TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2646 – Informační technologie
Studijní obor: Informační technologie

Aplikace pro sledování pozice v mapě pro OS Android

Application for tracking the position on the map for Android OS

Bakalářská práce

Autor: **Petr Matouš**
Vedoucí práce: Ing. Přemysl Svoboda
Konzultant: Ing. Martin Vlasák

V Liberci 19. 5. 2011

Vzhledem ke správnému číslování obsahu a stránek. Tato strana nebude obsahem BP a bude nahrazena oficiálním zadáním.

Prohlášení

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

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

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

Bakalářskou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum

Podpis

Poděkování

Chtěl bych poděkovat mému vedoucímu práce za čas, který věnoval jako pomoc při vypracování mé bakalářské práce. Jeho vědomosti a poznámky mi byly velice užitečné.

Abstrakt

Cílem práce je vytvořit plně funkční aplikaci pro operační systém Android a otestovat ji na mobilním telefonu, který tento OS podporuje. Tato aplikace bude představovat sledovací zařízení vašeho vozu, či motocyklu, nebo čehokoli, co bude obsahovat GSM modul, který je schopen pravidelně odesílat SMS zprávy s aktuální GPS souřadnicí. Souřadnice budou zobrazovány v mapě a ukládány do databáze v mobilním telefonu. Bude možno rozlišovat mezi více samostatnými sadami GPS souřadnic. Tyto bude možno zpětně zobrazovat na display zařízení. Aplikaci bude také možno deaktivovat, aby nekontrolovala příchozí SMS zprávy.

Dále zde budou vysvětleny základní programovací principy a tvorba samotného projektu pro OS Android. Seznámíte se s Android SDK a s programovacím prostředím Eclipse, které umí tento balíček využít. V neposlední řadě pak s balíčkem pro práci s mapovými podklady Google API. Čtenář by měl získat základní přehled a být schopný začít vytvářet aplikace pro OS Android.

Klíčová slova

Android OS, mobilní zařízení, GPS, sledování, Google mapy, Java

Abstract

The goal is to create a fully functional application for the Android operating system and test it on a mobile phone that supports this OS. This application is going to represent a tracking device of your car or motorcycle or anything that will contain a GSM module that is able to periodically send SMS messages with current GPS coordinates. Coordinates are displayed in the map and stored in a database on a mobile phone. There will be possibility to distinguish between multiple separate sets of GPS coordinates. These can be re-displayed on the device. You will also be able to deactivated application to not to check incoming SMS messages.

Basic principles of programming and development project for Android OS will be also explained. We meet with Android SDK (software development kit) and the Eclipse programming environment that can take advantage of this package. Last but not least, a package for working with maps with the Google API. The reader should gain basic knowledge and be able to begin creating applications for Android.

Keywords

Android OS, mobile devices, GPS, tracking, Google Maps, Java

Obsah

1 Úvod

V dnešní době, kdy většina standardních mobilních telefonů začíná být hromadně nahrazována dotykovými telefony, vznikl operační systém šitý na míru těmto telefonům, tabletům a navigacím. Jmenuje se OS (Operating systém) Android. Je založen na platformě Linux. Původně ho vyvíjela samostatná společnost Android Inc., kterou ale v roce 2005 převzala společnost Google společně s platformou Android. Google ji dále rozvíjel, pořádal různé programátorské soutěže (Android Developer Challenge) o nejlepší aplikaci. Programátory motivovala finanční odměna. Bylo založeno konsorcium OHA (Open Handset Alliance), které původně sdružovalo 34 výrobců hardwaru, softwaru a telekomunikačních společností. Dále se značně rozrůstá. Dne 23.září 2008 je oficiálně představen systém Android 1.0. Nejnovější verze k dnešnímu dni je 3.0. Využívá objektově orientovaného programovacího jazyka Java. Tento systém je velice uživatelsky nebo spíše programátorsky přívětivý. O tomto svědčí velice přehledný ADT (Android Development Tool) plugin, který je vydáván zároveň s novými verzemi tohoto systému. Vývojový plugin je psán na míru programovacímu prostředí Eclipse, které je stejně jako plugin dostupné bez jakýchkoli omezení a plně zdarma [1].

Tato práce je zaměřena na vytvoření aplikace, která bude schopna zpracovávat, ukládat a zobrazovat GPS (Global Positioning systém) souřadnice. Tyto budou získávány z příchozích SMS (Short Message Service) zpráv. Bude fungovat jako doplněk libovolného GSM (Global System for Mobile Communications) modulu, který umí určit svoji pozici na Zemi a tu pak odeslat pomocí SMS zprávy na předdefinované mobilní číslo.

Čtenář získá základní vědomosti, jak zprovoznit Android SDK (Software Development Kit) ve vývojovém prostředí Eclipse. Dále jak používat emulátor a testovat svůj kód v něm. Základní povědomí o vnitřní struktuře platformy a jejích principech. Jsou zde také zmíněny rozdíly mezi standardním programováním pro stolní počítače a pro OS Android.

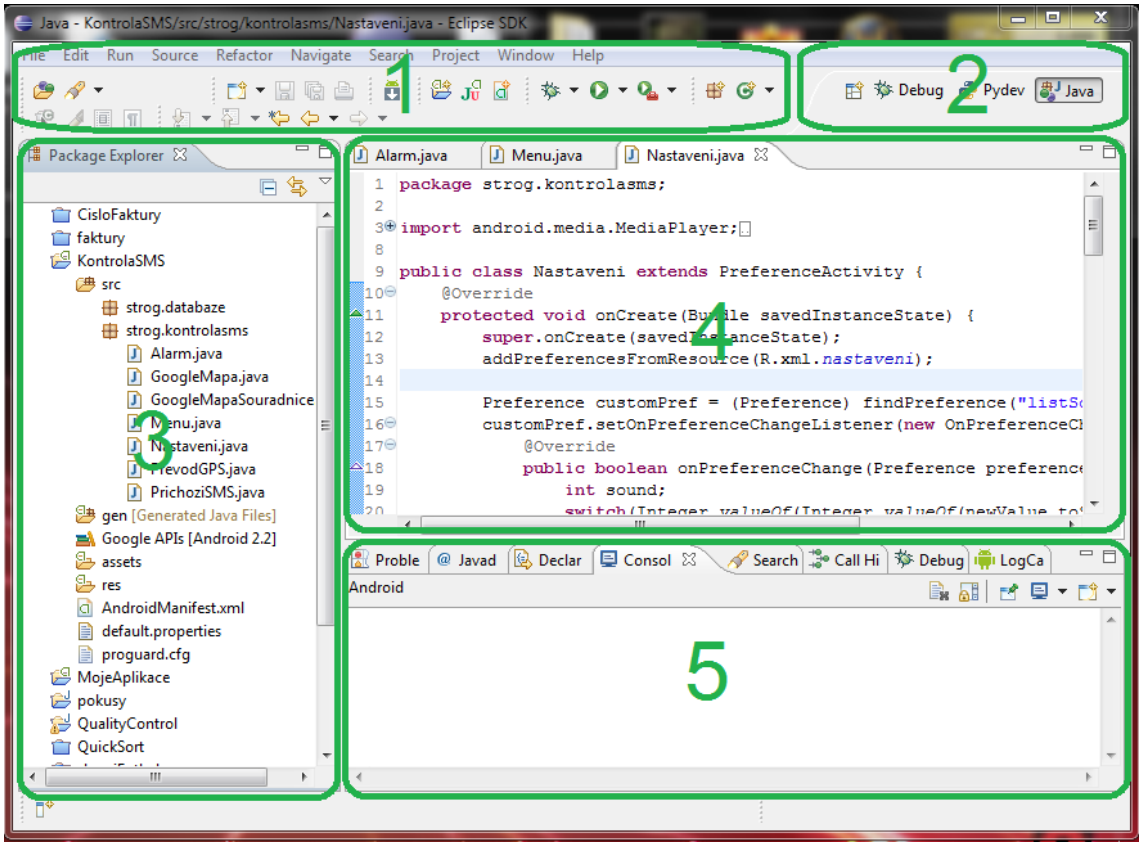
Od čtenáře se očekává základní znalosti programovacího jazyku Java a celkově znalost tématu o vývoji softwaru.

2 Začínáme s OS Android ve vývojovém prostředí Eclipse

Pro vytvoření jednoduché aplikace stačí z internetu zadarmo stáhnout vývojové prostředí Eclipse. Toto lze najít na stránkách http://www.eclipse.org/downloads/. Projekt Eclipse byl původně v roce 2001 vytvořen firmou IBM (International Business Machines Corporation) a byl podporován konsorciem dodavatelů softwaru. Nadace Eclipse byla založena v roce 2004 jako nezávislé neprofitující sdružení, které mělo sloužit a slouží široké komunitě lidí využívající dané prostředí. Toto prostředí je open source. Není ho potřeba instalovat, stačí rozbalit stažený zip soubor. [2] Dále je potřeba stáhnout vývojový balíček Android SDK, tento balíček je také zdarma dostupný na stránkách http://developer.android.com/sdk/. Jako poslední část instalace je potřeba pomoci prostředí Eclipse stáhnout a nechat nainstalovat ADT plugin.

2.1 Seznámení s prostředím a jeho zprovoznění

Na obrázku 1 je vidět základní vzhled okna pro vybranou perspektivu programovacího jazyku Java. V tomto případě je rozdělena do pěti částí. Tyto části je



Obr. 1: Vývojové prostředí Eclipse

možné libovolně umísťovat do hlavního okna nebo i další přidat za cenu toho, že se ostatní zmenší. Pouze část jedna a dvě zůstávají stále na stejném místě.

