VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

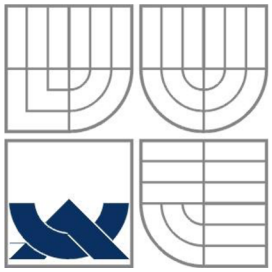
PODPORA GSM ALARMU NA MOBILNÍM ZAŘÍZENÍ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

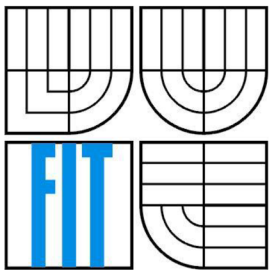
AUTOR PRÁCE
AUTHOR

Bc. JIŘÍ KALUS

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PODPORA GSM ALARMU NA MOBILNÍM ZAŘÍZENÍ

GSM ALARM SUPPORT ON MOBILE DEVICE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JIŘÍ KALUS

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JIŘÍ KOUTNÝ

BRNO 2011

Abstrakt

Tato diplomová práce popisuje mobilní aplikaci, která poskytuje podporu pro ovládání GSM/GPS alarmů. Aplikace obohacuje tyto alarmy o novou funkcionalitu, která vychází z možností těchto bezpečnostních zařízení a novodobých mobilních přístrojů. Práce se zaobírá především výběrem vhodné mobilní platformy, návrhem samotné aplikace pro tuto platformu a způsobem, jakým bude zaručena nezávislost programu na konkrétním typu alarmu. V závěru práce jsou poté vyhodnoceny dosažené výsledky a popsány možné rozšíření aplikace do budoucnosti.

Abstract

This master's thesis describes the mobile application that provides the support to control the GSM/GPS alarms. Based on the possibilities of modern security equipment and mobile devices, the application adds new functionality for these alarms. This work mainly describes selecting the appropriate mobile platform, application designs and the independence on the specific alarm. In conclusion, the results and possible extension of the application are talked over.

Klíčová slova

GSM/GPS alarmy, mobilní platformy, Android, Java, MVC, XML.

Keywords

GSM/GPS alarms, mobile operating systems, Android, Java, MVC, XML.

Citace

Kalus Jiří: Podpora GSM alarmu na mobilním zařízení, diplomová práce, Brno, FIT VUT v Brně, 2011

Podpora GSM alarmu na mobilním zařízení

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Jiřího Koutného. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……

Jiří Kalus

20. května 2011

Poděkování

Velmi rád bych poděkoval Ing. Jiřímu Koutnému za poskytnutí odborné pomoci a cenných rad při tvorbě této diplomové práce.

Obsah

1 Úvod

Se zprávami o krádeži automobilu či o vyloupeném bytu se setkáváme téměř každý den. Není proto divu, že poptávka po zabezpečovacích systémech na trhu neustále roste a tudíž i jejich vývoj je hnán značně kupředu. Kde jsou ty časy, kdy pod slovem alarm si člověk představil pouze sirénu, která po narušení chráněného objektu začala houkat a snažila se na pachatele upozornit okolí. Alarmy v dnešní době disponují širší škálou možností, jak na vloupání upozornit nebo jak mu zabránit. Z tohoto pohledu bylo výrazným krokem kupředu zapojení telefonní komunikace pro ohlášení poplachu. Systémy v takové situaci mohou kontaktovat centrálu bezpečnostní agentury či rovnou policii, které mohou okamžitě zasáhnout.

Souběžně s masovým rozšířením mobilních telefonů byla i tato funkce alarmu postupem času rozšířena o možnost vzdáleného ovládání. Bezpečnostní systémy lze v současné době ovládat vzdáleně prostřednictvím volání do elektronického záznamníku či zasláním příkazových SMS zpráv. Ty však často mívají velmi těžko zapamatovatelný formát, což vzdálenou správu velmi komplikuje. Z tohoto důvodu se zrodila myšlenka vytvoření mobilní aplikace, která by tuto komunikaci usnadnila.

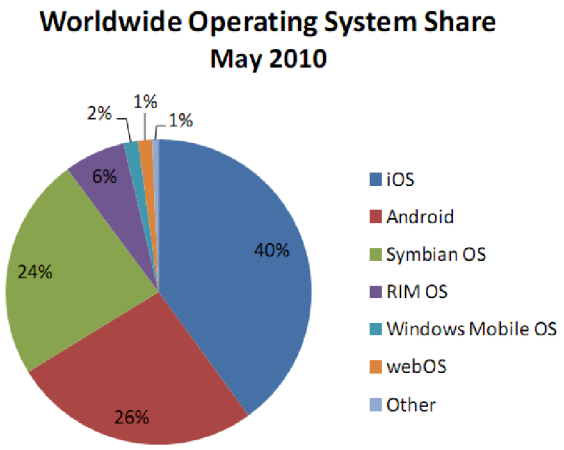
Tato diplomová práce se zabývá tvorbou programu pro mobilní zařízení, který má umožnit uživateli vzdáleně spravovat bezpečnostní systémy, aniž by musel vlastnoručně zadávat SMS příkazy či si je pamatovat. Toto ovládání musí být velmi intuitivní a zcela nezávislé na konkrétním typu alarmu. V kombinaci s novými mobilními telefony (tzv. smartphony) a jejich schopnostmi by aplikace měla dále obohatit alarmy o určitou novou funkčnost, která například využije mobilního připojení k internetu, GPS modulu nebo mobilních senzorů. V této souvislosti by také bylo vhodné využít další nové vlastnosti některých alarmů, které umožňují pomocí GPS lokalizátoru zjistit polohu hlídaného objektu. S touto informací by aplikace měla umět pracovat, uchovávat ji a vhodným způsobem zobrazovat tak, aby uživatel pro tyto účely nemusel využívat služeb třetích stran.

Věřím, že tato aplikace bude ve výsledku velmi úspěšná a bude velmi nápomocna mnoha lidem při zabezpečení a ochraně jejích majetku a možná i v budoucnu dodávána přímo jako součást některých alarmů.

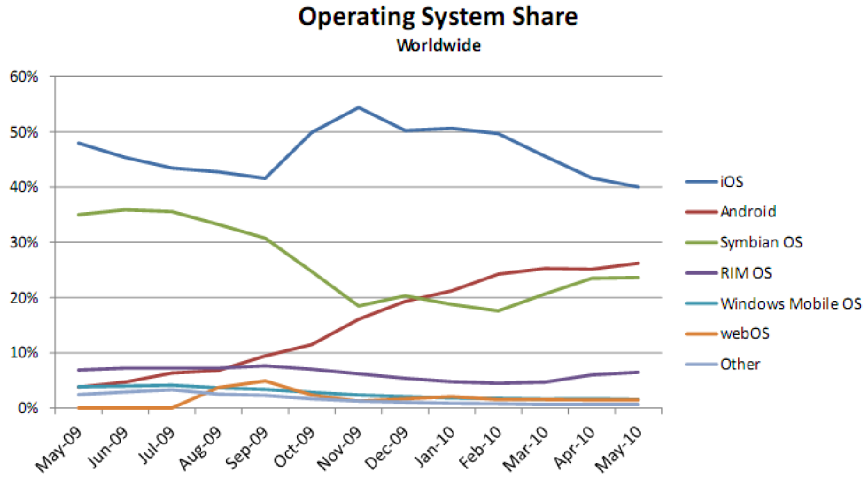
2 OS pro mobilní zařízení

Většina z nás vlastní mobilní telefon nebo jiné přenosné zařízení, nelze se tedy divit, že v rámci IT sektoru toto odvětví představuje značný potenciál. Původně tato zařízení neposkytovala možnost implementovat vlastní aplikaci, avšak s postupem času jejich výpočetní výkon rostl, což umožnilo vytvořit mobilní operační systémy, které by nasazení vlastní aplikace umožňovaly.

V současné době na trhu nalezneme spoustu mobilních operačních systémů, my se však zaměříme na ty, které se používají především v tzv. smartphonech, tedy telefonech, které nabízí pokročilé funkce (připojení k internetu, GPS lokalizaci, či dotykový displej). Jak můžeme vidět z obrázku č. 1 a také z obrázku č. 2, mezi přední představitele patří operační systémy iOS, Android, Symbian a Windows Mobile, se kterými se v této kapitole více seznámíme.



Obrázek 1: Celosvětový podíl mobilních operačních systému v květnu 2010 [1].



Obrázek 2: Historický růst podílu jednotlivých mobilních OS v období květen 2009 až květen 2010 [1].

2.1 iOS

Operační systém iOS je v dnešní době nejpoužívanějším systémem na světě v oblasti mobilních zařízeních (viz graf na obrázku č. 1). Tento systém byl vyvinut společností Apple původně pouze pro telefony iPhone, nicméně nyní se s ním můžeme setkat také v dalších zařízeních, jako jsou například iPod Touch, iPad nebo Apple TV.

Základem tohoto systému je jádro XNU s komponentami Darwin, které bylo převzato ze systému Mac OS X (proto lze tento systém zařadit do kategorie Unix-like). Model tohoto systému je tvořen čtyřmi abstraktními vrstvami**:**

- vrstvou jádra OS,
- vrstvou služeb,
- multimediální vrstvou,
- vrstvou Cocoa Touch.

Poslední zmíněná vrstva poskytuje GUI, které je značně přizpůsobeno pro ovládání více prsty či pomocí pohybů. Od nové verze iOS4 tento systém poskytuje zcela nový multitasking, který umožňuje běžet aplikacím na pozadí. K tomu lze využít těchto sedm služeb:

- **Background audio** – Služba, která umožňuje, aby aplikace přehrávala audio, i když aplikace běží na pozadí.
- **Voice over IP** – Služba umožňuje přijímat komunikaci přes IP, přestože aplikace není na popředí.
- **Background location** – Služba, jež umožňuje na pozadí efektivně přijímat souřadnice o poloze, aniž by docházelo k enormnímu vybíjení baterií.
- **Push notifications** – Tato služba umožňuje dostávat upozornění od vzdáleného serveru, aniž by aplikace musela běžet.
- **Local notifications –** Služba, která umožňuje upozornit uživatele na naplánovanou událost, aniž by bylo třeba využít serveru.
- **Task finishing –** Služba umožňující zpracování rozpracovaných výpočtů na pozadí.
- **Fast app switching** – Tato služba umožňuje rychlé přepínání mezi aplikacemi.

Jediným oficiálním jazykem pro vývoj aplikací pro tento OS je programovací jazyk objective-C či zkráceně objC. Abychom však mohli vytvářet aplikace pro tuto platformu, je třeba si zakoupit od společnosti Apple vývojářskou licenci, která nám poskytuje certifikát pro podpis programu. Podepsaný program lze poté nahrát na fyzické zařízení pouze prostřednictví App Store,

jinak to není možné. Z tohoto pohledu je tedy vývojář oproti ostatním operačním systémům značně omezen. [2]

