem byl nucen v mnoha případech testovat na fyzickém zařízení, což v konečném důsledku prodlužovalo samotný vývoj aplikace.

Pro zobrazení dat na mapě bylo vzato v úvahu několik poskytovatelů mapových podkladů. Jak se nakonec ukázalo, jediným vyhovujícím poskytovatelem pro platformu Android jsou Google Maps, které nabízejí potřebné knihovny pro práci s mapou a integrují mapové funkce do prostředí Android.

Vytvořená aplikace byla podrobena několika testovacím jízdám. Výsledky tohoto malého testu pomohly při vytváření plánu dalšího vývoje aplikace. Výslednou aplikaci tedy nelze považovat za konečné řešení. Do budoucna se předpokládají další úpravy vedoucí k tomu, aby aplikace byla schopná plně nahradit klasickou, papírově vedenou knihu jízd. Dále by aplikace mohla být rozšířena o webovou část, kde by uživatelé měli možnost přihlášení pod vlastním uživatelským účtem.

## ZÁVĚR V ANGLIČTINĚ

The aim of my thesis was to create an application for mobile workers who use company cars for business trips as well as for private purposes and they have to keep a route book. The application called Mobile Workers should allow tracking mobile routes according to GPS, after its activation, and recording it to mobile database. There have been some algorithms implemented for interference of significant route points including their names in the application. Collected data are processed after each route and shown in the form of reports and statistics with the possibility of sending them to an email. This application was created originally for Android platform under the requested specification. The available software analysis regarding this usage area was part of this thesis. It was found out that there is no such a free application for a Czech users with sufficient intuitive management providing favourable output.

There have been some problems mainly at equipment position simulation in the emulator. Therefore, some routes of various length in the GPX format via GPS tracker for the testing purposes have been recorded. The emulator allows automatic loading of individual coordinates of geographical location from a file and sending them to the equipment so that it was possible to simulate mobile movement on the map. The other detailed information such as velocity, exact time and altitude is not possible to simulate. Additional complication was caused by inaccessible service of reverse geocoding via Google Maps App, so testing had to be done on an actual equipment. This also prolonged the application development.

Various map providers were taken in account for displaying data on the map. It was found out that there was only one suitable provider for Android platform and it was Google maps, because it offers the needed API and integrates map functions for Android.

The created application was subjected to several test drives. The test results helped to create a new plan for developing this app. The resulting should not be considered as a final solution. After some additional adjustments it can replace the classical paper form. The application could be also extended to the web part where a user could register their account.

**SEZNAM POUŽITÉ LITERATURY**

- [1] History of Android. *Bright hub* [online]. 2011 [cit. 2012-04-13]. Dostupné z: <<http://www.brighthub.com/mobile/google-android/articles/18260.aspx>>
- [2] Industry Leaders Announce Open Platform for Mobile Devices. *Open handset alliance* [online]. 2007 [cit. 2012-04-09]. Dostupné z: <[http://www.openhandsetalliance.com/press\\_110507.html](http://www.openhandsetalliance.com/press_110507.html)>
- [3] What is Android?. *Android developers* [online]. 2012 [cit. 2012-04-11]. Dostupné z: <<http://developer.android.com/guide/basics/what-is-android.html>>
- [4] Application Fundamentals. *Android developers* [online]. 2012 [cit. 2012-04-11]. Dostupné z: <<http://developer.android.com/guide/topics/fundamentals.html>>
- [5] Activities. *Android developers* [online]. 2012 [cit. 2012-04-10]. Dostupné z: <<http://developer.android.com/guide/topics/fundamentals/activities.html>>
- [6] MURPHY, Mark L. *Android 2: průvodce programováním mobilních aplikací*. Vyd. 1. Brno: Computer Press, 2011, 375 s. ISBN 978-80-251-3194-7.
- [7] MEDNIEKS, DORNIN, MEIKE a NAKAMURA. *Programming Android*. 1st ed. Sebastopol: O'Reilly, 2011, 482 s. ISBN 978-1-449-38969-7.
- [8] Obtaining User Location. *Android developers* [online]. 2012 [cit. 2012-04-14]. Dostupné z: <<http://developer.android.com/guide/topics/location/obtaining-user-location.html>>
- [9] ZAKHOUR, Sharon. *Java 6: výukový kurz*. Vyd. 1. Brno: Computer Press, 2007, 534 s. ISBN 978-80-251-1575-6.
- [10] RAPANT, Petr. *Družicové polohové systémy*. Vyd. 1. Ostrava: Vysoká škola báňská - Technická univerzita, 2002, 197 s. ISBN 80-248-0124-8.
- [11] FRENCH, Gregory T. *Understanding the GPS an introduction to the Global Positioning System*. 1st ed. Bethesda, MD: GeoResearch, 1996. ISBN 09-655-7230-7.
- [12] *U.S. Coast Guard Navigation Center* [online]. [cit. 2012-04-16]. Dostupné z: <<http://www.navcen.uscg.gov/>>