1) Tato část je vyhrazena pro lištu nabídek a nástrojů, které jsou pevně svázané s prostředím Eclipse. Její sestavu a pořadí nelze měnit. Pro základní použití prostředí jsou nejdůležitější nabídky: File, Run, Window a Help. Pomocí File lze otevřít, ukládat a vytvářet nové projekty. Run umožňuje, mimo jiné, ladit a spouštět kód. Window spravuje a vytváří perspektivy, také obsahuje položku Preferences, ve které se nastavuje cesta k nainstalovaným vývojovým balíčkům a umožňuje tak prostředí pracovat s různými programovacími jazyky. Poslední je tlačítko Help, to umožňuje přístup k rozsáhlé nápovědě a přidávání balíčků do prostředí. Všechny výše zmíněné nabídky a mnoho dalších je možné také používat pomocí zkratek v podobě obrázků, které se nacházejí hned pod lištou nabídek.

2) Část, která umožňuje jednoduše přepínat perspektivy. Perspektiva určuje uspořádáni oken pro daný programovací jazyk nebo práci s kódem. Při psaní kódu je u vzhledu dán důraz na to, aby největší část okna byl editor kódu. Všechna ostatní okna jsou menší a jsou určena pro co nejrychlejší pohyb v různých částech projektu. Naopak pokud je kód potřeba odladit, což se dělá pomocí perspektivy „Debug“, dojde k přerovnání oken a je možné vidět náhled právě používaných proměnných po spuštění programu.

3) Souborový manažer. Rychlé procházení balíčky a částmi projektu, které jsou automaticky generované nebo nabalené v rámci použitého programovacího jazyku. Přehled všech aktuálně vytvářených projektů a správa jejich zobrazení.

4) Editace textu. Při psaní kódu převážně největší a nejvíce využívaná část. Barevný vzhled a struktura textu lze přednastavit. V levé části je vidět číslovaní řádků pro lepší orientaci při hledání chyb, které jsou vypsané do konzole (část 5). Je zde také pomocí levého tlačítka možné vyvolat nabídku a normalizované části kódu, jako jsou konstruktor, getry a setry, automaticky generovat.

5) Poslední pátá část je konzole. Zde se zobrazují chyby při ladění a jejich pozice v kódu. Speciálně pro Android sem lze přidat tzv. „Logcat“. Ten spolupracuje přímo s emulátorem mobilního zařízeni a odchytává jeho veškerou činnost a tu pak zobrazuje v místě, kde se normálně nachází konzole. Jak konzole tak logcat fungují souběžně a jejich zobrazení lze přepínat pomocí oušek v horní části tohoto pod okna.

Po rozbalení na pevný disk a po prvním spuštění ještě aplikace neumí pracovat s Android soubory a s emulátorem. Je potřeba pomocí prostředí doinstalovat výše zmíněný ADT plugin. Dále je potřeba určit umístění Android SDK na disku a tuto cestu také nastavit do Eclipsu. Ten pak bude moci využít lokalizované soubory k překládání kódu.

Nejdříve je tedy potřeba přidat nástrojový balíček pro Android. V nabídce Help → Install New Software se klikne na tlačítko Add. Pole Name se vyplní smyšleným názvem odpovídajícím významu aplikace. Location, čili umístění, kde se nacházejí soubory pro stažení a nainstalovaní, najdete na stránkách http://developer.android.com/sdk/eclipse-adt.html. V prostřední části okna by se měli zobrazit možné balíčky ke stažení. Poté několikerým stisknutím tlačítka Next a potvrzením, že souhlasíte s licencí, se posounete až do chvíle, kdy okno zmizí. Plugin je úspěšně přidán. Teď už jen zbývá nastavit cestu k Android SDK. V nástrojové liště zvolíte Window → Preferences. V levém menu vyberete Android. Pomocí tlačítka Browse určíte umístěný SDK na disku. Tím je proces zprovoznění prostředí u konce.

2.2 Android SDK a práce s emulátorem

Obsahuje systém spravování více emulátorů mobilních zařízení a kompletní sadu nástrojů pro vývoj a ladění programu.

Emulátor je schopen napodobit všechny základní funkce a procesy, které probíhají v mobilním telefonu. Jako je třeba vybíjení baterie, přístup k internetu pomocí 3G (3rd Generation) sítě, nabíjení baterie nebo odchozí/příchozí hovor. Také je možné spustit emulátory dva a realizovat jejich vzájemnou komunikaci. Při vytváření daného emulátoru v manažeru je možné nastavit širokou škálu parametrů. Ať je to rozlišení displaye, velikost interní paměti zařízení, či verze podporovaného operačního systému. Díky tomuto emulátoru nemusíte ani vlastnit reálné zařízení a přitom můžete vyvíjet plnohodnotné aplikace.

Virtuální zařízení je možné spustit zvlášť ještě před vývojovým prostředím. Toto ale není nutné, jelikož naše prostředí obsahuje zásuvný modul, který samostatně vybere vyhovující, již vytvořené zařízení a dojde k jeho spuštění. Pokud nemáme vytvořené zařízení, které vyhovuje verzi napsaného kódu, spustí se místo emulátoru manažer.

Obr. 2: Emulátor

2.3 Založení nového projektu

V nabídce File → New → Other se zvolí Android projekt. Objeví se okno, kde je potřeba vyplnit pár nezbytných informací pro založení nového projektu.

Jako první je název, ten nesmí obsahovat písmena s interpunkcí jinak může mít libovolný tvar. Dále je potřeba zaškrtnout verzi operačního systému pro kterou se bude psát aplikace. V rámci práce to bylo 2.2, jelikož pro telefon na kterém probíhali testy aplikace je jako nejnovější uvolněná právě tato verze s kódovým označením FroYo. Emulátor, který je již vytvořený manažeru, nebo bude teprve vytvořen, samozřejmě musí splňovat daný API level, či ho převyšovat. API level udává jednoduchý číselný ekvivalent k verzi OS. Například verze 2.2 odpovídá APL level 8.

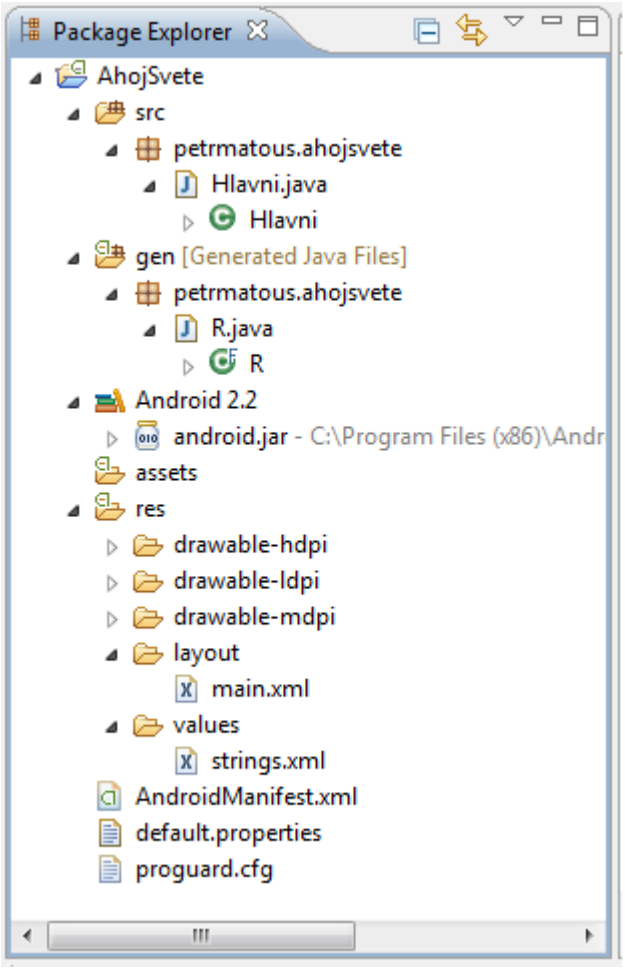
Ve spodní části okna je na závěr potřeba pojmenovat tři základní vlastnosti projektu. Název aplikace, který se bude zobrazovat například na spouštěcí ikonce po nainstalování do mobilního zařízení. Může obsahovat znaky s interpunkcí. Název balíčku musí obsahovat alespoň dva identifikátory. Ty se oddělují tečkou. Je zvykem psát názvy balíčků malými písmeny. Kde do první části identifikátoru většinou zařadíte své jméno či přezdívku a do druhé pak název vystihující, co balíček obsahuje. Například „petrmatous.ahojsvete“. Toto je dobré, pokud se aplikace vyvíjí ve větší skupině lidí tvořících si své vlastní balíčky. Nedochází pak ke kolizi jejich názvů při sloučení do

společného projektu.

Poslední vlastností je název vytvořené aktivity. Toto pole představuje název jediné třídy, která se vytvoří do vašeho balíčku. Zde je pro změnu zvykem název třídy začínat velkým písmenem. Pokud některé z předchozích polí nevyplníte, či vyplníte špatně Eclipse vás nenechá projekt založit. Pokud vše zvládnete správně, budou předgenerovány soubory pro správnou funkčnost.

2.4 Struktura souborů v projektu

Na obrázku 3 je vidět výřez souborového manažeru, kde jsou zobrazeny všechny soubory projektu, které se vygenerovaly při jeho založení.



Obr. 3: Souborový manažer

- src – Třídy, které vytvoří programátor a které představují jádro psaného kódu. Je zde vidět korektně pojmenovaný balíček a v něm jedna třída.
- gen – Třída s názvem R. Tato třída je automaticky generovaná pomoci IDE a nikdy by se neměla ručně editovat. Obsahuje pojmenování hexadecimálních

adres všech prvků, které jsou umístěny na grafickou plochu naší aplikace. Například tlačítka, textová pole, obrázky, zvuky a další.