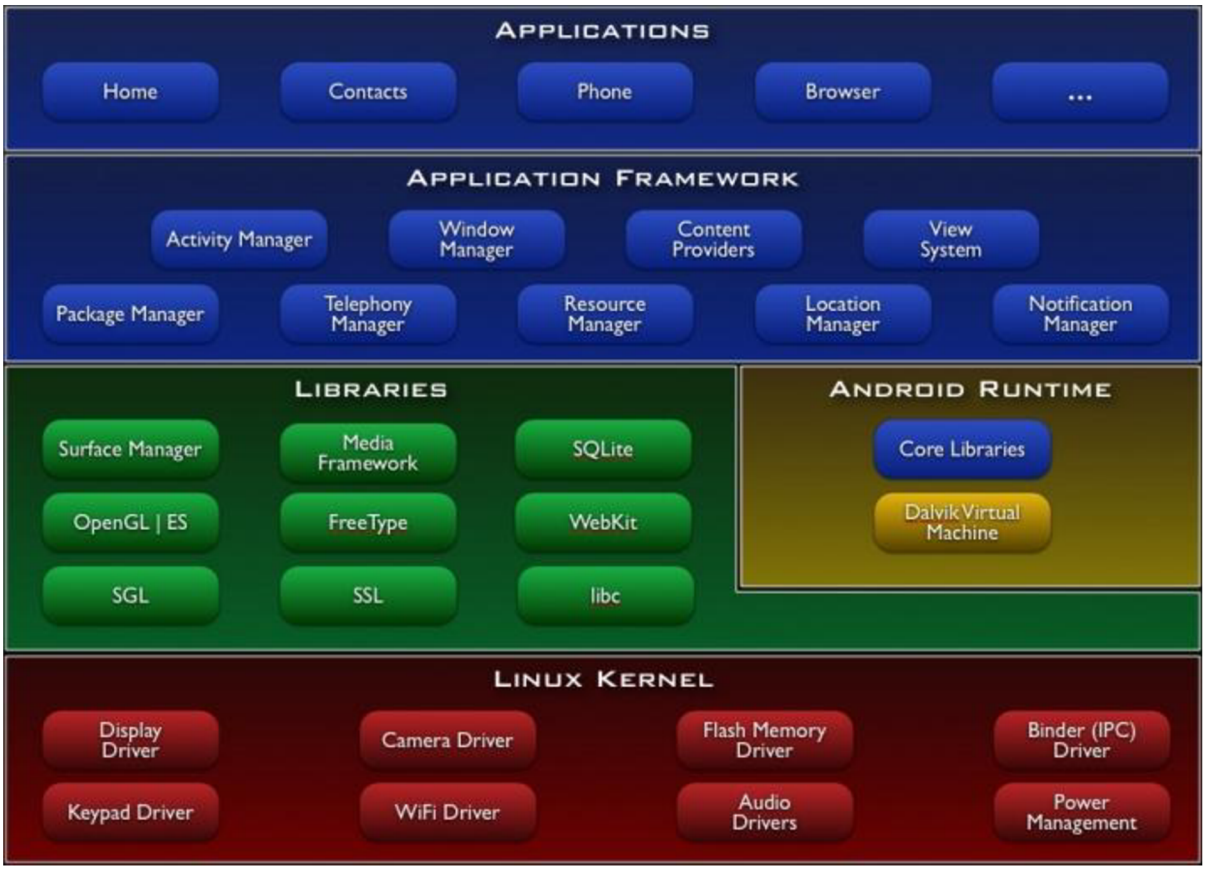
2.2 Android

Android je softwarová platforma vyvíjená sdružením Open Handset Alliance (jejím členem je např. společnost Google), jež je koncipována především pro použití v mobilních zařízeních a v současné době i v televizích. Tato platforma je založená na jádru Linuxu a skládá se z operačního systému, middleware a dalších nezbytných aplikací. Vyvíjet na této platformě lze v programovacím jazyku Java, můžeme tedy využít standardní vývojové prostředí jako například NetBeans, či Eclipse. Programy (oproti běžnému vývoji v tomto jazyce) však nejsou interpretovány klasickým virtuálním strojem, nýbrž speciálním, tzv. *Dalvik Virtual Machine*. Ten je uzpůsoben tak, aby při běhu aplikace bylo využito co nejméně paměti a výkonu procesoru, čímž je v systému docíleno snížení odběru energie z baterie. Pro spuštění je však nutné všechny Java soubory překompilovat do formátu *dex*[1]. [3]

Architektura tohoto systému je postavena na pěti hlavních komponentách (vrstvách), které umožňují vytvářet aplikace sdílející svou funkcionalitu s jinými, či aplikace, jejichž části lze vyměnit či opětovně využít. Těmito vrstvami jsou:

- **Aplikační vrstva** – Tato vrstva obsahuje základní aplikace, které jsou na mobilních zařízeních od začátku přítomny (emailový klient, kalendář, Google mapy, program pro zasílání SMS).
- **Aplikační Framework –** Hlavním cílem této vrstvy je zpřístupnit nejčastěji používané komponenty a umožnit vývojáři psát aplikace (další komponenty), které budou poskytovat data a služby dalším aplikacím.
- **Vrstva knihoven –** Množina C/C++ knihoven, které jsou využívány různými Android komponentami. Tyto knihovny umožňují například pracovat s 2D a 3D grafikou, nebo přehrávat či ukládat různé multimediálními formáty (MPEG4, H. 264, MP3, AAC, AMR, JPG, PNG). Díky knihovně SQLite je ve všech aplikacích přístupna odlehčená verze relační databáze, do níž lze ukládat perzistentní data.
- **Vrstva Android runtime –** Vrstva obsahující sadu knihoven programovacího jazyka Java a virtuální stroj Dalvik Virtual Machine, na kterém běží aplikace (každá aplikace má k dispozici vlastní instanci tohoto virtuálního stroje).
- **Vrstva jádra –** Abstraktní vrstva mezi SW a HW. Jádro vychází z Linuxu verze 2.6 a poskytuje veškeré systémové služby, jako správu paměti či správu procesů.

[1] Dalvik Executable



Obrázek 3: Architektura systému Android [3].

2.3 Symbian

Symbian je jeden z operačních systémů vyvíjený společností Nokia, který později přešel spolu s jeho uživatelskými rozhraními (S60, UIS, MOAP(S)) do vlastnictví nadace Symbian Foundation, jejímž hlavním cílem bylo poskytovat Symbian OS pod licencí open-source [4]. V únoru roku 2010 byla vydána první verze nové platformy Symbian pod zmíněnou licencí. Tato éra však neměla dlouhého trvání. V listopadu roku 2010 dochází k opětovnému přesunu platformy pod mateřskou společnost Nokia a ve své podstatě končí vývoj pod touto svobodnou licencí [5].

Nová Symbian platforma, která byla vydána ještě pod záštitou nadace Symbian Foundation, vychází z původního systému a je založená na jeho rozhraní S60. Na rozdíl od svého předchůdce, lze použít jak na procesorech ARM, tak i procesorech s instrukční sadou x86. Ke svému běhu využívá micro-jádro, které poskytuje pouze minimum funkcionality, čímž dokáže poskytovat odezvu téměř real-time. Systém poskytuje zpětnou kompatibilitu se staršími S60 verzemi a také kompatibilitu napříč všemi zařízeními. Vývojáři je tedy zaručeno, že aplikace poběží na všech přístrojích stejně.

Tohoto bylo docíleno využitím třívrstvého modelu, který systém tvoří. Model se skládá z následujících vrstev:

- **Aplikační vrstva** – Vrstva obsahující veškeré aplikace, které jsou součástí Symbian platformy (organizátory, multimediální aplikace, programy pro ovládání nastavení).
- **Middleware vrstva** – Vrstva poskytující služby pro aplikace (programy, se kterými uživatel komunikuje prostřednictvím UI), jako například zasílání zpráv, přístup k multimédiím nebo IP služby.
- **OS vrstva** – Vrstva provádějící odstínění HW a jádra OS, která umožňuje vyšším vrstvám nezávislý přístup.

Značnou výhodou, především pro programátory, je fakt, že se systém neváže na konkrétní programovací jazyk. Pro vývoj aplikací na této platformě lze využít celou řadu jazyků, například Symbian C++, Python, Ruby, Java ME, .Net nebo Flash. Od nové verze Symbian[4] můžeme dokonce využívat Qt toolkit, jenž umožňuje velmi rychlé vytvoření GUI a poskytuje rozsáhlou sadu knihovních funkcí, které vývoj značně ulehčí. Není proto divu, že tento OS je velmi populární a to jak u programátorů, tak i u uživatelů. Do svých mobilních přístrojů jej proto instaluje celá řada předních výrobců, jako je například Nokia, Sony Ericsson, Motorola a další, díky čemuž tato platforma patří mezi nejrozšířenější systémy na trhu (obrázek č. 1). [6][7][8]

2.4 Windows Mobile

Windows Mobile je kompaktní verze operačního systému Windows od společnosti Microsoft, který je určen pro mobilní zařízení a chytré telefony. Původně byl tento systém zaměřen pro využití v průmyslu, nicméně díky společnosti HTC došlo k jeho rozšíření do řad široké veřejnosti, kde začal zaujímat pozici významného „hráče“ na trhu.

Architektura Windows Mobile je založena na 32b virtuálním adresovém prostoru, jenž lze rozdělit do maximálního počtu 64 slotů po 32MB. Lze mít spuštěno maximálně 32 procesů. OS využívá API založeného na Win32 a poskytuje podporu pro ovládání stylusem. Ovládání pomocí prstů bylo umožněno až díky nástavbě od společnosti HTC, známé jako HTC Sense. [9]

2.4.1 Windows Mobile 5.0

Windows Mobile ve verzi 5.0 (známý pod kódovým označením Magneto) lze považovat jako první verzi tohoto OS, jež byla určena také pro „mainstream“. Systém je postaven na Windows CE 5.0 [10] a využívá .NET Compact Framework 1.0 SP2 [11], který umožňuje vyvíjet aplikace založené na .NET Frameworku. Systém přináší změnu při práci s pamětí. Do paměti RAM se ukládají pouze

programy a data, které jsou často používány. Vše ostatní je ukládáno ve flash paměti. Tato změna přináší výraznou úsporu spotřeby energie a také zabraňuje ztrátě dat při vypnutí telefonu po vybití baterie, jelikož data jsou uchována na nevolatilním médiu. Mezi další nové vlastnostmi tohoto systému patří:

- rozhraní pro správu GPS záznamů,
- rozšíření podpory Bluetooth a USB 2.0,
- podpora Exchange 2003 SP2,
- podpora QWERTY klávesnice.

2.4.2 Windows Mobile 6.0

Následníkem Windows Mobile 5.0 se stal OS s označením „Crossbow“ (Windows Mobile 6.0). Tento systém se oproti svému předchůdci vyznačuje především větší stabilitou a také velmi silným napojením na služby Windows Live a produkty Exchange 2007. Využívá .NET Compact Framework v.2 SP2 a Microsoft SQL Server 2005 Compact Edition, jež jsou předinstalovány v paměti ROM. Vývoj aplikací pro tuto platformu je tedy snazší a umožňuje využívat sofistikovanějších prostředků, které by si vývojář jinak musel obstarávat sám. Další velmi vítanou novinkou je podpora technologií AJAX [12], JavaScript [13] a XMLDom [14] ve vestavěném prohlížeči Internet Explorer. V této verzi také dochází k zabezpečení dat pomocí šifrování.

Na trhu se tento systém vyskytuje ve třech variantách:

- **Windows Mobile 6 Standard** – Pro smartphony bez podpory dotykového displeje.
- **Windows Mobile 6 Classic** – Pro Pocket PC bez možnosti volání.
- **Windows Mobile 6 Professional** – Pro Pocket PC s možností volání.

2.4.3 Windows Mobile 6.5

Tato verze není příliš odlišná od předcházející, nicméně většina starších systémů byla na tuto verzi aktualizována. Systém přináší změnu hlavně v podobě nového GUI, které je upraveno pro ovládání prsty a také přináší možnost distribuovat vyvíjené aplikace prostřednictvím webové aplikace Marketplace. [15]

V současné době jde o poslední verzi tohoto systému, která se bude dále vyvíjet pouze a jen pro Pocet PC. Pro smartphone bude tento systém nahrazen zcela novým Windows Phone 7.

2.4.4 Windows Phone 7

Jde o zcela nový operační systém pro mobilní telefony, který není zpětně kompatibilní s předchozí platformou Windows Mobile. Tato nová platforma přináší spoustu inovativních technologií a myšlenek, jako například organizace kontaktů do hubů, která umožňuje propojit telefonní seznam s kontakty ze sociálních sítí (Facebook), nebo propojení telefonu s jinými přístroji v domácnosti, které lze navzájem synchronizovat. Tato platforma má však také spoustu omezení týkajících se hlavně HW telefonu. Telefony musí být vybaveny nejméně 1GHz procesory, akcelerometrem s kompasem a 6 HW tlačítky, která jsou využita pro návrat zpět nebo pro zobrazení nabídky „Start“. Vývoj aplikací na této platformě je založen na technologii Silverlight 3 [16], XNA [17] a .NET Compact Frameworku 4. Součástí platformy je široká integrace služeb společnosti Microsoft, jako například vyhledávání a GPS lokalizace prostřednictvím služby Bing nebo napojení na Xbox Live, který umožňuje hrát online multiplayer hry. Jediným místem, odkud mohou uživatelé stahovat do mobilu aplikace, bude Microsoft Marketplace (obdobně jako u iPhonu). Tato politika však přináší značné omezení jak pro uživatele, tak i pro vývojáře, jelikož v současné době je tato služba přístupna pouze z určitých států, mezi které nepatří téměř žádný ze střední či východní Evropy. [18]

3 GSM/GPS alarmy

Moderní zabezpečovací zařízení jsou v dnešní době často obohacena o GSM moduly, které umožňují vzdálenou komunikaci s majitelem prostřednictvím zasílání SMS zpráv nebo rovnou zavoláním na předem uvedené číslo. Majitel se tedy o incidentu okamžitě dozví a může ihned kontaktovat policii, popřípadě vzdáleně zkontrolovat stav a v případě planého poplachu alarm deaktivovat.

V současné době se také objevují autoalarmy, které jsou kromě GSM modulu vybaveny rovněž GPS lokalizátorem. Tato kombinace dvou moderních technologií umožňuje v případě krádeže sledovat aktuální pozici vozidla. Takto získané informace lze pak snadno využít pro nalezení automobilu a k dopadení pachatele provedeného trestného činu.

V této kapitole se tedy především zaměříme na autoalarmy, které obsahují jak GSM modul, tak i GPS čip. Budeme se zabývat zejména jejich vzdálenou komunikací a ovládáním pomocí příkazových SMS. Na závěr kapitoly si uvedeme jednoho zástupce alarmů pro zabezpečení domů, abychom tyto dva druhy porovnali a mohli rozdíly při tvorbě aplikace zohlednit.