- [13] *My car tracks* [online]. [cit. 2012-04-17]. Dostupné z:  
<<http://www.mycartracks.com>>
- [14] Reference. *Android developers* [online]. 2012 [cit. 2012-04-21]. Dostupné z:  
<<http://developer.android.com/reference/packages.html>>
- [15] *Stackoverflow: Android* [online]. 2012 [cit. 2012-03-12]. Dostupné z:  
<<http://stackoverflow.com/questions/tagged/android>>
- [16] PECINOVSKEÝ, Rudolf. *Myslíme objektově v jazyku Java: kompletní učebnice pro začátečníky*. 2., aktualiz. a rozš. vyd. Praha: Grada, 2009, 570 s. ISBN 978-80-247-2653-3.
- [17] Calculate distance between two points on a globe. *Code codex: the free code wiki* [online]. 2012 [cit. 2012-03-10]. Dostupné z:  
<[http://www.codecodex.com/wiki/Calculate\\_Distance\\_Between\\_Two\\_Points\\_on\\_a\\_Globe#Java](http://www.codecodex.com/wiki/Calculate_Distance_Between_Two_Points_on_a_Globe#Java)>

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

ADT	Android Development Tools
API	Application Programming Interface (rozhraní pro programování aplikací)
APK	Android Application Package
AVD	Android Virtual Device
BTS	Base transceiver station
CPU	Central Processing Unit (mikroprocesor)
CSV	Comma-Separated Values
DDMS	Dalvik Debug Monitor Server
DEX	Dalvik Executable
DVM	Dalvik Virtual Machine
EDGE	Enhanced Data Rates for GSM Evolution
GNU	GNU's Not Unix (GNU není Unix)
GPS	Global Positioning System (globální družicový polohový systém)
GPRS	General Packet Radio Service
GPX	GPS eXchange Format
GUI	Graphical User Interface (grafické uživatelské rozhraní)
HTML	HyperText Markup Language
IDE	Integrated Development Environment
JVM	Java Virtual Machine
KML	Keyhole Markup Language
MIME	Multipurpose Internet Mail Extensions
PDF	Portable Document Format
PRN	Pseudo Random Number
SDK	Software Development Kit

S-JTSK    Systém Jednotné Trigonometrické Sítě Katastrální

SQL        Structured Query Language

TTFE      Time To First Fix

XML        Extensible Markup Language

WGS        World Geodetic System

**SEZNAM OBRÁZKŮ**

Obr. 1. Vrstvy operačního systému Android [3] .....	12
Obr. 2. Životní cyklus aktivity [5].....	15
Obr. 3. Intent filtr aktivity.....	17
Obr. 4. Ukázka implementace objektu LocationListener [8].....	19
Obr. 5. Vývojové prostředí Eclipse – nástroj DDMS.....	22
Obr. 6. TMCZ Drivebook.....	30
Obr. 7. Auto Mileage Log.....	31
Obr. 8. Travel LogBook .....	33
Obr. 9. My Car Tracks [13].....	34
Obr. 10. AndiCar .....	35
Obr. 11. Souhlas k instalaci.....	39
Obr. 12. Nastavení tachometru.....	40
Obr. 13. Korekce tachometru.....	40
Obr. 14. Hlavní obrazovka .....	41
Obr. 15. Menu aplikace .....	42
Obr. 16. Nastavení aplikace.....	44
Obr. 17. Dialog pro přidání zastávky .....	45
Obr. 18. Kontextové menu.....	46
Obr. 19. Zahájení jízdy .....	48
Obr. 20. Mapa se symboly .....	48
Obr. 21. Seznam jízd.....	48
Obr. 22. Detail jízdy .....	48
Obr. 23. Ukázka vytvoření a odeslání záměru (intentu) .....	51
Obr. 24. Adresářová struktura projektu .....	52
Obr. 25. Ukázka XML návrhu obrazovky .....	53
Obr. 26. Ukázka načtení preferencí v metodě onResume().....	54
Obr. 27. Model tříd .....	55
Obr. 28. Ukázka části metody onGPSUpdate().....	59
Obr. 29. Ukázka části metody startListening().....	60
Obr. 30. Ukázka odeslání reportu na e-mail.....	63
Obr. 31. Ukázka připojení k databázi pomocí příkazové řádky .....	66
Obr. 32. Datový model .....	67

---

Obr. 33. Ukázka třídy Geocoder..... 70

## SEZNAM PŘÍLOH

- I. Elektronická příloha

## **PŘÍLOHA P I: ELEKTRONICKÁ PŘÍLOHA**

Obsah přiloženého CD:

*bin/* Instalátor programu.

*src/* Kompletní zdrojové texty programu se všemi potřebnými knihovnami a dalšími soubory pro bezproblémové vytvoření spustitelných verzí programu.

*data/* Testovací data (trasy) ve formátu GPX použité při testování v emulátoru.

*readme.txt/* Instrukce pro instalaci a spuštění programu.

*fulltext.pdf/* Diplomová práce ve formátu PDF.