- Android 2.2 – Sada všech Android, Java, Apache a Dalvik knihoven, které jsou obsažené v dané verzi a potřebné pro běh aplikace.
- assets – Přídavné soubory, které mohou být přibaleny k výsledné aplikaci.
- res – Zkratka pro slovo resources (zdroje). Obsahuje šablony vzhledů aktivit (obrazovek), které se budou uživateli zobrazovat. Dále pak všechny obrázky, melodie a přednastavené hodnoty.
- AndroidManifest.xml – Specifikační soubor, který určuje, co všechno může daná aplikace provádět a jak je strukturovaná. Více viz kapitola 3.3 .
- proguard.cfg – Konfigurační soubor, který definuje, jak upravit vaši aplikaci, pokud ji chcete uvolnit pro ostatní uživatele a nechcete, aby ji byli schopni zpětně zrekonstruovat a použít její části ve svůj prospěch. V základním nastavení, které je automaticky generováno, dojde k pokrytí pouze základních tříd. Pro zvýšení bezpečnosti se doporučuje pomocí nástroje ProGuard upravit tento soubor tak, aby přesně vyhovoval vaší aplikaci [3].

3 Základní vlastnosti OS Android

Pro psaní aplikací se využívá programovací jazyk Java. Ten je objektově orientovaný a umožňuje velice pohodlnou a rychlou práci. Android využívá principu minimálních privilegií. Celý projekt, po zkompilování pomocí Android SDK nástrojů, vytvoří pouze jeden balíček, který obsahuje veškerý napsaný kód knihovny jazyku Java a také všechny datové soubory přiložené k projektu. Přípona takto vytvořeného balíčku je *.apk (Android Package)*. Tento balíček je zároveň instalační. Do zařízení podporujícího OS Android je možné ho zkopírovat přes datový kabel, pomocí bluetooth, WiFi (Wireless Fidelity), či stáhnout z Android marketu. Po instalaci každá aplikace žije ve svém vlastním sandboxu [4].

3.1 Architektura

Jádro Androidu je postaveno na principu systému Linux. To nám umožňuje běh více aplikací najednou. Množství aplikací bežících zároveň je limitované pouze velikostí paměti daného zařízení. Pokud je potřeba spustit aplikaci, která by přesáhla povolený limit, dojde k ukončení té aplikace, která je nejdelší dobu bez využití [4].

Základní rozdělení do pěti vrstev: Applications, Application Framework, Libreries, Android Runtime, Linux Kernel [5].

3.1.1 Applications

Základní sada aplikací, kterou obsahuje čerstvě nainstalovaný operační systém. Obsahuje aplikace jako jsou práce s SMS zprávami, seznam kontaktů, realizace odchozího a příchozího hovoru, kalendář, přístup ke Google mapám, internetový prohlížeč, mailový klient a další.

3.1.2 Application Framework

Zde se nacházejí základní programové rutiny, které výrazně zjednoduší psaní kódu. Místo toho, aby v kódu byly věci řešeny složitě a bylo potřeba znát úplnou základní strukturu a metodiku všech operací, stačí využít některého z dříve vytvořených manažerů. To je například manažér pro upozornění na události, zahájení odchozího hovoru, přístup k datovým zdrojům telefonu nebo zobrazování aktivit

a přechody mezi nimi. Díky těmto prostředkům je možné vytvářet aplikace na úrovni, kterou mají samotní programátoři jádra. Vyvolávat upozornění ve status baru, vibrovat s telefonem, měnit telefonní kontakty, zobrazovat mnoho grafických prvků v naší aplikaci. Například: tlačítka, textová pole, listy, vestavěný webový prohlížeč. Ty pak lze zarovnat do úhledné mřížky a to vše bez větší námahy.

3.1.3 Libraries

Sada knihoven, která je používána OS Android. Základem jsou knihovny jazyka C/C++ upraveny pro potřebu vestavných zařízení na bázi Linuxu. Tyto knihovny jsou programátorovi zpřístupněny pomoci frameworku (kapitola 3.1.2). Dále jsou to pak knihovny pro zpracování obrazu a zvuku, které obsahují jedny z populárních formátů (MPEG4, MP3, AAC, JPG a PNG) nebo SQLite (Structured Query Language Lite) knihovna pro práci z databázemi. V neposlední řadě je to 3D (3 Dimensions) knihovna, která je implementována na základě OpenGL ES (Open Graphics Library for Embeded Systems) 1.0 API. Umožňuje 3D akceleraci a vysoce optimalizované softwarové rastrování.

3.1.4 Android Runtime

Zde běží virtuální stroj Dalvik. Funguje jako prostředník mezi hardwarem a softwarem. Díky tomu je možné provozovat stejné aplikace na různých architekturách. Pomoci nástroje s názvem *dx*, který je obsažen v Android SDK se zkompiluje kód v podobě tříd Javy do podoby jiných tříd, které jsou formátu *.dex*. Všechny třídy v podobě toho formátu jsou pak uloženy do již výše zmíněného jediného souboru s příponou *.apk*. Tento soubor se spustí ve virtuálním stroji [6].

Každá spuštěná aplikace má svou instanci virtualizovanou v tomto zařízeni. To umožňuje mnohem lépe hlídat bezpečnost.

3.1.5 Linux Kernel

Obsahuje ovladače pro display, kameru, klávesnici, WiFi, zvuk, správu napájení, správu flash paměti a další. Umožňuje k nim přístup a liší se podle zařízení.

3.2 Základní komponenty

Základní stavební kameny celé aplikace, které jsou čtyři. Activites (aktivity), Services (služby), Content Providers (poskytovatelé obsahu), Broadcast recievers (přijímače). Každý „kámen“ umožňuje systému rozdílný způsob přístupu k aplikaci. Ne všechny komponenty jsou reálnými body vstupu, v tomto ohledu závisí jedna na druhé. Každá jedna je jedinečný stavební blok a pomáhá tak definovat celou aplikaci.

- **Aktivita**

  Každá aktivita představuje samostatný vzhled stránky, se kterou může uživatel nějakým způsobem pracovat. Má k sobě přiřazený soubor s rozestavěním prvků, který je umístěn v res → layout (viz obrázek 3). Aktivitu lze považovat za malou podaplikaci. Celý projekt může být tvořen z více aktivit. Například jednoduchý poznámkový blok. Bude obsahovat aktivitu, ve které bude seznam poznámek, dále bude obsahovat okno pro vkládání nové poznámky. Každá aktivita může existovat sama o osobě nezávisle na druhé a dohromady tvoří aplikaci. Pokud je to dovoleno, je možné jednotlivé aktivity využívat i v úplně jiných aplikacích, než pro které jsou určeny. Například pokud je potřeba využít fotoaparát, předpokládáme, že v zařízení již existuje aktivita, která toto umožňuje, a nám stačí pouze zavolat požadavek na systém (tzv. intent) a ten nám dá na výběr všechny aplikace, které odpovídají našemu volání. My si jednu vybereme. Z této aktivity se nám pak vrátí pořízena fotografie. Pro uživatele to vypadá, jakoby fotoaparát byl součástí naší aplikace [7].

- **Služby**

  Jsou to procesy, které běží dlouhodobě na pozadí systému, a jsou nečinné až do momentu, než nastane jejich chvíle. Nemají žádné uživatelské rozhraní. Příkladem služby může být přehráváni hudby na pozadí telefonu, přičemž je umožněno vykonávat úplně jinou činnost, třeba surfování po internetu. Dalším příkladem je nastavený alarm na určitý čas a datum. Nebo přenos dat v rámci aplikace, aniž by blokoval její další používání [8].

- **Poskytovatelé obsahu**

  Jak již název napovídá, tento prvek bude sloužit k poskytováni přístupu k datům. Umožní tak sdílení dat mezi aplikacemi. Dokonce i to, že i aplikace, která data

nevytvořila, je může změnit, či dokonce smazat. Vše zase závisí na tom, jaká práva jsou nastavena pro ostatní aplikace. Příkladem je třeba seznam telefonních kontaktů, který přestože je již implementován jako aplikace v samotném jádru, umožňuje ke kontaktům přidávat informace nebo je upravovat [9].

- **Přijímače**

  U tohoto prvku jde o zachytávání systémových i nesystémových volání (již zmíněné intenty).

  Pokud nastane v zařízení nějaká eventualita, například je indikováno vybití baterie, vypnutí displaye, příchod nové SMS zprávy, snaha otevřít nějaký dokument, či webovou stránku, dojde k vyslání systémového volání, které to oznamuje. Na tento signál mohou okamžitě reagovat všechny aplikace, které obsahují přijímač odpovídající tomuto volání. Přijímače samy o sobě nezahajují aktivitu, ale využívají k tomu například manažer upozornění [10].

3.3 Android manifest

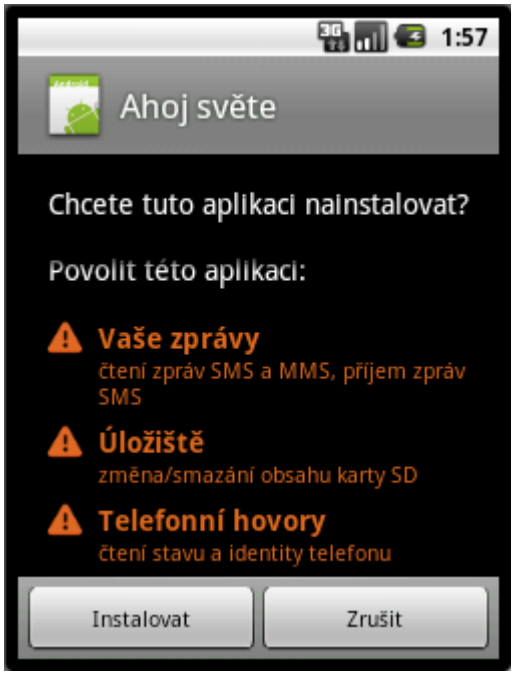
Soubor *AndroidManifet.xml*, který je obsažen bez vyjímky ve všech vytvořených aplikacích, definuje bezpečnostní rámec. Určuje tak, co všechno je a není možné, kam

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="petrmatous.ahojsvete"
      android:versionCode="1"
      android:versionName="1.0">
<uses-permission android:name="android.permission.READ_SMS"></uses-permission>
<uses-permission android:name="android.permission.BROADCAST_SMS"></uses-permission>
<uses-permission android:name="android.permission.RECEIVE_SMS"></uses-permission>
<uses-permission android:name="android.permission.SEND_SMS"></uses-permission>
<uses-permission android:name="android.permission.CALL_PHONE"></uses-permission>