3.1 GSM alarm CA-1803 Athos

Autoalarm s označení CA-1803 Athos pochází od české společnosti Jablotron s.r.o, která je významným výrobcem automatizačních a zabezpečovacích zařízení jak v České republice, tak i na Slovensku. Toto zařízení vychází ze staršího GSM modelu CA-1802, který však obohacuje o modul GPS, jenž umožňuje získat velmi přesné souřadnice prostřednictvím satelitního systému. Přestože hlavním úkolem tohoto zařízení je hlídat vozidlo před odcizením, poskytuje tento modul i další služby, jako například automatickou tvorbu knihy jízd či sledování provozu na silnici. Zařízení je tedy velmi univerzální.

Stejně jako jeho předchůdce, lze i toto zařízení ovládat dvěma způsoby. Buď klasicky prostřednictvím dálkového ovládání od vozidla, nebo prostřednictvím SMS zpráv.

Informace k tomuto alarmu byly čerpány z oficiální webové stránky firmy [19] a z oficiálního uživatelského manuálu [20].

3.1.1 SMS příkazy

Soupis všech SMS příkazů, které lze použít pro ovládání tohoto modulu, je uveden v tabulce č. 1. SMS příkazy lze zasílat z libovolného telefonu, ale vždy s uvedením uživatelského hesla (pokud jde o autorizovaný telefon, heslo není nutné zadávat). Tato zpráva musí mít následující tvar:

```
HESLO PŘÍKAZ
```

V textu těchto zpráv je povoleno používat jak velká, tak i malá písmena. Háčky, čárky a další interpunkční znaky však ve zprávě nejsou povoleny.

Pokud je automobil zajištěn alarmem a dojde k aktivaci poplašného zařízení (otevření dveří, aktivace detektoru), zašle zařízení na uvedené číslo poplašnou zprávu. Pro zjištění poslední známe polohy vozidla lze poté využít příkazu `GPS`. V reakci na tento příkaz alarm zašle zprávu například v tomto formátu:

```
Autoalarm hlasi: Poloha: (GMT:3.12.10:35)
50*43.495N;15*11.253E;515;50, Cas:3.12.05 11:35
```

kde jednotlivé části mají následující význam:

- **(GMT:3.12.10:35)** – Světový čas v Greenwichi.
- **50*43.495N;15*11.253E** – Souřadnice zeměpisné šířky a výšky.
- **515** – Nadmořská výška, ve které se vozidlo nachází.
- **50** – Aktuální rychlost automobilu.
- **Cas:3.12.05 11:35** – Aktuální čas zprávy.

Pro delší sledování lze poté využít příkaz `GPS ON`, který zapříčiní, že alarm bude maximálně po dobu 15 minut zasílat každou 1 minutu výše uvedenou zprávu o pozici. Nemusíme tedy příkaz pro zjištění polohy neustále opakovat. [20]

SMS příkaz	Akce alarmu
AM	Zajištění autoalarmu (s ovladači RC-80 i uzamčení vozu).
AM EXT	Částečné zajištění autoalarmu (s ovladači RC-80 i uzamčení vozu).
DM	Odjištění autoalarmu (s ovladači RC-80 i odemčení vozu).
IMO	Zablokování vozidla (trvalá imobilizace).
UNIMO	Odblokování vozidla.
STATUS	Vyžádání SMS o aktuálním stavu autoalarmu.
GPS	Vyžádání poslední zjištěné polohy GPS.
LOCATOR	Vyžádání SMS o přibližné poloze vozidla.
HELP	Vyžádání SMS s nápovědou.
SIREN	Alarm spustí sirénu na 30 vteřin.
CREDIT	Zjištění kreditu (při použití předplacené SIM).
AUX parametr	Ovládání spotřebičů.
UC xx...x	Nastavení nového hesla (heslo délky 4-8 znaků).

Tabulka 1: SMS příkazy pro ovládání GSM alarmu CA-1803 Athos [20].

3.2 TS GPS alarm

TS GPS je alarm, který pro svou činnost nepotřebuje žádné další komponenty. Zařízení se skládá ze dvou částí, jednotky GPS lokalizátoru a GPS antény, které mohou být nainstalovány odděleně. Pro zabezpečení vozidla nebo motocyklu poskytuje tři režimy zabezpečení:

1. **INPUT ochranu** – Zasílání SMS zprávy při vniknutí do vozidla.
2. **GPS ochranu** – Zasílání SMS zpráv obsahující GPS souřadnice polohy vozidla, které je v pohybu (bez zapnutého zapalování).
3. **GPSS ochranu** – Jde o režim zvýšené ochrany, kdy jsou zasílány GPS souřadnice i při zapnutém zapalování.

Ovládat tento modul lze pouze prostřednictvím SMS zpráv a to z jakéhokoliv telefonu. V příkazu je však nutné vždy uvést bezpečnostní kód, který tento příkaz autorizuje. Formát vybraných zpráv je popsán v následujících řádcích. Úplný seznam příkazů je uveden v tabulce č. 2. [21]

SMS příkaz	Význam
PIN	Nastavení nového hesla.
ITEXT	Nastavení textu SMS zprávy při INPUT poplachu (max. 52 znaků).
GTEXT	Nastavení textu SMS zprávy při GPS poplachu (max. 52 znaků).
PTEXT	Nastavení textu odpovědi při dotazu na GPS pozici.
PHONE	Nastavení autorizovaných čísel pro příjem zpráv o poplachu.
PERIOD	Nastavení periody pro opětovné zasílání GPS pozice při poplachu.
CONFIRM	Nastavení zařízení, aby (ne)potvrzovalo příjem SMS příkazů.
CONFIG	Žádost o zaslání konfigurace.
RESET	Resetování zařízení do továrního nastavení.
GSTART	Aktivace GPS ochrany.
GSTOP	Deaktivace GPS ochrany.
POSITION	Požadavek na GPS polohu.
SWITCH	Ovládání relé zařízení.
CREDIT	Požadavek na zjištění aktuálního kreditu na SIM kartě zařízení.
OFF	Úplné vypnutí lokalizátoru.

Tabulka 2: SMS příkazy pro ovládání alarmu TS GPS.

3.2.1 Konfigurace čísel pro příjem zpráv

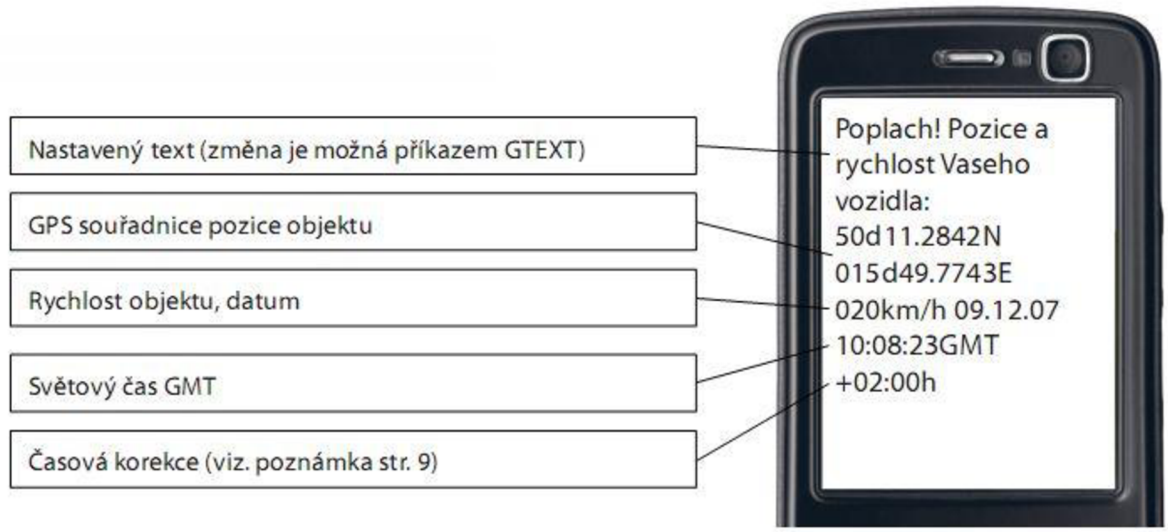
Zařízení umožňuje nastavit až tři telefonní čísla, na které v případě poplachu budou zasílány poplašné zprávy. Tato čísla se uvádí v mezinárodním formátu, aby systém fungoval i v zahraničí. Zpráva pro nastavení čísel má tento tvar:

```
HESLO PHONE +xxxxxxxxxxxxM  + xxxxxxxxxxxxM  +xxxxxxxxxxxxM
```

Parametr *M* na konci telefonních čísel umožňuje, aby na dané číslo byly zasílány pouze zprávy z vybraného bezpečnostního režimu. Pokud není zadán, budou zprávy zasílány vždy. Hodnoty tohoto parametru jsou zobrazeny v následující tabulce. Formát poplašné zprávy je poté znázorněn v obrázku č. 4.

Hodnota	Význam
I (velké písmeno 'i')	Při této volbě budou na telefonní číslo zasílány pouze zprávy při INPUT poplachu.
G	Na toto číslo budou zasílány pouze zprávy při GPS poplachu.

Tabulka 3: Hodnoty parametru M při konfiguraci telefonních čísel pro příjem poplašných zpráv.



Obrázek 4: Formát poplašné zprávy alarmu TS GPS [21].

3.2.2 Nastavení periody pro odesílání GPS pozice

Tato volba nastavuje intenzitu odesílání zpráv obsahující GPS lokalizaci, pokud dojde k pohybu vozidla (aktivace alarmu). Tato volba lze nastavit dvěma způsoby:

1. Nastavením časového intervalu (v minutách), po jehož vypršení dojde k opakovanému zaslání.

   HESLO PERIOD Txx

2. Nastavením vzdálenosti (v kilometrech), po jejímž ujetí dojde k zaslání nové GPS polohy.

   HESLO PERIOD Dxx

3.3 GPS LOC

GPS LOC[2] je jeden z představitelů GSM alarmu od společnosti Keetec [22], který může být využit jak pro hlídání automobilů, tak i například pracovních strojů. Tento systém je napájen buď přímo z baterie hlídaného stroje (12V) nebo z vlastního náhradního zdroje, který lze k alarmu dokoupit. Systém je tedy funkční i v případě, kdy při odcizení útočník odpojí vozidlo od baterie.

Jako všechny GSM alarmy i tento systém komunikuje s majitelem prostřednictvím SMS zpráv, je tedy nutné do mobilu vložit SIM kartu libovolného operátora buď s měsíčním paušálem, nebo přednabitým kreditem. SMS zprávy mohou být zasílány z libovolného telefonu, na začátku je však nutné vždy uvést uživatelské heslo, aby systém rozpoznal právoplatného uživatele. V případě, že uživatel zašle příkazovou zprávu ve špatném formátu nebo se špatně zvolenými parametry, modul mu odpoví zprávou s textem `COMMAND ERROR` nebo v druhém případě zprávou s klíčovým slovem `ERROR`.

Příkazy tohoto alarmu jsou obdobné jako u modulu TS GPS (viz kapitola 3.2), nicméně pár příkazů se liší, proto si je probereme.

Informace k celé této kapitole byly čerpány z oficiálního návodu [23], který je dodáván současně s alarmem nebo je přístupný na webových stránkách společnosti Keetec [22].

3.3.1 Aktivace/Deaktivace GPS ochrany

Aktivace a deaktivace GPS ochrany se v tomto zařízení provádí pouze pomocí jedné příkazové SMS zprávy, ve které je parametrem *X* zadáno, zda ochrana má byt spuštěna či vypnuta. Pokud uživatel do tohoto parametru zadá hodnotu 0, pak dojde k deaktivaci, pokud hodnotu 1, pak ostraha bude aktivována. Formát této zprávy vypadá následně:

HESLO GPS X

[2] GPS lokalizátor

3.3.2 Upozornění voláním

Modul umožňuje při incidentu upozornit majitele buď zasláním zprávy, nebo voláním. Aby mohl na autorizovaná čísla volat, je nutné na nich mít aktivní službu CLIP (identifikace volajícího), jinak by tento způsob varování nebyl funkční. Modul je také nutné daným způsobem nastavit a to pomocí zprávy obsahující klíčové slovo `CALL`.

3.3.3 Nastavení periody pro odesílání výstrahy

Jedním z příkazů, který má odlišný formát, je příkaz `PERIOD`, který slouží k nastavení intervalu pro odesílání výstražných SMS. Formát této zprávy je následující:

`HESLO PERIOD xy Nz`

Parametr '*x*' a '*y*' se oproti předešlému systému neliší, přibyl však nový parametr '*z*', který slouží pro nastavení maximálního počtu odeslaných zpráv při jednom incidentu. Systém tedy není limitován přednastaveným počtem a lze různě modifikovat.

3.3.4 Přepínání relé

Externí zařízení tohoto alarmu lze ovládat třemi způsoby:

1. Použitím zprávy, která aktivuje relé po dobu, která je přednastavená v bezpečnostním zařízení.

   `HESLO SWITCH`

2. Aktivací relé na neomezenou dobu, a jakmile je potřeba zařízení vypnout, tak jej opět deaktivovat. K této činnosti je třeba použít příkaz `SWITCH` s parametrem *X*, který nabývá hodnoty 1 pro aktivaci a hodnoty 0 pro deaktivaci.