    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".Hlavni"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name="SMS">
            <intent-filter>
                <action android:name="android.provider.Telephony.SMS_RECEIVED"></action>
            </intent-filter>
        </receiver>

    </application>
</manifest>
```

Obr. 4: Ukázka manifest souboru

má aplikace přístup a kam ne. Tyto údaje se při instalaci převezmou do systému a aplikaci se přidělí identifikační číslo, které zná jenom systém. Identifikační číslo je úzce svázáno s povoleními. Toto je krásná ukázka využití systému minimálních privilegií. Aplikace v základu nemůže přistoupit k žádným systémovým voláním ani datům. Nemůže po systému nic požadovat. Vše , co je potřeba provést pomocí aplikace, je nutné nadefinovat [4].

Na obrázku číslo 4 je vidět ukázka manifestu. Pokud ho budete postupně procházet jako když je čtena kniha, tak na prvnímu řádku je norma textu *utf-8*. Dále pak, že je určen pro balíček *petrmatous.ahojsvete*. Poté přichází sada pěti povolení. Tato aplikace tedy bude moci číst, přijímat a odesílat SMS zprávy. Zachytávat hromadný broadcast v situaci, kdy přijde nová SMS zpráva. Poslední povolení se týká možnosti zahájit hovor. Tím ale síla manifest souboru nekončí. Aplikace má povoleno poslouchat, zda přišla nová SMS zpráva, ale nemá definováno, která třída má tento broadcast zpracovat, a proto se nabídnou řádky 21 až 25. Zde je definován interní filtr. Díky němu má třída SMS povoleno zpracovávat příchozí SMS zprávu. Všechny třídy, které jsou v projektu vytvořeny, musí být definovány jinak, jako by neexistovaly.

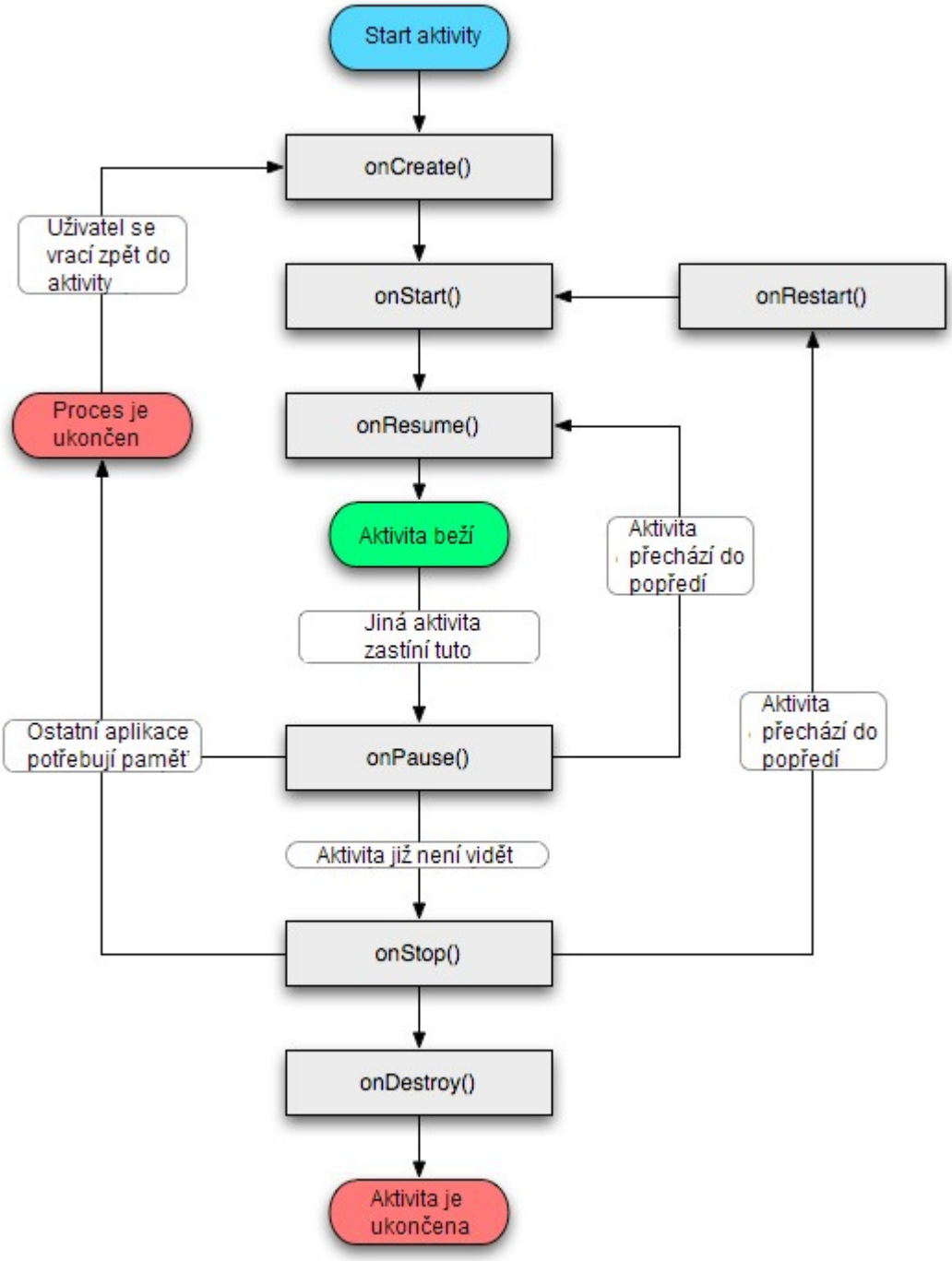


Obr. 5: Instalace aplikace

Všechny údaje obsažené v manifest souboru se zobrazí v přehledné tabulce při instalaci aplikace na mobilní telefon. Uživatel má tedy přehled o tom, co instalovaná aplikace bude dělat a kam má přístup. Pokud s něčím nesouhlasí, odmítne instalaci.

3.4 Životní cyklus aktivity

Každá aktivita se může nacházet v několika stavech. Tyto stavy jsou vyjádřeny ve třídě s aktivitou jako metody s následujícími názvy. Samotný cyklus aktivity je znázorněn v obrázku 6.



Obr. 6: Životní cyklus aktivity [7]

- **onCreate()** – Základní metoda, která je volána po prvním spuštění aktivity. Při provádění této metody se aktivita nenachází nikde v zásobníku aktivit. Využívá se k nastavení zobrazení na display, nastavení statických proměnných, přiřazení dat do listů a tak podobně. Až do chvíle, než je aktivita úplně zničena z důvodu uvolnění paměti, či ukončena uživatelem, tato metoda se znovu neprovádí.
- **onStart()** – Tato metoda následuje ihned po metodě onCreate(). Je možné se do ni dostat i po metodě onRestart().
- **onResume()** – V této části očekáváme data z jiných aktivit. Tato metoda nastane ve chvíli, kdy naše aktivita nebyla odstraněna ze zásobníku aktivit. Předchází ji metoda onPause().
- **onPause()** – Tato metoda je běžně využívána jako poslední chvíle, kdy je potřeba uložit data předtím, než se aktivita ukončí. Je také dobré ukončit všechny animace a podobné věci, které nejsou potřebné, když není aktivita v popředí, a zároveň jsou velikou zátěží pro procesor.
- **onStop()** – Volá se ve chvíli, kdy už aktivita není vidět. Po této metodě muže nastat buď onDestroy() nebo onRestart().
- **onDestroy()** – Volá se ve chvíli, kdy je aktivita ukončena.

Z výše zmíněných metod je povinnost mít implementováno pouze onCreate(), zbytek je možné vynechat. Bez této metody by se na display nic nezobrazilo a ani nic nestalo. Díky těmto metodám jsme plně schopni kontrolovat základní tři životní cykly.

Celkový životní cyklus, který probíhá mezi voláním metod onCreate() a onDestroy(), kde se tedy při volání onCreate() nastaví statické proměnné a při onDestroy() se uvolní poslední zbytky zdrojů.

Viditelný životní cyklus, který probíhá mezi voláním metod onStart() a onStop(). Vše, co se děje mezi těmito dvěma metodami, může uživatel vidět na display a reagovat na to.

Životní cyklus v popředí, ten probíhá mezi voláním metod onResume() a onPause(). V průběhu tohoto cyklu je aplikace před všemi ostatními aplikacemi a je zároveň zobrazena na obrazovce. Aktivita může pravidelně přecházet z a do popředí.

Například metoda onPause() je volána ve chvíli, kdy zařízení přechází do režimu spánku nebo se zobrazí dialogové okno. Vzhledem k tomu, že přechody mohou nastávat často, je dobré do metod onPause() a onResume() psát co nejméně kódu. Tím je zajištěno minimální zpomalení chodu celého zařízení.

3.5 Rozdíly mezi programováním pro OS Android a pro stolní PC

Odlišnosti mezi psaním v jazyce Java pro OS Android a například C# (C sharp), který je v dnešní době hojně využíván pro psaní aplikací pod OS Windows a je nástupce zaběhlého jazyku C/C++, vycházejí z vlastností obsažených v minulých kapitolách a ze samotného chování jazyka Java a jeho běhu na virtuálním stroji. Za nejzásadnější by mohlo být považováno to, že struktura projektu pro OS Android již od začátku vede k bezpečnosti. Pomocí Android manifest souboru uživatel přesně ví, co jeho aplikace bude mít možnost provádět. Toto je doplněno tím, že každá aplikace běží jako samostatný virtuální proces v Sandboxu. Ten pak může striktně ohlídat to, co je aplikací povoleno či nikoliv. Dalším rozdílem je konzole. Tam, kde je programátor zvyklý hledat výpis chyb při běhu programu, najdeme pouze, že aplikaci se úspěšně podařilo nahrát do emulátoru. Při další prací s emulátorem se již obsah konzole nemění. Naštěstí je zde v rámci pluginu tzv. Logcat, který je spojený s emulátorem a zaznamenává veškeré jeho operace. Místo příkazu *Console.println(„Obsah chybové hlášky“)* v jazyce *C# nebo System.out.println(„Obsah chybové hlášky“)* v jazyce Java je zde potřeba používat příkaz *Log.d(„Název chybové hlášky“, „Obsah chybové hlášky“)*. Ten je pak přebírán z emulátoru do logcatu a zobrazován mezi ostatními údaji načtenými z emulátoru.

4 Aplikace pro sledováni GSM modulu

Aplikace je schopna sledovat pouze jeden GSM modul. Identifikátorem modulu pro aplikaci je jeho telefonní číslo. Pro představu příkladem takovéhoto modulu může být například namátkou vybraný DAVISCOMMS EaziTRAC1000, který dle specifikací výrobce dokáže odesílat až dvacet druhů SMS zpráv včetně GPS souřadnic na definované číslo [11].

Obr. 7: Modul DAVISCOMMS EaziTRAC1000 [11]

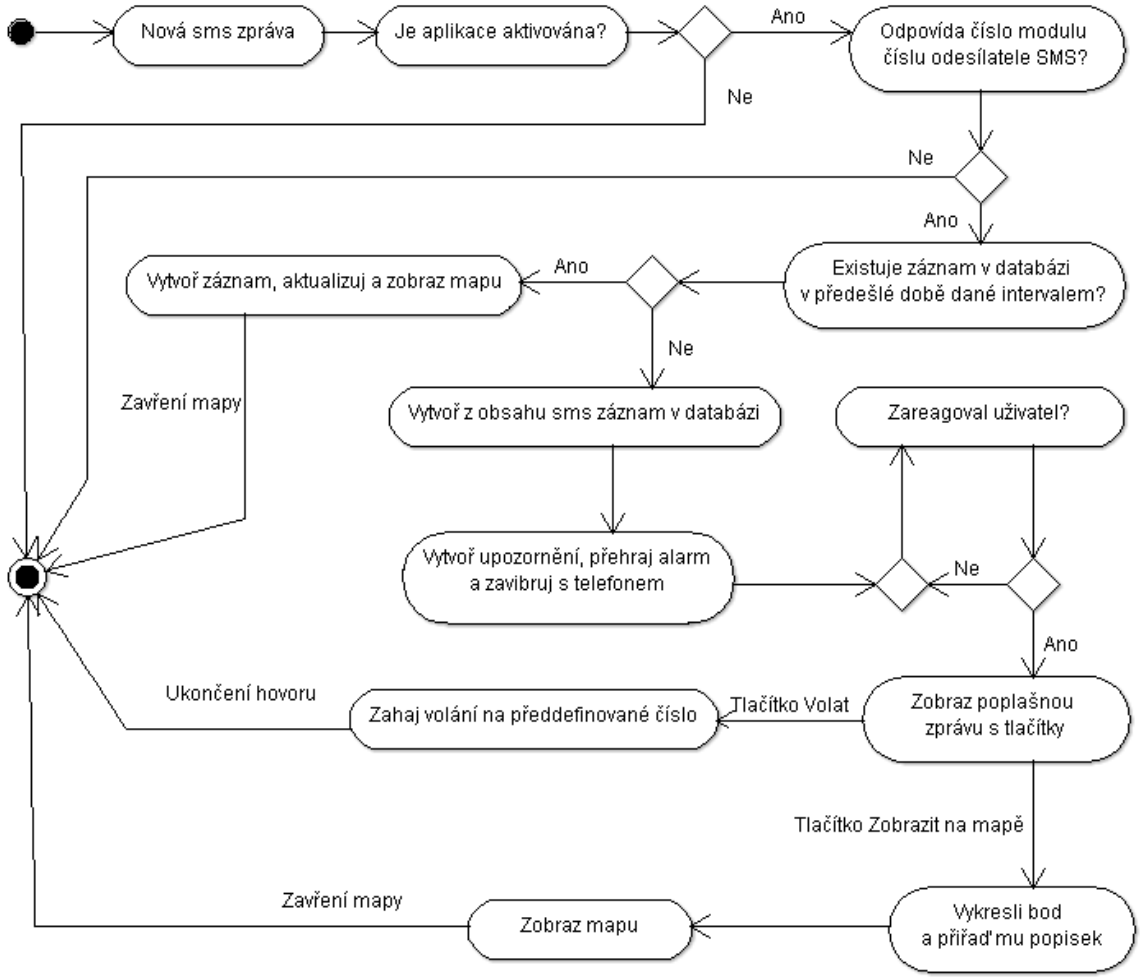
Další podstatnou hodnotou je předpokládaný interval mezi aktivacemi modulu. V obsahu SMS zpráv je pouze údaj o zeměpisné šířce a délce. Z hlavičky zprávy se odečítá ještě datum a čas, kdy byla zpráva přijata. Aby šlo rozeznat, zda se jedná o aktuální sadu souřadnic, která nepatří například k té před měsícem, využijeme právě hodnotu intervalu. Ta je udávána v hodinách. Minimální možná hodnota, kterou lze zadat je jedna hodina. Doporučená hodnota je 12 hodinová a maximální je neomezená. Zprávy jsou za sebe v databázi ukládány chronologicky od té s nejstarším datem po nejnovější. Toto je dáno tím, že je ani jinak přijmout do mobilního telefonu nemůžeme. Do databáze nelze uživatelsky zasahovat. Je možné pouze zobrazit si historii poplachů a z té zobrazovat staré trasy na mapě.

Nastavit lze také hlášku, která se zobrazí na display a uvede nás okamžitě do situace. K této hlášce lze nadefinovat tlačítko, kterým se zahájí hovor na telefonní číslo. Výchozí hodnota je 158, což odpovídá nejbližší krajské pobočce státní policie České republiky.

Ke zobrazení trasy do mapy jsou využity mapové podklady Google. K těmto je přístup pouze pokud jsme připojeni k internetu.

4.1 Běh aplikace

Na následujícím diagramu je zjednodušeně znázorněna základní rutina aplikace. Nezohledňuje všechny možné situace, které mohou nastat.



Obr. 8: Diagram aplikace

Průchod diagramem začíná na pozici plného černého bodu a končí v černém bodu s bílým okrajem. Po přijetí nové SMS zprávy přichází první rozhodovací podmínka, zda je aplikace aktivována. Po instalaci je aplikace vždy nastavena do aktivní naslouchající pozice. Uživatel ji může v nastavení deaktivovat, všechny SMS zprávy budou tak zahozeny. To je tedy větev Ne. Pro větev Ano je tu další podmínka: kontrola čísla modulu. Program zde buď zase skončí nebo pokračuje dále. Pokud existuje již záznam v databázi, který je dostupný v rámci přednastaveného intervalu, aplikace se zachová jinak, než když příchozí zprávu vyhodnotí jako zahajovací pro nové sledování. V případě, že jde o již probíhající sledování pouze se doplní nový bod do mapy a dogeneruje se k němu cesta. Tím tato větev končí. V opačném případě je potřeba

vytvořit náležitou notifikaci, která upozorní uživatele. Pokud okamžitě nezareaguje na vibrování a zvonění, nic se neděje. Aplikace se zde ukončí a čeká se na reakci uživatele. Toto čekání není realizováno přesně podle diagramu. Systém pak nadále hlídá reakci uživatele na notifikaci. Po této reakci je opět spuštěna aplikace. Na výběr jsou dvě možnosti jak pokračovat. Buď zahájit hovor na předdefinované číslo nebo zobrazit získanou souřadnici na mapě. Po ukončení hovoru je samozřejmě také možné zobrazit mapu.

4.2 Základní třídy aplikace a jejich náhled

Aplikace je rozdělena do dvou balíčků. Balíčky s názvem *databaze* a *gpslokalizator.* Databáze obsahuje pouze jednu třídu a tou je ovladač obsahující sadu metod pro práci s SQL databázi, více viz kapitola 4.5 . U tohoto balíčku se předpokládá znovupoužití u dalších aplikací, proto je oddělen od zbytku tříd. V druhém balíčku jsou obsaženy třídy specifické pro tuto aplikaci. Zde je jejich krátký popis:

- *Alarm.java* – Tato třída zobrazuje varovnou obrazovku s předdefinovanou hláškou a dvěma tlačítky. První tlačítko umožňuje zahájit hovor na definované číslo. Druhé pak zobrazit souřadnici na mapě.
- *GoogleMapa.java* – Hlavní třída pro vykreslení dat do mapy. Pomocí intentu do ní vstupují dva parametry. Tyto parametry udávají hranice souřadnic, které se mají vykreslit do mapy a má se mezi nimi vytvořit cesta. Pokud jsou hraniční souřadnice stejné, zobrazí se pouze jeden bod. Z těchto bodů se pak sestaví trasa. Aby trasa kopírovala tvar silnice, je využito webové adresy http://maps.google.com/maps? Z této adresy lze pomocí úpravy odkazu do specifického tvaru získat nazpět KML (Keyhole Markup Language) soubor. V tomto souboru je pak sada souřadnic, která přesně definuje průběh trasy mezi hraničními souřadnicemi. Tento soubor se bohužel musí generovat pro každou dvojici souřadnic. Odkaz nelze použít pro více než dvě lokace zároveň. Zde je příklad odkazu pro dvě souřadnice: http://maps.google.com/maps?f=d&hl=en&saddr=50,772670&daddr=15,072640&ie=UTF8&0&om=0&output=kml