   `HESLO SWITCH X`

3. Spuštěním externího zařízení pomocí příkazu `SWITCH`, který obsahuje dobu běhu zadanou buď v sekundách (parametr *U* musí být nastaven na hodnotu '*s*') nebo v minutách (v parametru *U* je nastavená hodnota '*m*'). Tento čas je zadáván v prvním parametru (*T*) a může nabývat hodnot od 0 až do 240.

   `HESLO SWITCH TU`

3.3.5 Zobrazení souřadnic na mapě v mobilu

Významnou změnou systému k lepšímu, oproti ostatním autoalarmům, je možnost přímého zobrazení pozice vozidla na mapě v mobilu, k čemuž je využito webové služby *satmaps.net* [24]. Tato možnost je aktivní pouze tehdy, pokud je parametr `LINK` v alarmu nastaven na hodnotu 1. V případě bezpečnostního incidentu je poté uživateli doručena pozměněná zpráva, která obsahuje URL adresu odkazující přímo na mapu. Formát této zprávy je zobrazen v následujícím obrázku.

Obrázek 5: Poplašná SMS zpráva systému GPS LOC, která obsahuje odkaz pro zobrazení souřadnice na mapě [23].

3.4 SEL-01

Poslední alarm, jehož specifikaci si uvedeme, se od těch ostatních liší tím, že nejde o alarm pro ochranu vozidla, nýbrž o alarm, který se používá k zabezpečení rodinných domů a staveb. Hlavní částí tohoto systému, jež se stará o chod, je ústředna. Ta bezdrátově komunikuje jak s ovládači, tak i s detekčními čidly, jejichž signály analyzuje a vyhodnocuje. V závislosti na aktuálním nastavení poté vyhlašuje poplach nebo udržuje systém v klidu.

Přestože jde o jeden z levnějších alarmů, systém je možné ovládat více způsoby. Buď pomocí dálkového ovládaní nebo použitím pevného či mobilního telefonu (přímým voláním nebo SMS zprávami). Značnou výhodou tohoto systému je také možnost napojení na různá externí zařízení, jako například domácí spotřebiče či topení. Pomocí příkazových SMS pak můžeme například korigovat teplotu uvnitř domu anebo ovládat jeho osvětlení. Příkazové SMS zprávy jsou však u tohoto systému velmi těžko zapamatovatelné, jelikož pro ovládání se nepoužívají klíčová slova, nýbrž číselné kódy. Část těchto příkazových zpráv si nyní uvedeme. [25]

3.4.1 Nastavení telefonních čísel pro ohlášení poplachu

Jak již bylo v předešlé části zmíněno, alarm pro ovládání používá číselné kódy, před nimiž musí být uvedeno heslo. Tento alarm lze nastavit pro oznamování poplachu až pro šest telefonních čísel a to pomocí následující příkazové zprávy, kde parametr *X* představuje pořadové číslo telefonu pro uložení (1-6).

```
*HESLO#838*PREDVOLBA_STATU#83X**TEL_CISLO#83X**TEL_CISLO#83X**TEL_CI
SLO#83X**TEL_CISLO#83X**TEL_CISLO#83X**TEL_CISLO#83X**TEL_CISLO#851*
TEXT#
```

Jako příklad si zde můžeme uvést zprávu, která do alarmu uloží dvě telefonní čísla (+420777111333 a +420596017895) a text zprávy nastaví na „*Poplach*“.

```
*1234#838*420#831**777111333#832*596017895#851*Poplach#
```

3.4.2 Aktivace a deaktivace alarmu

Pokud dojde k aktivaci alarmu, jsou okamžitě rozeslány SMS zprávy na uložená telefonní čísla a v domě je spuštěna poplašná siréna. Ta vždy houká maximálně po předem nastavenou dobu, kterou lze nastavit pomocí tohoto příkazu:

```
*HESLO#814*X#
```

Parametr *X* zde představuje čas uvedený v minutách, který může nabývat hodnot z intervalu 0 – 30. Pokud však chceme alarm vypnout dříve, lze využít zprávy pro jeho deaktivaci, která má tento tvar:

```
* HESLO#813*1#
```

Tuto zprávu lze poté použít také pro opětovnou aktivaci alarmu, kde je však poslední hodnota 1 nahrazena číslicí 0.

4 Specifikace aplikace

Jelikož zadání této diplomové práce nebylo iniciováno žádným výrobcem či prodejcem alarmů, neexistují žádné předem dané formální požadavky na funkčnost ani žádná kritéria, která by aplikace musela splňovat. Program však musí být obecný, nesmí být svázán pouze s jedním bezpečnostním zařízením, musí se tedy dát použít v kombinaci s různými přístroji od různých výrobců nebo s alarmy s rozdílným zaměřením (autoalarmy či alarmy pro ochranu objektů). Alarmy však musí vždy umět komunikovat prostřednictvím SMS zpráv, jejichž formát se může lišit od ostatních bezpečnostních systémů.

Specifikaci systému si tedy stanovíme sami. Při tomto návrhu se budeme snažit maximálně využít všech novodobých vymožeností mobilních zařízení a ty v kombinaci s možnostmi alarmů využít takovým způsobem, aby funkčnost celého systému byla co nejvíce obohacena. Důraz bude také kladen na co největší automatizaci, aby uživatel nemusel většinu operací provádět vlastnoručně, nýbrž aby je prováděl program samostatně, bez jakékoliv interakce. Návrh implementovaných metod bude vycházet z následující tabulky, která shrnuje všechny zjištěné poznatky týkající se možností alarmů z kapitoly č. 3. Soustředit se budeme především na autoalarmy, které poskytují také informaci o pozici (GPS souřadnice).

Funkce přístupné přes SMS / Alarm	CA-1803 Athos	TS GPS	GPS LOC	SEL 01
Nastavení telefonního čísla	X	X	X	X
Změna hesla	X	X	X	X
Zajištění/odjištění alarmu	X	X	X	X
Částečné zajištění/odjištění alarmu	X	X	X	
Zablokování/odblokování řízení vozidla	X			
Úplné vypnutí lokalizátoru		X	X	
Zjištění aktuálního nastavení alarmu	X	X	X	
Manuální vyžádání GPS polohy	X	X	X	
Automatické zasílání GPS polohy po určitý čas	X			
Zasílání GPS polohy při poplachu		X	X	
Nastavení periody pro opětovné zasílání polohy		X	X	
Zapnutí/Vypnutí upozorňovaní prostřednictvím SMS				X
Zapnutí/Vypnutí upozorňovaní voláním			X	X
Zapnutí/Vypnutí sirény	X			X
Nastavení času sirény				X
Reset zařízení do továrního nastavení		X	X	
Zjištění kreditu SIM karty	X	X	X	
Ovládání připojených spotřebičů	X	X	X	

Tabulka 4: Souhrn operací, které poskytují jednotlivé alarmy uvedené v kapitole č. 3.

4.1 Vzdálené ovládání alarmu

Minimální funkčnost, kterou určitě musí aplikace poskytovat, je vzdálené ovládání alarmu tak, aby uživatel nemusel psát příkazy manuálně. Jak jsme si mohli všimnout v předešlé kapitole, tak tyto SMS příkazy jsou značně rozdílné napříč všemi alarmy a ne vždy mají takový tvar, který by se dal snadno zapamatovat. Uživatelé tak často tyto příkazy mají předpřipraveny v telefonu a v případě potřeby je přeposílají. Tento proces však není příliš vhodný v situacích, kdy je nutno změnit parametry alarmu. Aplikace by tedy měla tuto činnost nahradit a umožnit uživateli ovládat alarm jednoduše prostřednictvím uživatelského rozhraní a sama tvořit SMS příkazy v závislosti na konfiguraci.

4.2 Upozornění na poplašnou zprávu alarmu

Pokud během zapnutého alarmu dojde k vniknutí či k jakémukoliv narušení hlídaného objektu, GSM alarm zavolá či zašle SMS zprávu na předdefinovaná čísla. Problém však může nastat, pokud si tohoto upozornění zavčas nevšimneme, například když má mobil zapnutý tichý režim a vyzvánění je tedy vypnuto. S tímto problémem by se měla aplikace vypořádat a to tak, že vypne tichý režim mobilu a pomocí dostupných prostředků telefonu začne majitele upozorňovat pomocí zvuků, vibrací či blikání displeje. Pokud by si i přes tyto akce uživatel varování nevšiml, měl by telefon po určité době zprávu přesměrovat na další telefonní čísla, která si majitel v mobilu pro tuto situaci nastavil.

4.3 Získání GPS polohy při poplachu

Alarmy a zařízení, které jsou vybaveny GPS lokalizátorem, v některých případech zasílají současně s poplašnou zprávou také zeměpisné souřadnice aktuální polohy hlídaného objektu. Ne vždy je však tato činnost automatická, někdy je nutné se na polohu dotázat dodatečně. Tuto režii by měla řešit sama aplikace. Pokud dojde k narušení, měla by se automaticky dotázat alarmu na aktuální polohu a tento dotaz po určitém čase opakovat, aby byla vždy známa co nejpřesnější poloha. Aktuální polohu a několik starších hodnot bude poté zobrazovat na mapě přímo v mobilu. Majitel tak získá velmi důležité informace, které mohou být využity k dopadení pachatele nebo nalezení ukradeného vozidla.

4.4 Sledování polohy vozidla

Jak ukazuje tabulka č. 4, všechny alarmy, které disponují GPS modulem, zasílají na požádání aktuální zeměpisné souřadnice vozidla. Této služby bude v aplikaci využito ke sledování pohybu vozidla (při zapnutém alarmu ve stavu klidu). Tuto funkčnost by mohli přivítat především řidiči,

kteří si musí vést knihu jízd a ujeté kilometry vykazovat. S touto podporou se budou moci kdykoliv zpětně podívat, kdy a kam cestovali. Alarm tak bude zastávat obdobnou funkci jako elektronická kniha jízd.

Nevýhodou této navržené funkce je, že při nastavení velmi malého intervalu pro získávání pozice, dojde k značnému navýšení toku SMS zpráv, které může vést až k vybití přednabitého kreditu SIM karty a tedy k zablokování funkčnosti celého systému. Bohužel, tento tok nelze nijak zredukovat. Jediným možným řešením, jak se vyhnout vysokým telefonním poplatkům, je využít vhodných mobilních tarifů, které mají dostatek volných SMS zpráv, nebo které umožňují zasílat zprávy na vybraná telefonní čísla zdarma. Toto však závisí pouze a jen na samotném uživateli, jaký tarif si v telefonu a v GSM alarmu zvolí.

4.5 Zobrazení polohy na mapě

V rámci obou předcházejících funkcí by bylo vhodné, aby jak aktuální pozici, tak i ujetou trasu, bylo možné zobrazit na mapě. Tato mapa by měla být integrována přímo v aplikaci, neměla by tedy pouze generovat odkaz na webové stránky s mapovými podklady. Vyhneme se tak situaci, kdy by zmíněná funkčnost přestala fungovat, pokud by se změnilo rozhraní webové služby.

4.6 Automatická aktivace/deaktivace alarmu

Některé GPS alarmy poskytují speciální režimy zabezpečení, které se zapínají a vypínají pouze přes SMS. Alarm v tomto režimu zůstává aktivní i po odemknutí vozu, i po zapnutí zapalování. Před každým použitím automobilu tedy musíme zaslat deaktivující zprávu, jinak alarm bude hlásit poplach.

Přestože tento typ zabezpečení vykazuje vysokou míru spolehlivosti, řidiči jej moc často nepoužívají, jelikož často zapomínají alarm před jízdou deaktivovat či opětovně zapínat. Tuto činnost je potřeba určitým způsobem zautomatizovat. Aplikace by měla rozpoznat, kdy je řidič ve vozidle a poté sama alarm vypnout. Naopak, pokud řidič opustí vůz, tak by alarm měl být opětovně aktivován. Bezpečnostní systém by tak fungoval obdobně jako elektronické karty u nových luxusních automobilů.

4.7 Ovládání spotřebičů pomocí plánovače

Některé alarmy mohou být propojeny s jinými externími zařízeními a umožňují je pomocí tohoto napojení vzdáleně zapínat či vypínat. K tomuto bude aplikace poskytovat dodatečnou podporu ve formě plánovače. Uživatel si bude moci navolit, kdy a která externí zařízení budou zapnuta a po jak dlouhou dobu. Bude možné si vybrat buď konkrétní den a hodinu, nebo bude možné tento

plánovač nastavit obecně, tzn. nastavit si den/dny a hodinu v týdnu, kdy se pokaždé zařízení zapne. Tuto funkci by mohli například ocenit řidiči v zimním období, kdy by s její pomocí mohlo být zapínáno topení vozidla.

4.8 Kontrola kreditu

Velmi značný problém u GSM alarmu představují SIM karty založené na kreditovém systému, kdy je nutné mít vždy nabitou určitou částku, aby bylo možné z daného čísla volat či odesílat SMS zprávy. Pokud by došlo k situaci, že není nabit dostatečný kredit a došlo by k odcizení vozidla, alarm by nebyl schopen upozornit majitele. U takových karet je tedy nutné vždy kredit kontrolovat, aby k této situaci nedošlo. Tuto kontrolu bude poskytovat i naše aplikace a v případě nízkého kreditu upozorní na tuto skutečnost majitele.