```
77          // Projde všechna data v kurzoru, vytvoří z nich trasu a přidá jim
78          // popisek
79          while (c.moveToNext()) {
80              point1 = point2;
81              ts.setTime(Long.parseLong(c.getString(0)));
82              point2 = new GeoPoint(Integer.parseInt(c.getString(1)),
83                      Integer.parseInt(c.getString(2)));
84              gmSouradnice.addOverlay(new OverlayItem(point2, ts.toString(),
85                      PrevodGPS.intToGPS(Integer.parseInt(c.getString(1)), true)
86                              + ", "+ PrevodGPS.intToGPS(
87                                      Integer.parseInt(c.getString(2)), false)));
88
89              String pairs[] = getDirectionData(
90                      ((double) point1.getLatitudeE6()) / (double) 1000000 + ","
91                              + ((double) point1.getLongitudeE6())
92                              / (double) 1000000,
93                      ((double) point2.getLatitudeE6()) / (double) 1000000 + ","
94                              + ((double) point2.getLongitudeE6())
95                              / (double) 1000000);
96
97              gp2 = point1;
98              for (int i = 1; i < pairs.length; i++) {
99                  lngLat = pairs[i].split(",");
100                 gp1 = gp2;
101
102                 gp2 = new GeoPoint((int) (Double.parseDouble(lngLat[1]) * 1E6),
103                         (int) (Double.parseDouble(lngLat[0]) * 1E6));
104                 mapOverlays.add(new GoogleMapaTrasa(gp1, gp2));
105             }
106         }
```

Obr. 9: Ukázka kódu – tvorba trasy

- *GoogleMapaSouradnice.java* – Pomocná třída, která slouží jako stavební blok pro třídu GoogleMapa. Rozšiřuje Androidem definovanou třídu ItemizeOverlay. Taro třída umožňuje na mapě zobrazovat objekty, kterým lze přiřadit určité vlastnosti.
- *GoogleMapaTrasa.java* – Také pomocná třída třídy GoogleMapa. V této třídě se z převzatých souřadnic z KML souboru vytvoří křivka, která se vloží do mapy.
- *Historie.java* – Třída zobrazující generický list ze všech souřadnic v databázi.
- *Hlavni.java* – Základní třída, která se spustí jako první po instalaci. Obsahuje dvě tlačítka. Tlačítko historii, které nás přepne do zobrazení třídy *Historie*. A druhé tlačítko nastavení, které umožňuje upravovat hodnoty ve struktuře Preferences viz kapitola 4.3 .
- *Nastaveni.java* – Třída, která obstarává změny hodnot v preferences a jejich zobrazení v popisu hodnoty. Jsou zde definované různé reakce na stisknutí tlačítek v nastavení. Například pří výběru zvuku alarmu dojde k jeho přehrání.
- *PrevodGPS.java* – Pomocná třída pro převod mezi souřadnicemi. Rozeznáváme

tři základní tvary souřadnic. Tvar stupně, minuty, sekundy a desetiny sekund. Toto je nejběžnější tvar. Například: 50°46'21,614"N, 15°4'21,505"E. Přesná poloha je vždy tvořena dvojicí souřadnic. Pak je to tvar desetinný a z něj vycházející desítkový. Oba tyto tvary jsou mnohem lépe zpracovatelné pomocí programu než předešlé. Z tvaru 50°46'21,614"N získáme desetinný tvar jednoduše pomoci násobení: 50 + 46 * 1/60 + 21,614 * 1/3600, logika je stejná jako při převodu času.

- *PrichoziSMS.java* – Tato třída je stěžejní. Realizuje zachytávání a kontrolu příchozích SMS zpráv. Porovnává je s nastavením a vyhodnocuje, zda již probíhá sledování, či je potřeba uživatele upozornit pomocí zvuku a vibrací. Rozšiřuje třídu BroadCast.

4.3 Preferences

Velice užitečná předprogramovaná struktura pro ukládání dat. Umožňuje uložit dvojice dat, kde první je klíč, pomocí kterého přistupujeme k hodnotě a druhá je samotná hodnota. Podle definice na stránkách Android Developers [12] je možné do této dvojice ukládat všechna primitivní data jako jsou: boolean, float, int, long, a string a takto k nim i přistupovat pomocí metod *getString, getLong, getFloat* a tak dále. Toto se bohužel ukázalo jako nepravda. Ostatní metody než *getString* vracejí také string a proto je potřeba všechny hodnoty jiné než string přetypovat. I přes tuto, jako jednu z mála nefungujících věcí, jsou preferences velice užitečné pro ukládání jednoduchých dat, u kterých potřebujeme, aby přetrvala mezi jednotlivými spuštěními aplikace.

4.4 Google API

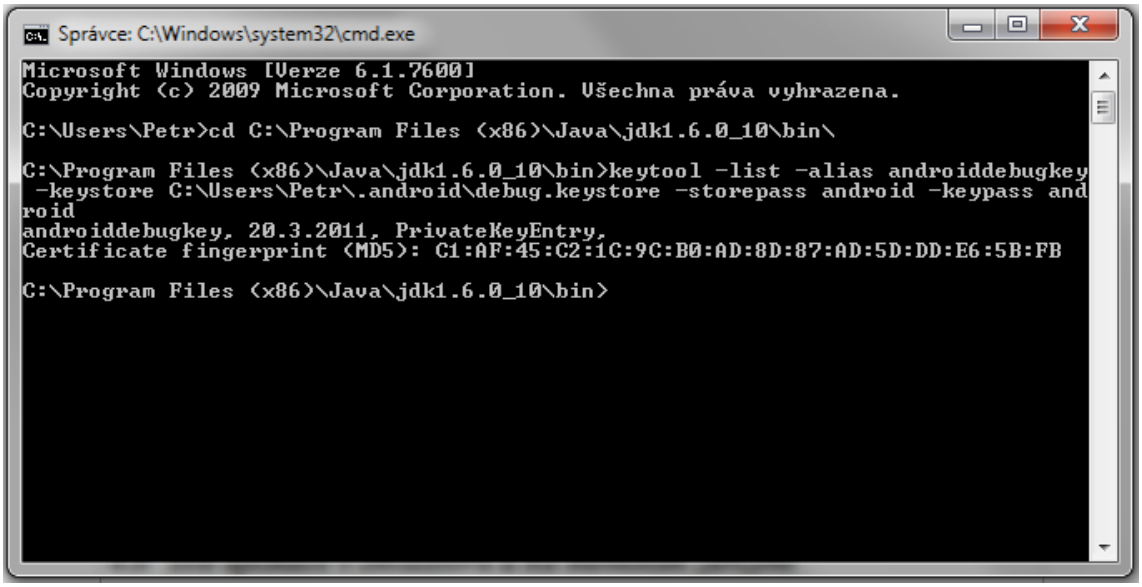
Pro práci s mapou je potřeba splnit pár podmínek. Nejdříve je nutné pomocí AVD Manažeru stáhnout Google API balíčky, nebo pouze jeden balíček, pokud přesně víte pro jakou API verzi budete vyvíjet. Tyto balíčky obsahují třídy pro veškerou práci s mapami. Mapy jsou pouze v online verzi. Je tedy nutné mít po dobu práce s mapou připojení k internetu. Zobrazení mapy jako takové na display není úplně volné. Do aplikace se musí zakomponovat vygenerovaný kód, který ji opravňuje k využíváni mapy. Tento kód lze získat pomocí generátoru na www stránce http://code.google.com/android/maps-api-signup.html. Do tohoto generátoru je potřeba vložit otisk certifikátu. Každá aplikace musí být podepsána certifikátem. Tento certifikát

je možné získat dvojím způsobem. První, pokud chcete distribuovat aplikaci na Android market. Potřebujete vlastní certifikát. Nepotřebujete žádnou certifikační autoritu, stačí vám vlastní privátní klíč, který si sami podepíšete. Vznikne takzvaný self signed certificate. V druhém případě pro ladící a testovací účely postačí certifikát, který se vytvoří uvnitř emulátoru při jeho instalaci. Další postup je již společný. Využije se nástroj *keytool.exe*, který je například obsažen v Java development kitu nebo nějaký externí program. Pomocí tohoto nástroje se získá otisk certifikátu. Umístění certifikátu na disku je pro různé systémy různé [13]. Zde jsou tři nejpoužívanější:

- Windows Vista: C:\Users\<user>\.android\debug.keystore
- Windows XP: C:\Documents and Settings\<user>\.android\debug.keystore
- OS X and Linux: ~/.android/debug.keystore

Dále se použije příkazový řádek, ve kterém spustí nástroj keytool s následujícími parametry:

keytool -list -alias androiddebugkey -keystore <path_to_debug_keystore>.keystore -storepass android -keypass android



Obr. 10: Příkazová řádka – získaní otisku

Získaný MD5 otisk se vloží do pole na výše zmíněné adrese a vygeneruje se kód, který se vloží dovnitř layout souboru určeného pro aktivitu, která používá mapu.

4.5 SQL databáze

V Androidu je implementováno využití SQLite databáze. Tato databáze je velmi populární a je použita ve více aplikacích, než jsme schopni spočítat. Je šířena pod licencí public domain (volné dílo). Není tedy nijak omezena jak pro soukromé tak komerční účely. Na rozdíl od plnohodnotné SQL databáze tato nemá serverovou část. Data ukládá přímo do souborů na paměťové zařízení, na kterém se nachází [14].

Struktura databáze použité v Aplikaci pro sledování je velice jednoduchá. Stačí nám uchovávat pouze čtyři údaje:

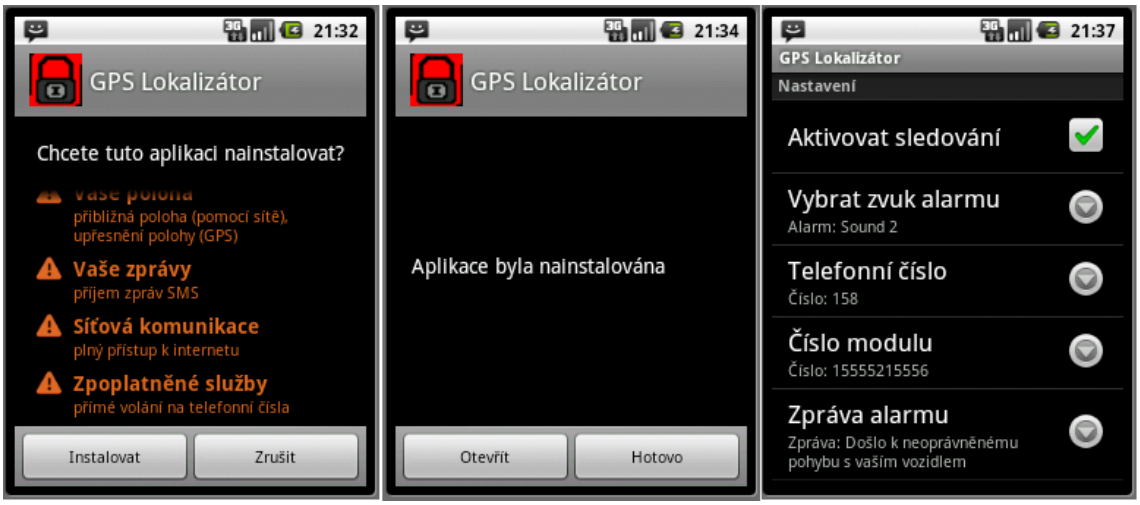
- ID – Jeho hodnota se automaticky zvětšuje po jedné od nuly. Je zde pouze pro lepší práci s daty. Hodnota ve sloupci TimeStamp roste nelineárně a proto je nepraktická pro operace nad jednotlivými řádky databáze.
- TimeStamp – udává hodnotu v milisekundách, která uběhla od data 1. leden 1970, 00:00:00.000 GMT (Greenwich Mean Time), hodnotu lze získat z právě příchozí SMS zprávy příkazem *getTimestampMillis()*. Díky předpokladu, že není reálné obdržet dvě SMS zprávy obsahující souřadnice ve stejnou dobu, by bylo možné sloupec TimeStamp využít jako primární klíč.
- Latitude, Longtitude – zeměpisná šířka a délka: zbylé dva údaje, které se získají z těla zprávy a slouží k přesnému určení pozice na zemském povrchu. Mohou nabývat buď kladných nebo záporných hodnot a mohou se opakovat.

Pro samotnou práci s databází je v aplikaci vytvořena třída *Databaze.java*, která je rozšířena o *SQLiteOpenHelper*. Díky tomuto rozšíření je možné pohodlně používat obdobu standardních manipulačních SQL příkazů jako jsou:

- UPDATE – změna údaje v databázi
- INSERT – vložení údaje do databáze
- DELETE – smazáni záznamu
- SELECT – výběr definovaných podmnožin z databáze

4.6 Test aplikace v emulátoru a na mobilním telefonu

Na následujících obrázcích je nejdříve znázorněna modelová situace, ve které je kompletní program nainstalován do emulátoru. Tomu jsou již přednastaveny potřebné parametry, aby komunikoval s emulovaným modulem, který bude odesílat SMS zprávy. Ten se bude tedy chovat jako ukradené vozidlo. Na závěr této kapitoly je zobrazeno několik fotografií, znázorňujících, jak aplikace vypadá v reálném zařízení.



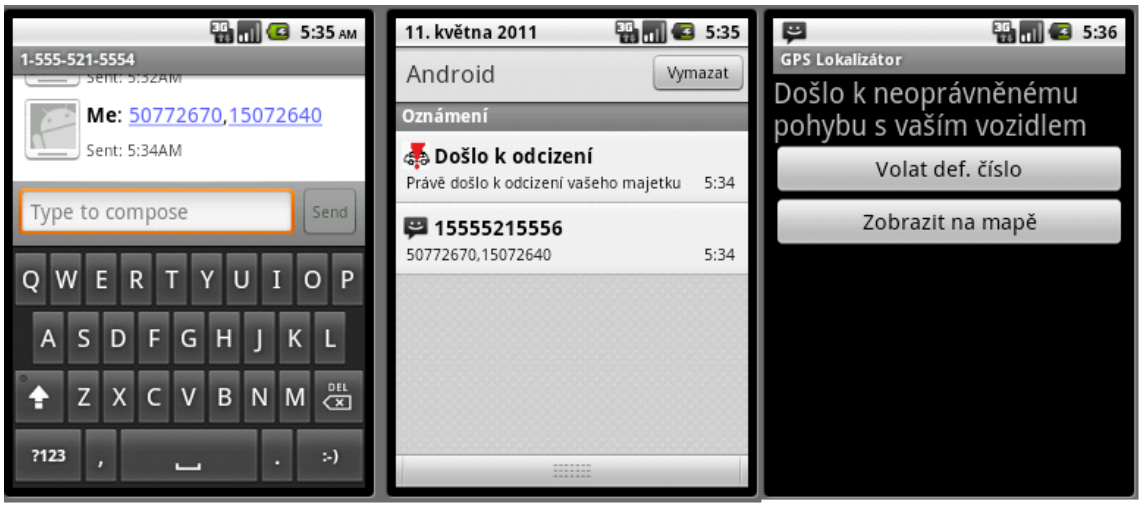
Obr. 11: Emulátor – průběh instalace a nastavení

Na první části obrázku je vidět již několikrát zmiňované okno generované z manifest souboru. Na druhé části pak, že se instalace zdařila a tlačítkem Otevřít ji lze spustit. Objeví se menu s nastavením. Zde můžete hlavním zatržítkem aktivovat sledování příchozích SMS zpráv. Pokud toto zatržítko není zaškrtnuté, aplikace o příchozí SMS zprávy nejeví žádný zájem. Dále je možné vybrat zvuk alarmu. Na výběr je jedna ze dvou možností. Dále pak Telefonní číslo, na které je možné jedním kliknutím volat při stresové situaci, která je vyvolána odcizením dopravního prostředku. Číslo modulu je hned po zatržítku Aktivovat sledování nejdůležitější hodnota. Pokud nebude nastaveno korektně, i správně poslané poplašné zprávy nebudou vyhodnoceny jako užitečné pro aplikaci.

Poslední dvě položky jsou Zpráva alarmu a Interval. Zpráva alarmu je celkem nepodstatná, zobrazí se vám na poplašné obrazovce, pro rychlou orientaci ve vzniklé situaci. Jejím obsahem může být jakýkoli uživatelsky definovaný text.

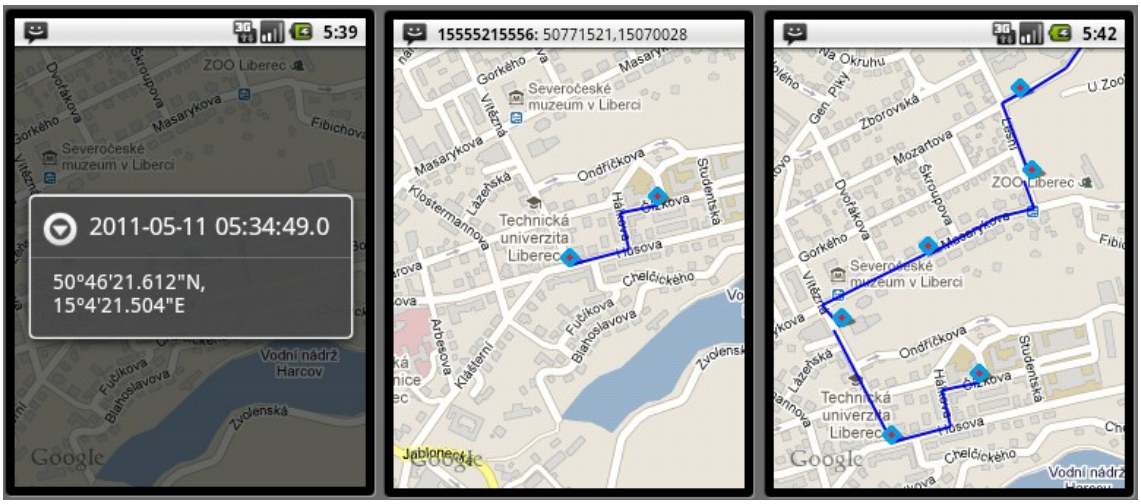
Interval je naopak velmi důležitá hodnota. Určuje předpokládaný rozestup mezi jednotlivým poplachy. Udává se v hodinách. Přednastavená hodnota je dvanáct hodin.

Je dost nepravděpodobné, aby ke krádeži vozu došlo během takto krátké chvíle vícekrát.



Obr. 12: Emulátor – reakce na SMS

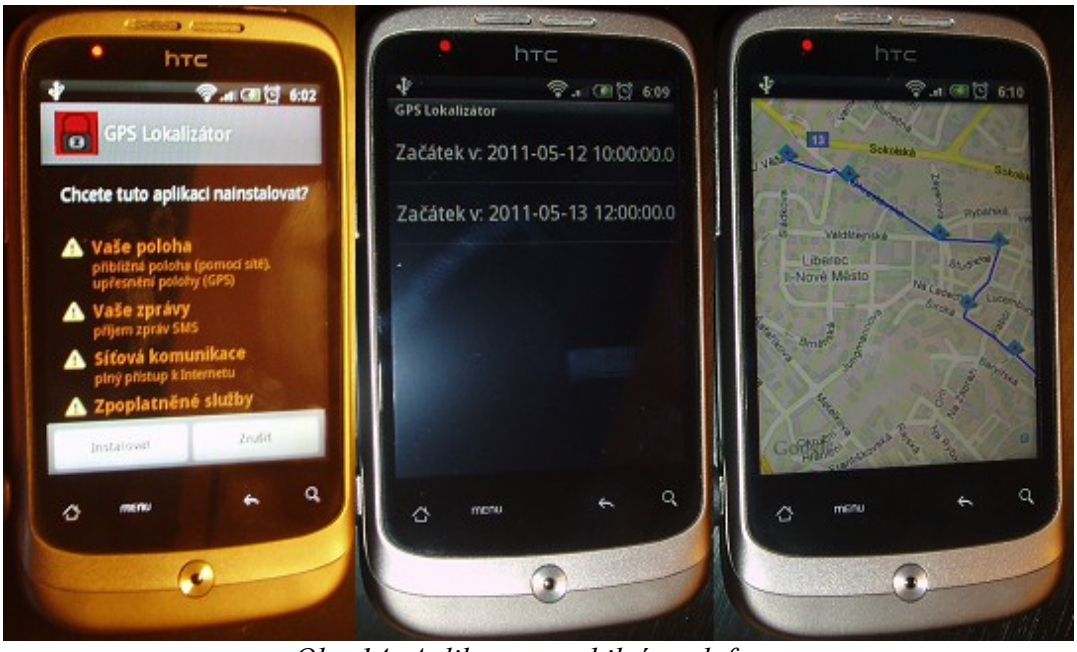
Z levé strany první část představuje emulaci GSM modulu, z kterého přijde poplašná zpráva. Je zde vidět napsaná aktuální souřadnice modulu ve formátu 50772670, 15072640, tato souřadnice představuje 50,772670° a 15,072640°. Standardní souřadnice GPS tak jak ji známe, se většinou uvádí ve tvaru DD°MM'SS.SSS"X. DD – stupně, MM – minuty, SS.SSS – sekundy s přesností na tři desetinná místa, X – znak X může nabývat podob písmen N (north, sever) S (south, jih) W (west, západ) E (east, východ). K určení polohy jsou potřeba dvě. V tomto případě je nejpříhodnější použít desítkový formát. V prostřední části vidíte reakci na příchod poplašné zprávy. Přehraje se vybraná melodie. Telefon také zavibruje a začne blikat indikační dioda. Bohužel v tomto ohledu je emulátor nedokonalý a vibrace ani blikající indikační diodu nelze nijak otestovat. V horní části obrazovky v taskbaru se objeví hláška vysvětlující vibrování a zvonění. Pokud taskbar stažením dolů otevřete, tak jak je patrné z obrázku, vidíte celý text zprávy. Po kliknutí na položku z taskbaru prichází poslední část obrázku. Zde je vidět poplašná obrazovka se dvěma tlačítky a přednastaveným textem. První zahájí hovor na předdefinované číslo. Druhé zobrazí mapu.



Obr. 13: Emulátor – zobrazení trasy

Poslední částí testu je ukázka zobrazení příchozích souřadnic na mapě a tvorba cesty mezi nimi. Pokud je v rámci sledování pouze jeden bod v databázi, cesta logicky není vytvořena. Pokud na bod kliknete, zobrazí se datum, čas a souřadnice ve známém tvaru, nikoli desítkovém, který byl odeslán ve zprávě. Toto vidíte na první části obrázku. Prostřední část znázorňuje situaci po zpracování druhého bodu (souřadnice). V tuto chvíli se již cesta vytvoří. I nadále můžete na oba body kliknout a zobrazit si tak jejich pozici a čas. V pravé části obrázku je poslední případ, ve kterém již bylo přijato více souřadnic, a je mezi nimi vytvořena trasa.

Na obrázku číslo 14 naleznete tři fotografie testovaného telefonu, které ve zkrácené formě prezentují to, co bylo ukázáno na předchozích obrázcích pro emulátor.



Obr. 14: Aplikace v mobilním telefonu