4.9 Přeposílání zpráv

Přestože všechny uvedené alarmy v kapitole č. 3 umožňovali nastavit až tři telefonní čísla, na které budou odesílány poplašné zprávy, některým uživatelům by se mohlo hodit, aby počet telefonních čísel nebyl nijak omezen nebo aby se zprávy přeposílaly raději z jiného telefonu, který má například výhodnější sazbu za odeslání SMS. Aplikace by tedy sama měla nabízet tuto možnost pro neomezený počet čísel a zároveň by měla umožnit vybrat, zda chceme přeposílat pouze poplašné zprávy nebo rovnou všechny příchozí SMS z daného alarmu.

5 Analýza a návrh

V této celé kapitole se budeme zabývat analýzou a návrhem aplikace. Získáme tak dobrý základ pro následnou implementaci. Před samotným návrhem je však nutné si nejprve zvolit konkrétní mobilní platformu, pro kterou budeme program vyvíjet. Pokud bychom totiž návrh tvořili bez tohoto rozhodnutí, mohlo by se stát, že by navržený systém nebylo možné na zvoleném operačním systému implementovat a museli bychom dodatečně provádět změny v návrhu. Touto volbou se tedy budeme zaobírat hned v první podkapitole.

5.1 Výběr mobilní platformy

Při výběru cílové platformy budeme vybírat především z nových mobilních operačních systémů. Vybraný systém však musí splňovat určité předpoklady, které vychází z podstaty námi implementované aplikace. Těmito předpoklady jsou:

1. Rozšířenost a životaschopnost platformy.
2. Škála služeb, které jsou s platformou propojeny (především propojení s mapovými podklady).
3. Snadný přístup k telefonním službám a k uloženým informacím uživatele (posílání a čtení SMS zpráv, přístup ke kontaktům).

Dle prvního předpokladu se při výběru můžeme zaměřit pouze na operační systém Android, iOS, Symbian a Windows Phone 7. Změny posledních dní ve strategii společnosti Nokia, kdy tato firma navázala spolupráci se společností Microsoft, však nasvědčují, že operační systém Symbian již nebude hlavní platformou pro mobilní zařízení této finské společnosti [26]. Nahradit by jej měla poslední zmíněná platforma, Windows Phone 7. Nebudeme tedy s operačním systémem Symbian při výběru nadále počítat a zaměříme se pouze na zbývající tři platformy.

Z pohledu druhého kritéria jsou platformy velmi podobné. Operační systém iOS je sice propojen hlavně se službami od společnosti Apple, nicméně u mapových podkladů je tomu jinak. K mapám lze přistupovat prostřednictvím MapKit Frameworku [27], který je přímo propojen s Google Maps a Google Earth API. Z tohoto pohledu poskytuje stejné možnosti jako OS Android, jehož hlavní síla vychází z propojení mobilu se službami od společnosti Google, tedy i on je napojen na tyto mapové služby. Windows Phone 7 z tohoto pohledu také nezaostává a umožňuje v aplikaci využít mapové podklady Bing Maps. Jelikož všechny tři operační systémy splňují druhý předpoklad, budeme s nimi počítat při posuzování dalšího předpokladu.

Poslední posuzované kritérium vychází ze způsobu komunikace s alarmem. Z textu uvedeného v kapitole č. 3 můžeme zjistit, že téměř všechny GSM alarmy komunikují především prostřednictvím textových SMS zpráv, případně pomocí volání. Je tedy nutné, aby platforma minimálně umožňovala programově zasílat a také číst příchozí SMS. Bez této podpory by nebylo možné aplikaci vytvořit. Co se týče zasílání textových zpráv, všechny tři platformy toto programově umožňují, nicméně se čtením příchozích zpráv je tomu jinak. Windows Phone 7 ani iOS v současné době neumožňují SMS zprávy v programech číst [28][29]. Jediný operační systém Android toto umožňuje. Aplikace na této platformě může být informována o přijetí SMS zprávy a na základě této události může získat jak telefonní číslo odesilatele, tak obsah zprávy. Krom toho, systém také umožňuje kdykoliv programově vyvolat telefonní hovor, Android je tak vhodný i pro případ, kdy komunikace mezi alarmem a uživatelem neprobíhá pouze skrze SMS zprávy, ale také prostřednictvím telefonního hovoru. Aplikaci tedy budeme vyvíjet pod tímto systémem. Nebudeme tak muset platit žádnou vývojářskou licenci ani vyvíjet program v neznámém vývojovém prostředí.

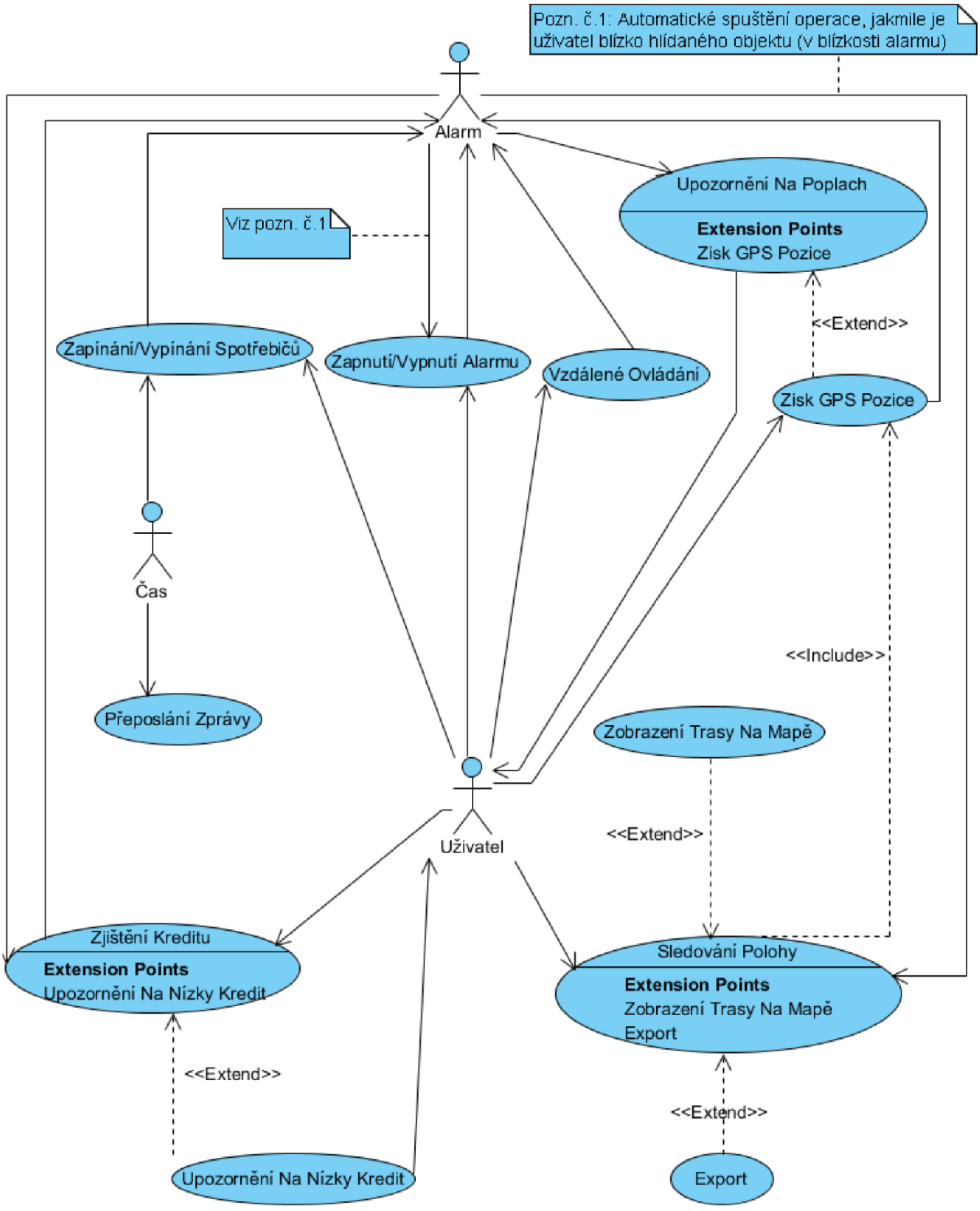
5.2 Případy užití

Pro zachycení požadavků kladených na systém a vymezení hranice systému se používají diagramy případu užití (use-case diagramy), které jsou součástí modelovacího jazyka UML [30]. Prostřednictvím tohoto diagramu jsme schopni vytvořit si první představu o tom, jakou funkčnost bude systém nabízet a kteří aktéři budou v systému figurovat. Celý use-case diagram popisující služby aplikace můžeme vidět na obrázku č. 6.

V aplikaci pro ovládání a vzdálenou správu alarmů budou figurovat pouze tři aktéři. Prvním z nich bude majitel mobilního telefonu (zkráceně uživatel), přes kterého bude alarm ovládán. Jelikož jako majitele budeme chápat uživatele, který je schopen ovládat telefon, nebude třeba řešit žádné vytváření uživatelských účtů ani následné přihlašování. Zabezpečení přístupu k aplikaci necháme na samotném telefonu, u kterého lze nastavit, aby při odemykání ověřil právoplatného uživatele, například heslem či gesty. Pokud uživatel touto autentizací neprojde, mobilní telefon nedovolí, aby mohla tato osoba ovládat mobilní přístroj, tedy ani ovládat nainstalované aplikace. S tímto zabezpečením tak nemůže dojít k nežádoucí ztrátě dat či k neoprávněnému ovládání alarmu.

Druhým aktérem tohoto systému bude samotný alarm, který jako externí zařízení nebude přímo pracovat s aplikací, nýbrž bude vyvolávat akce v závislosti na zaslané textové zprávě. Tohoto aktéra je nutné v diagramu zachytit, aby při implementaci funkce byly správně zachyceny vstupní podmínky akce.

Posledním aktérem v této aplikaci je čas, který představuje plánovač (časovač), na základě kterého jsou v aplikaci spuštěny určité činnosti. Tímto aktérem může být například vyvoláno zapnutí či vypnutí externích zařízení, které jsou připojeny k alarmu, nebo přeposlání příchozí SMS zprávy na zvolené číslo.



Obrázek 6: Use-case diagram aplikace.

5.2.1 Specifikace případu užití

Součástí každého diagramu případu užití by také měla být specifikace jednotlivých akcí. Významné případy užití si tedy nyní blíže popíšeme. V prvním případě se zaměříme na automatizované zapínaní a vypínání externích zařízení na základě uživatelem definovaného úkolu, který říká kdy a po jak dlouhou dobu má být připojený spotřebič spuštěn. Tento případ je podrobně uveden v tabulce č. 5. Dále se také budeme zabývat sledováním polohy vozidla (tabulka č. 6), při němž může nastat situace, kdy nelze zjistit polohu z mobilního přístroje (tabulka č. 7). Na tento stav musí být aplikace schopna adekvátním způsobem reagovat, aby tato služba vždy fungovala správně. V závěru si poté popíšeme, jak bude program postupovat při vyvolání poplachu. Tato činnost je popsána v tabulce č. 8.

Identifikátor	UC1	
Název	**Zapínání/Vypínání připojených spotřebičů**	
Popis	UC1 zapne a následně vypne daný připojený spotřebič.	
Primární aktéři	Čas	
Sekundární aktéři	Alarm	
Vstupní podmínky	1. Alarm umožňuje vzdáleně aktivovat/deaktivovat připojené zařízení. 2. Nastal čas, kdy se má zařízení spustit.	
Výstupní podmínky	1. Externí zařízení je zapnuto a po uplynutí zvolené doby opětovně vypnuto.	
Základní posloupnost	**Krok**	**Činnost**
	1	Časovač vyvolá událost, že se má zapnout externí zařízení.
	2a	Pokud alarm podporuje zapnutí externího zařízení na neomezenou dobu, pak:
	2a.1	Aplikace zašle zprávu pro zapnutí daného externího zařízení.
	2a.2	Jakmile uplyne doba, po kterou má být zařízení spuštěno, pak program zašle zprávu, která toto zařízení deaktivuje.
	2b	Alarm podporuje zapnutí externího zařízení na určitou dobu.
	2b.1	Program nastaví do zprávy čas s ohledem na maximální a minimální možnou hodnotu.
	2b.2	Pokud maximální hodnota je menší než doba, po kterou by mělo zařízení běžet, pak se akce 2b.1 opětovně provede po uplynutí času, po který bylo nakonec zařízení spuštěno. Doba dalšího běhu bude o tento čas zkrácena.
Alternativní posloupnost	**Krok**	**Činnost**
		Žádná
Výjimky		**Název**
		Žádné

Tabulka 5: Detail případu užití „Zapínání/Vypínání připojených spotřebičů na základě plánovače“.