Testovaný telefon je HTC Wildfire zakoupený od firmy T-Mobile. Na tento telefon byla také celá aplikace vyvíjena. Velikost displaye je 3,2 palce při rozlišení 320x240 pixelů. Na mobilní telefon byla nahrána upravená verze aplikace, ve které se při prvním spuštění naplní databáze dvěma sledováními. Ty je pak možné zobrazit. Na první části opět výpis z obsahu souboru AndroidManifest. Na prostřední části vytvořený seznam sledování z databáze pomocí předdefinovaného intervalu. Na části úplně vpravo je zobrazeno jedno ze dvou sledování. Na zobrazené body lze opět kliknout a vyvolat jejich GPS pozici v mapě a kdy byly zaznamenány do databáze.

Závěr

Z kapitoly 2 vyplývá, že vývoj aplikací pro OS Android je velice přátelský a intuitivní. Vytvoření základního spustitelného projektu mezi programátory nazývaným „Ahoj světe“ je otázkou několika desítek minut. Vývojový plugin od Androidu v kombinaci s prostředím Eclipse je silná zbraň pro rychlý a jednoduchý vývoj i poměrně složitých aplikací. Dalším praktickým přínosem je pak emulátor telefonu, který vám umožňuje vyzkoušet své vlastní aplikace na různých variacích zařízení, která jsou běžně používána širokou veřejností, bez nutnosti je přímo vlastnit. Ať už je to nejnovější dotykový telefon HTC Desire s velikým displayem a rychlým procesorem, nebo výrazně pomalejší verze telefonu: HTC Wildfire se skromnější obrazovkou.

Z kapitoly 3 se lze dozvědět, že systém Android dbá na bezpečnost. Programátora svou strukturou a funkcionalitou nutí k jasnému definování toho, jaké třídy jeho aplikace obsahuje a které služby mohou tyto třídy po operačním systému požadovat. Všechny tyto údaje jsou uloženy v jediném souboru AndroidManifest.xml, který musí být součástí každé aplikace. Tento soubor je převzat instalátorem mobilního telefonu a zobrazen uživateli. Ten se na jeho základě může rozhodnout, zda povolí instalaci dané aplikace.

Pokud se k aplikaci přiřadí funkční GSM modul, měla by být plnohodnotně využitelná na mobilním telefonu. Měla by splňovat všechna požadovaná kritéria. Pro její fungování ji stačí nainstalovat a nastavit pár základních parametrů, jako je číslo modulu, ze kterého budou přicházet SMS zprávy a předpokládaný interval mezi jednotlivými poplachy. Jeho význam je definován v kapitole 4 . Data ze všech SMS zpráv budou ukládána do SQL databáze a uživateli bude umožněno zpětně si prohlížet trasy všech poplachů, které při běhu aplikace vznikly. Přijaté SMS zprávy tedy mohou být okamžitě smazány. Není potřeba mít jedna data uložena v telefonu vícekrát. Aplikace je pro tvorbu trasy při více souřadnicích velmi pomalá. Toto je způsobeno nutností generovat každý úsek mezi jednotlivými body samostatně.

Pro další rozvoj aplikace připadá v úvahu přidat možnost spravovat více GSM modulů najednou. Sledovat tedy více objektů zároveň. Zajímavé by také bylo, pokusit se implementovat vlastní mapové podklady a odbourat tak nutnost připojení k internetu.

Seznam zkratek

3D – 3 Dimensions

3G – 3rd Generation

ADT – Android Development Tool

API – Application Programming Interface

apk – Android Package

GMT – Greenwich Mean Time

GPS – Global Positioning system

GSM – Global System for Mobile Communications

IBM – International Business Machines Corporation

KML – Keyhole Markup Language

OHA – Open Handset Alliance

OpenGL ES – Open Graphics Library for Embeded Systems

OS – Operating System

SDK – Software Development Kit

SMS – Short Message Service

SQLite – Structured Query Language Lite

WiFi – Wireless Fidelity

Seznam použité literatury

[1] *Wikipedie, otevřená encyklopedie*. [online]. 24. 4. 2011 [cit. 2011-05-11]. Android (operační systém). Dostupné z WWW: <http://cs.wikipedia.org/wiki/Android_(operační_systém)>.

[2] *Eclipse - The Eclipse Foundation open source community website*. [online]. c2011 [cit. 2011-05-11]. About the Eclipse Foundation. Dostupné z WWW: <http://www.eclipse.org/org/>.

[3] *Android Developers*. [online]. c2011 [cit. 2011-05-11]. ProGuard. Dostupné z WWW: <http://developer.android.com/guide/developing/tools/proguard.html>.

[4] *Android Developers*. [online]. c2011 [cit. 2011-05-11]. Application Fundamentals. Dostupné z WWW: <http://developer.android.com/guide/topics/fundamentals.html>.

[5] *Android Developers*. [online]. [cit. 2011-05-11]. What is Android?. Dostupné z WWW: <http://developer.android.com/guide/basics/what-is-android.html>.

[6] *Dalvik Virtual Machine insights*. [online]. c2008 [cit. 2011-05-11]. Dalvik Virtual Machine. Dostupné z WWW: <http://www.dalvikvm.com/>.

[7] *Android Developers*. [online]. c2011 [cit. 2011-05-11]. Activities. Dostupné z WWW: <http://developer.android.com/guide/topics/fundamentals/activities.html>.

[8] *Android Developers*. [online]. c2011 [cit. 2011-05-11]. Services. Dostupné z WWW: <http://developer.android.com/guide/topics/fundamentals/services.html>.

[9] *Android Developers*. [online]. c2011 [cit. 2011-05-11]. Content Providers. Dostupné z WWW: <http://developer.android.com/guide/topics/providers/content-providers.html>.

[10] *Android Developers*. [online]. c2011 [cit. 2011-05-11]. Intents and Intent Filters. Dostupné z WWW: <http://developer.android.com/guide/topics/intents/intents-filters.html>.

[11] *Daviscomms*. [online]. c2010 [cit. 2011-05-19]. EaziTRAC1000. Dostupné z WWW: <http://www.daviscomms.com.sg/products/eazitrac1000/EaziTRAC1000.pdf>.

[12] *Android Developers*. [online]. c2011 [cit. 2011-05-19]. Using Shared Preferences. Dostupné z WWW: <http://developer.android.com/guide/topics/data/data-storage.html#pref>.

[13] *Google Projects for Android*. [online]. c2011 [cit. 2011-05-17]. Obtaining a Maps API Key. Dostupné z WWW: <http://code.google.com/android/add-ons/google-apis/mapkey.html>.

[14] *SQLite*. [online]. c2011 [cit. 2011-05-17]. About SQLite. Dostupné z WWW: <http://www.sqlite.org/about.html>.

Seznam obrázků

Příloha – CD

Na přiloženém CD je kompletní projekt obsahující zdrojové kódy a zkompilovaný soubor GPSLokalizator.apk, kterým lze aplikaci nainstalovat do mobilního telefonu. Další součástí je bakalářská práce v elektronické podobě.