Identifikátor	UC2	
Název	**Sledování polohy**	
Popis	UC2 zaznamená trasu pohybu sledovaného zařízení.	
Primární aktéři	Uživatel	
Sekundární aktéři	Alarm	
Vstupní podmínky	1. Alarm musí umět získávat GPS záznamy nebo mobilní přístroj musí disponovat GPS modulem. 2. Služba sledování polohy je aktivní.	
Výstupní podmínky	1. Je zaznamenána trasa pohybu vozidla.	
Základní posloupnost	**Krok**	**Činnost**
	1	Uživatel aktivuje sledování vozidla.
	2	Aplikace začne získávat GPS souřadnice podle navoleného časového intervalu buď z mobilního telefonu, nebo prostřednictvím alarmu. Všechny získané pozice jsou během akce ukládány.
	3	Uživatel deaktivuje sledování.
Alternativní posloupnost	**Krok**	**Činnost**
		Žádná
Výjimky		**Název**
		Nelze získat pozici z mobilního přístroje.

Tabulka 6: Detail případu užití „Sledování polohy“.

Identifikátor	UC2.E.1	
Název	**Sledování polohy: Nelze získat pozici z mobilního přístroje**	
Popis	Nelze získat GPS souřadnice, jelikož mobilní přístroj má deaktivován GPS modul.	
Vstupní podmínky	1. Nelze získat GPS pozici.	
Výstupní podmínky	1. Uživatel je upozorněn, že nelze získat pozici alarmu. 2. Operace sledování je pozastavena.	
Posloupnost	**Krok**	**Činnost**
	1	Aplikace zjistí, že nelze získat pozici z mobilního přístroje.
	2	Program ukončí sledování a upozorní na tuto skutečnost uživatele.

Tabulka 7: Výjimka, která nastane, když nelze získat GPS souřadnice z mobilního přístroje při sledování vozidla.

Identifikátor	UC3	
Název	**Upozornění na poplach**	
Popis	UC3 upozorní uživatele na vyvolaný poplach.	
Primární aktéři	Alarm	
Sekundární aktéři	Uživatel	
Vstupní podmínky	1. Alarm detekuje bezpečnostní incident a zasílá zprávu na mobilní telefon.	
Výstupní podmínky	1. Uživatel je upozorněn na poplach	
Základní posloupnost	**Krok**	**Činnost**
	1	Aplikace příjme poplašnou zprávu.
	2	Pokud ještě nebyl uživatel na tento poplach upozorněn, vyvolá se notifikace o tomto poplachu.
	3	Pokud SMS zpráva neposkytuje informace o aktuální pozici a alarm umožňuje získat GPS souřadnice, aplikace tuto polohu získá.
Alternativní posloupnost	**Krok**	**Činnost**
		Žádná
Výjimky		**Název**
		Žádné

Tabulka 8: Případ užití „Upozornění na poplach“.

5.3 Návrh aplikace

Základem každého úspěšného projektu je propracovaný návrh. Je třeba si dobře uvědomit, z jakých částí se aplikace skládá a jakou budou mít tyto části funkci. Při tomto logickém pohledu musíme již brát zřetel na cílovou platformu, jelikož mobilní platformy jsou značně odlišné a každá může nabízet různé prostředky a možnosti.

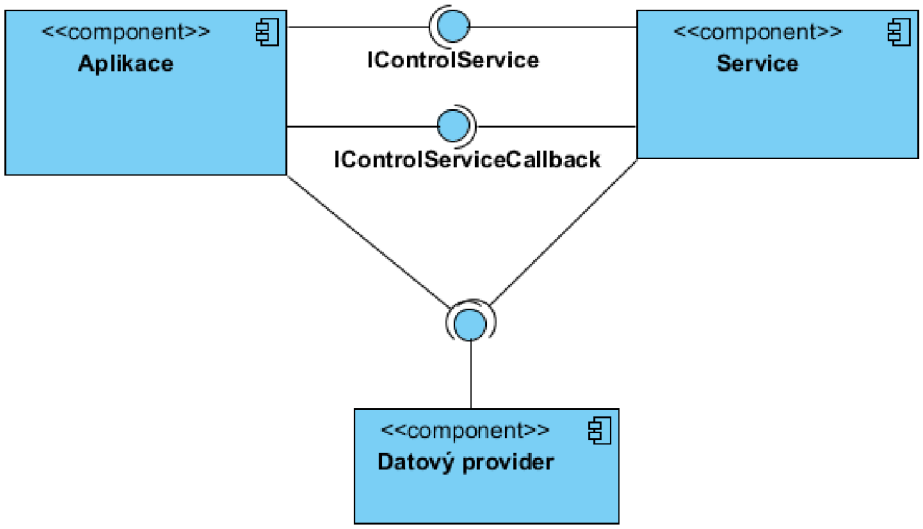
Aplikace vytvářené pod operačním systémem Android lze rozdělit na základní čtyři typy [31]:

- **Foreground aplikace** – Aplikace, která pracuje a vykonává svou funkci pouze tehdy, pokud běží v popředí a je zobrazena. V opačném případě je běh aplikace pozastaven. Tento typ aplikace má v systému nastavenu nejmenší prioritu (pokud není v popředí) a pokud je systém nucen uvolnit systémové prostředky, pak tento typ je často prvotním kandidátem, jenž bude uvolněn z paměti. Je proto nutné u dlouho neběžících aplikací ukládat aktuální stav, aby bylo možné jej obnovit při opakovaném spuštění.
- **Background aplikace** – Tento typ aplikace běží po většinu času v pozadí s minimální možností uživatelské interakce. Program reaguje na příchozí zprávy a události od systému, hardwaru či jiné běžící aplikace. Priorita je do značné míry větší než u předchozího typu, systémové prostředky procesu budou uvolněny pouze tehdy,

pokud již neexistuje žádná jiná možnost, jakým lze uvolnit prostředky pro běh aktivní aplikace na popředí.

- **Intermittent aplikace** – Jde o kombinaci předchozích dvou typů, kde většina užitečné činnosti je stále prováděná v pozadí (service), avšak služba komunikuje také s uživatelem a reaguje na jeho vstupy. Interakce uživatele je realizována klasickým způsobem, a to prostřednictvím uživatelského rozhraní (aktivity).
- **Widgets** – Posledním typem je widget, který slouží jako vizuální komponenta pro interakci s uživatelem, která může být umístěna na hlavní obrazovce. Tento typ je spíše doplňkem, nežli samotnou aplikací. Používá se především pro rychlou interakci a pro zobrazování důležitých informací, jako například zobrazení stavu baterie či aktuálního času.

Často bývá velmi obtížné program přímo zařadit pouze do jedné konkrétní skupiny, avšak v našem případě tomu tak není. Náš program musí neustále naslouchat příchozím SMS zprávám a ty zpracovávat a vyhodnocovat. Musíme zaručit, že běh aplikace nebude pozastaven při spuštění jiného programu, nebo při uvolňování systémových prostředků. Program tedy musí mít vysokou systémovou prioritu, a proto využijeme service, která tuto vlastnost má a bude činnost na pozadí zajišťovat. Jelikož aplikace bude závislá jak na komunikaci s bezpečnostním zařízením, tak i na interakci s uživatelem, musí rovněž poskytovat uživatelské rozhraní. K tomuto by se dalo využít widgetu, který však není vhodný pro zobrazování komplexnějších ovládacích prvků, ani pro zadávání většího množství dat. Využijeme tedy třetího typu, kombinaci aplikace běžící jak na popředí, tak i na pozadí. Tyto dvě části spolu budou komunikovat a zároveň sdílet společné úložiště dat. Schéma tohoto návrhu a nástin vzájemné komunikace je zobrazen v následujícím diagramu.



Obrázek 7: Návrh aplikace v podobě diagramu komponent.

5.3.1 Vymezení činností jednotlivých částí aplikace

Jelikož jsme si aplikaci rozdělili na dvě samotné části, je třeba si vymezit, jaké činnosti budou tyto prvky vykonávat. Začněme nejprve service, která poběží po celou dobu v pozadí.

Service bude tvořit jádro celé aplikace. Bude obstarávat jak logiku, tak i komunikaci s bezpečnostním zařízením. Tato komponenta tedy musí být schopna vytvořit SMS zprávu dle konfigurace a zaslat jí příslušnému alarmu. Naopak, pokud mobilní zařízení příjme textovou zprávu, musí být tato událost v service zachycena a zpracována. Pokud se zjistí, že odesilatelem zprávy je alarm, tak se obsah zprávy rozpozná dle uložených vzorů a v závislosti na konkrétním významu textu se provedou odpovídající činnosti. Pokud například bude detekován poplach, vyvolá se notifikace upozorňující na tuto událost.

Úkolem service bude také obstarávat veškerou funkčnost, kterou bude aplikace v kombinaci s možnostmi alarmu nabízet. Například, pokud bude uživatel chtít sledovat aktuální polohu vozidla, služba bude automaticky zasílat alarmu dotaz na polohu, kterou poté zpracuje a uloží do databáze. Dalším možným příkladem činnosti, kterou bude služba řídit, je automatické spuštění externích zařízení. Pokud bezpečnostní zařízení umožňuje vzdáleně ovládat připojené přístroje, budou službou automaticky zapínány a vypínány podle nastaveného plánu, který si uživatel sám navolí.

Grafická část aplikace bude sloužit pouze k interakci s uživatelem a bude provádět operace, které nebudou vázány na komunikaci s alarmem. Prostřednictvím této části bude moci uživatel povolovat jednotlivé služby a měnit jejich nastavení. O této změně bude informovat službu, která jeho požadavku přizpůsobí své chování. Úkolem klientské části bude také zobrazovat nasbíraná data, jako například zobrazit trasu vozidla na mapě.

5.4 MVC architektura

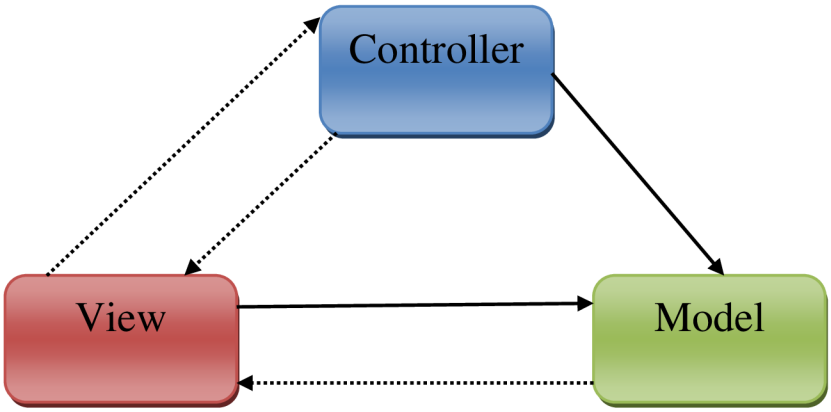
MVC[3] je novodobý, často používaný architektonický vzor, který se používá při tvorbě softwaru. Při využití tohoto návrhu dochází k rozdělení klasických vrstev uživatelského rozhraní, logiky a datového modelu do tří nezávislých částí. Tyto nově vytvořené části se nazývají *Model*, *View* a *Controller* a ve své podstatě obstarávají tuto činnost [32]:

- **Model** – Tato část odpovídá vrstvě domény. Obstarává logiku celé aplikace a provádí změnu v datech.
- **View** – Je vrstva, která je zodpovědná za prezentaci dat reprezentovaných modelem (například zobrazení dat ve formě dynamického HTML).

[3] Model-view-controller

- **Controller** – Je komponenta, která zpracovává a reaguje na události vyvolané interakcí uživatele. Tato část může jako jediná měnit stav jak objektů modelu, tak i objektů vrstvy View.

Rozdělení do těchto tří komponent má za následek, že při modifikaci libovolné části bude mít tato změna pouze minimální vliv na zbývající prvky. Toho je docíleno díky specifického trojúhelníkového propojení, které můžeme vidět na obrázku č. 8.

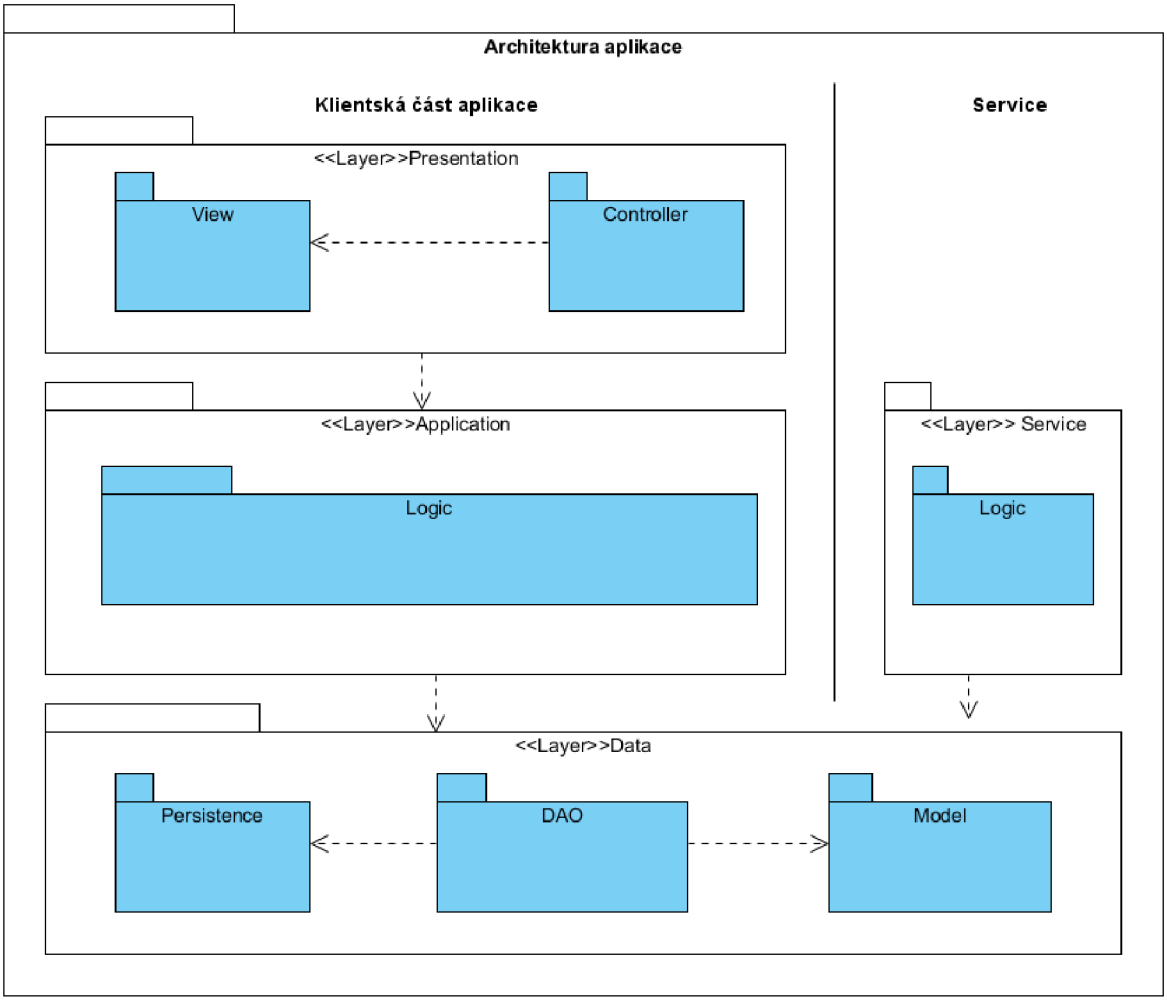


Obrázek 8: Architektura Model-View-Controller.

V tomto obrázku můžeme vidět dva typy hran. Hrany reprezentované plnou čarou, které značí přímé vazby (reference na konkrétní objekt), a hrany přerušované, které značí vazbu nepřímou. Přímé vazby musí být vždy dvě a to vazba ve směru *Controller-Model* a *View-Model*. Oproti tomu nepřímé vazby se mohou ve skutečnosti od obrázku lišit v závislosti na konkrétním způsobu realizace. Ve výsledku však musí být vždy uplatněn následující princip:

1. Řadič musí být informován o všech akcích, které uživatel provedl prostřednictvím uživatelského rozhraní (například kliknutí na tlačítko).
2. Controller reaguje na tuto událost. Pokud je třeba, vyvolá požadavek na změnu dat v modelu nebo provede změnu pohledu.
3. Pohled aktualizuje svůj stav, načte změněná data modelu a zobrazí je.

Tento architektonický vzor se budeme snažit aplikovat při návrhu a implementaci našeho programu. Jelikož se však program skládá ze dvou částí, které budou sdílet stejné úložiště dat, bude vhodné tento architektonický vzor využít pouze v rámci grafické části. Celou aplikaci navrhneme jako třívrstvou (viz obrázek č. 9).



Obrázek 9: Diagram znázorňující architekturu celé aplikace.

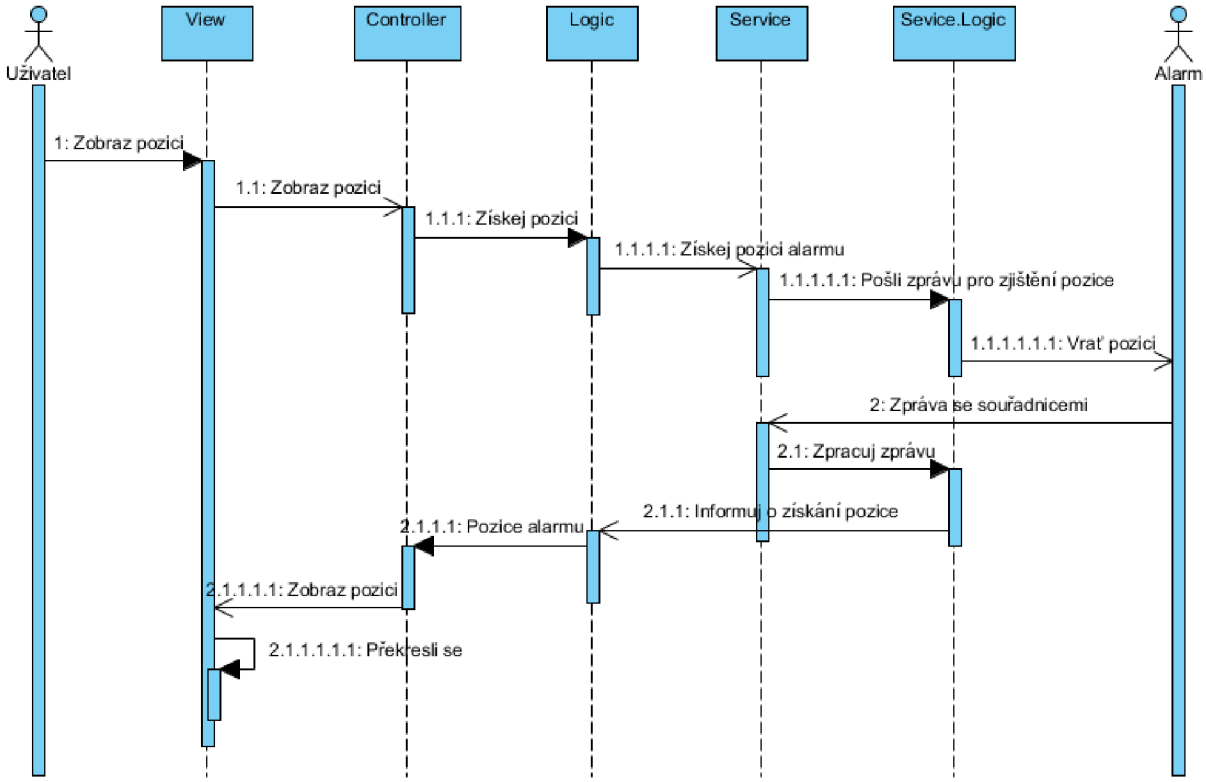
5.5 Komunikace

V této aplikaci bude docházet celkem ke třem různým typům komunikace. Grafická část aplikace bude komunikovat s klientem a jeho požadavky bude delegovat service. Service bude naopak komunikovat s alarmem prostřednictvím textových SMS zpráv a tyto informace bude předávat zpět do klientské části. Abychom tuto vzájemnou komunikaci lépe pochopili, vytvoříme sekvenční diagram, který bude znázorňovat tok zpráv v časové rovině při požadavku uživatele na polohu vozidla. Tento diagram je zobrazen na obrázku č. 10.

Jak můžeme vidět v diagramu, operaci získání polohy inicializuje uživatel prostřednictvím uživatelského rozhraní (například kliknutím na tlačítko). Tato událost podle návrhového vzoru MVC je delegována do controlleru, který jí zpracovává a prostřednictvím aplikační logiky zasílá požadavek na zjištění aktuální polohy vozidla do service. Komunikace mezi těmito dvěma částmi probíhá asynchronně, z tohoto důvodu nedojde k zablokování uživatelského rozhraní a uživatel tedy může dále s aplikací pracovat.

Když service obdrží požadavek od klientské části, předává jej ke zpracování vlastní logice, která zašle SMS zprávu s požadavkem na zjištění polohy alarmu. Komunikace mezi service a alarmem opět probíhá asynchronně, a tak ani zde nedojde k žádnému zablokování.

SMS zpráva s odpovědi je přijata v hlavní části service. Ta tuto zprávu předá k dalšímu zpracování, na základě kterého jsou získány GPS souřadnice, které jsou asynchronně předány do aplikace s uživatelským rozhraním. Zde je tato pozice uživateli zobrazena na mapě.



Obrázek 10: Sekvenční diagram znázorňující komunikaci při získávání pozice alarmu.

5.6 Uložení dat

Operační systém Android poskytuje celou řadu možností, jak lze ukládat persistentní data aplikace. Při výběru konkrétního úložiště záleží na požadavcích aplikace a na charakteru dat. Je důležité si uvědomit, zda jsou data privátní, nebo mohou být přístupny jiným aplikacím.

Pokud chceme, aby data byla přístupna i jiným aplikacím, můžeme využít externího úložiště pro ukládání souborů. K tomuto lze využit buď webové služby, se kterou budeme komunikovat přes síťové připojení, nebo vyměnitelného média, nejčastěji SD karty. Všechna data uložená tímto způsobem jsou poté přístupná všem aplikacím a všem uživatelům.

Pokud by ukládaná data neměla být přístupna ostatním aplikacím, můžeme například využít tzv. *Shared Preferences*. Toto úložiště poskytuje obecný rámec, který umožňuje ukládat primitivní

datové typy `boolean`, `float`, `int`, `long` a `string` v podobě asociativního pole, kde každá hodnota je identifikována jednoznačným klíčem (textovým řetězcem). Takto uložené hodnoty jsou přístupny i po ukončení programu.

Druhým způsobem je využít interní paměti zařízení, která pracuje obdobně jako výše popsané externí úložiště, kde jsou však data přístupná pouze a jen z aplikace, která tato data uložila. Takto uložená data jsou po odinstalování aplikace smazána. Není tedy třeba řešit jejich mazání.

Poslední možností, jak lze data v aplikaci privátně uchovávat, je využití vestavěné relační databáze *SQLite*. Ta je stejně jako interní úložiště přístupna pouze mateřské aplikaci. Tato malá databáze implementuje téměř celý standard SQL-92 a od verze 3.6.19 [33] poskytuje podporu cizích klíčů a s nimi související akce, jako například `ON DELETE CASCADE`. Pro uchovávání dat aplikace budeme využívat této databáze. [34]

5.6.1 Návrh databáze

Jelikož jsme si zvolili uchovávat data v databázi, je třeba navrhnout její schéma. Prezentovat jej budeme formou ER diagramu v notaci UML. Protože půjde relativně o malou aplikaci, databáze se bude skládat pouze z devíti databázových tabulek. Tyto tabulky jsou blíže popsány v následujícím textu, kde je vždy uveden účel této tabulky následován seznamem jejich atributů. Celé schéma databáze je zobrazeno v obrázku č. 11.

Tabulka „alarm“

Tato databázová tabulka slouží pro uchovávání informací týkajících se jednotlivých typů bezpečnostních systémů (alarmů), které aplikace podporuje a uživatel je může tedy prostřednictvím programu ovládat. Primárním klíčem této tabulky je atribut `_id`, jenž může nabývat celočíselných hodnot a je generován automaticky databázovým systémem. Dalším významným atributem je sloupec s názvem `name`, jehož hodnota musí být vždy unikátní. Tato hodnota by tedy mohla nahradit dosavadní primární klíč, nicméně pro vyhledávání a spojování tabulek je lepší zanechat primární klíč celočíselný.

- **_id** – Primární klíč záznamu.
- **name** – Název bezpečnostního zařízení. Hodnota musí být unikátní a musí být vždy zadána.
- **vendor** – Textový řetězec obsahující název výrobce alarmu, hodnota musí být zadána.
- **description** – Textové pole, které obsahuje popis alarmu, jeho možnosti a případnou nápovědu pro ovládání prostřednictvím programu.

- **devices_count** – Tato vlastnost uchovává informaci o počtu externích zařízení, které jsou na alarm napojena.
- **alert_msg_with_location** – Příznak, který nabývá hodnoty *true*, pokud poplašná zpráva obsahuje informace o aktuální pozici hlídaného objektu. Pokud hodnota bude *false*, aplikace bude automaticky zjišťovat pozici, pokud alarm zjišťování polohy podporuje.
- **device_min_time** – Minimální doba, po kterou může být spuštěno externí zařízení.
- **device_max_time** – Maximální hodnota času, po kterou může být spuštěno externí zařízení.
- **device_time_unit** – Časová jednotka pro spuštění externího zařízení. Toto pole může nabývat hodnot *milisec*, *sec*, *min* a *hour* reprezentující časové jednotky od milisekundy až po hodinu.

Tabulka „action“

Entitní množina reprezentující příchozí a odchozí komunikaci bezpečnostního systému. S alarmem lze komunikovat buď prostřednictvím SMS nebo pomocí volání. Konkrétní typ je určen příznakem s názvem `calling`.

- **_id** – Celočíselný primární klíč.
- **alarm_id** – Cizí klíč, který propojuje tabulku s konkrétním alarmem (se záznamem v tabulce `alarm`).
- **name** – Unikátní název akce v rámci daného alarmu. Některé názvy jsou vyhrazeny a slouží k rozpoznání známé operace, kterou aplikace využívá k provádění specifické činnosti.
- **label** – Jednoduchý popisek, který bude uživateli zobrazen při ovládání alarmu.
- **info** – Informace popisující účel akce. Tato informace bude zobrazena uživateli současně s položkou `label`.
- **xml** – Pokud jde o komunikaci prostřednictvím SMS, pak tato položka bude obsahovat XML vzor textu SMS zprávy (viz kapitoly č. 5.7.2 a 5.7.3).
- **calling** – Příznak značící zda jde o komunikaci přes SMS zprávu či o komunikaci prostřednictvím telefonního hovoru.
- **incoming** – Hodnota tohoto příznaku určuje, zda jde o příkaz nebo o odpověď alarmu.

Tabulka „device“

V této tabulce jsou uchovávána data nesoucí informaci o výstupních portech daného alarmu, na které mohou být připojena externí zařízení.

- **_id** – Primární klíč tabulky.
- **alarm_id** – Cizí klíč odkazující se na konkrétní bezpečnostní zařízení.
- **key** – Klíč, na základě kterého alarm rozeznává konkrétní výstup. Tato hodnota musí být často obsažena v SMS příkazu.

Tabulka „user_alarm“

Tabulka `user_alarm` je nejdůležitější a také největší entitní množina této databáze. Informace obsažené v ní reprezentují alarm uživatele, který vychází ze zvoleného typy bezpečnostního zařízení (záznamu uloženého v tabulce `alarm`). Záznamy jsou identifikovány podle primárního klíče s názvem `_id`, nicméně pro vyhledávání lze použít i atributy `name` či `phone`, které musí být unikátní v rámci celé tabulky.

- **_id** – Celočíselný primární klíč tabulky.
- **alarm_id** – Cizí klíč, který se odkazuje na zvolenou konfiguraci bezpečnostního zařízení, která je uložena v tabulce `alarm`.
- **name** – Unikátní název, který bude v aplikaci použit pro identifikaci tohoto alarmu.
- **phone** – Telefonní číslo GSM alarmu.
- **passwd** – Heslo, které slouží pro rozpoznání uživatele při komunikaci s alarmem. Tento údaj musí být uveden v textu téměř všech příkazových SMS.
- **alarm_on** – Příznak, který uchovává aktuální stav alarmu (alarm zapnut/vypnut). Implicitně je nastavena hodnota „vypnuto“ (*false*).
- **alert_activated** – Logická hodnota, která signalizuje, zda došlo k vyvolání poplachu z důvodu porušení ochrany hlídaného objektu.
- **alert_time_interval** – Tato hodnota udává časový interval v milisekundách, který slouží pro automatické získávání polohy hlídaného objektu, pokud GPS souřadnice nejsou součástí poplašné zprávy.
- **log_book_enable** – Tato hodnota uchovává informaci o tom, zda je služba pro tvorbu knihy jízd aktivována. Implicitně je tato služba deaktivována.
- **log_book_on** – Atribut, který signalizuje, zda je v současné době zaznamenávána pozice objektu pro tvorbu knihy jízd.

- **log_book_timeinterval** – Čas v milisekundách, který udává periodu získávání polohy vozidla při zaznamenávání jeho trasy.
- **log_book_mobile_GPS** – Pokud bude hodnota tohoto atributu *true*, bude se trasa zaznamenávat prostřednictvím GPS modulu v mobilu.
- **log_book_bluetooth_device_on** – V případě, že bude tento příznak nastaven, bude služba „Získávání trasy vozidla" spuštěna automaticky po připojení telefonu k zvolenému bluetooth zařízení.
- **bluetooth_device_name** – Název bluetooth zařízení pro automatickou aktivaci služeb.
- **bluetooth_device_mac** – MAC adresa bluetooth zařízení pro automatickou aktivaci služeb.
- **check_credit_on** – Příznak aktivace služby „Automatická kontrola kreditu".
- **phone_for_credit** – Telefonní číslo pro vyvolání aktuální hodnoty kreditu.
- **credit** – Aktuální hodnota kreditu GSM alarmu.
- **credit_limit** – Minimální hodnota kreditu, při jehož překročení bude uživatel upozorněn.
- **price_for_sms** – Cena jedné odeslané SMS zprávy z GSM alarmu.
- **automatic_managment_on** – Atribut povolení služby automatického zamykání a odemykání vozu.
- **automatic_managment_mode** – Mód, který je použit při automatickém zamykání a odemykání vozu.
- **time_forwarding** – Čas uložený v milisekundách, který udává dobu, po kterou musí uživatel zareagovat na poplašnou zprávu, jinak dojde k přeposlání zprávy na číslo uvedené v atributu `phone_number_forwarding`.
- **phone_number_forwarding** – Telefonní číslo pro přeposlání poplašné zprávy.
- **export_delete_trace** – Pokud je tato hodnota nastavena, budou se všechny mazané trasy exportovat a ukládat na SD kartu.
- **export_file_type** – Hodnota, která reprezentuje formát souboru, ve kterém budou vyexportovány trasy.
- **period_delete_trace** – Hodnota udávající dobu v měsících, po kterou budou uchovávány záznamy tras v databázi.

Tabulka „useralarm_devices"

Tato tabulka, podobně jako tabulka `devices`, nese informace o výstupních portech alarmů. Data jsou však vázána na uživatelský alarm, nikoliv na záznamy reprezentující fyzické zařízení.

Uživatel si tedy může nadefinovat různá nastavení pro alarmy stejného typu, aniž by se tyto konfigurace určitým způsobem ovlivňovaly.

- **_id** – Identifikátor tabulky.
- **user_alarm_id** – Reference na záznam alarmu uživatele.
- **key** – Hodnota klíče vycházející ze záznamu uloženého v tabulce `devices`. Tato redundance je zavedena z důvodu optimalizace přístupu. Jelikož hodnota je po celou dobu neměnná, nemůže dojít k porušení konzistentního stavu databáze.
- **name** – Název zařízení, který se bude zobrazovat v aplikaci uživateli.

Tabulka „device_task“

Protože aplikace bude poskytovat možnost zapnout vybraná externí zařízení v konkrétním čase po pevně zvolenou dobu, musíme si uchovávat data tohoto nastavení. K tomuto účelu bude sloužit právě tato tabulka. Záznamy v ní uložené ponesou informaci o čase, délce trvání a periodě, kterou můžeme navolit přes dny v týdny. Například, pokud uživatel bude chtít, aby zařízení bylo spuštěno každý den v 8:00, bude muset označit všechny dny.

- **_id** – Primární klíč, který je jediným unikátním záznamem v celé tabulce. Pro vyhledávání konkrétního záznamu je tedy nutné přistupovat právě přes tuto položku.
- **device_id** – Identifikátor zařízení, k němuž se úkol váže.
- **start_time** – Startovací čas v milisekundách.
- **time_duration** – Doba, po kterou bude zařízení spuštěné.
- **monday, tuesday, wednesday, thursday, friday, saturday, sunday** – Jednotlivé dny v týdnu, prostřednictvím kterých lze nastavit opakování činnosti.
- **active** – Příznak, který indikuje, že tato úloha je aktivní a má se spustit v uvedeném čase. Pokud je úloha neaktivní, nabývá atribut hodnoty *false*.

Tabulka „related_phone_number“

Pro uchovávání telefonních čísel, na která se mají přeposílat vybrané zprávy z alarmu, nám poslouží tabulka s názvem `related_phone_number`.

- **_id** – Primární klíč tabulky.
- **user_alarm_id** – Alarm, jehož zprávy se mají přeposílat na toto číslo.
- **phone_number** – Telefonní číslo pro přeposílání zpráv.
- **name** – Pojmenování záznamu.

Tabulka „trace“

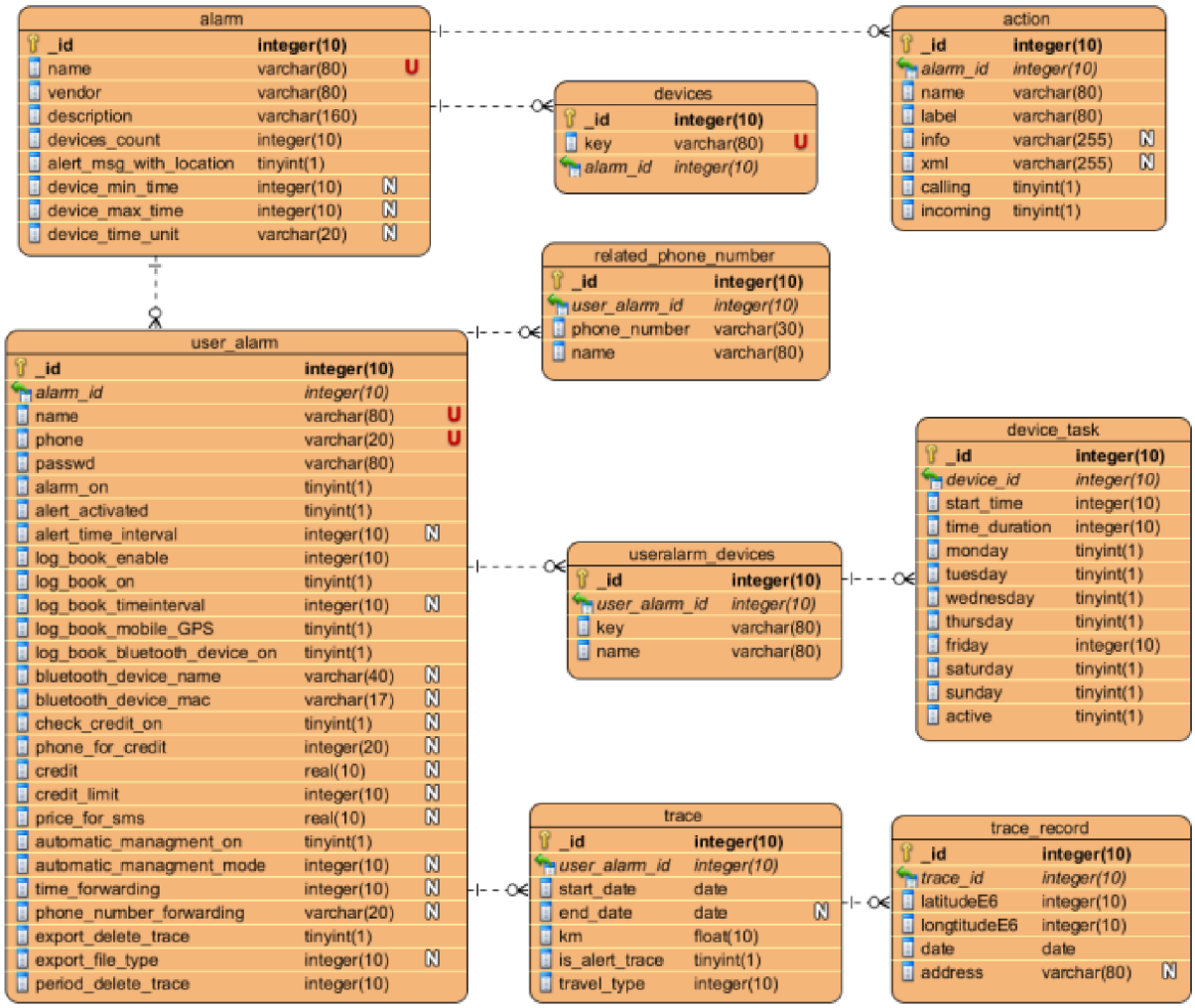
Pokud bezpečnostní zařízení poskytuje informace o aktuální pozici, mohou být tyto souřadnice v pravidelných intervalech získávány a uchovávány jako trasy. Ty mohou posloužit buď jako kniha jízd, nebo jako sledovací prostředek kontroly polohy vozu. V tabulce jsou také uloženy všechny pozice, které byly získány během bezpečnostního incidentu. Uživatel má tedy možnost vystopovat své ukradené vozidlo.

- **_id** – Identifikátor tabulky.
- **user_alarm_id** – Alarm, na který se tato trasa váže.
- **start_date** – Datová známka vytvoření záznamu.
- **end_date** – Datum a čas ukončení sledování.
- **km** – Počet ujetých kilometrů v metrech.
- **is_alert_trace** – Logická hodnota, která určuje, zda jde o trasu během poplachu, nebo o trasu pro knihu jízd.
- **travel_type** – Tato položka slouží pro odlišení soukromých a pracovních cest.

Tabulka „trace_record“

Poslední tabulka této databáze slouží pro uchovávání souřadnic trasy.

- **_id** – Celočíselný primární klíč tabulky.
- **trace_id** – Cizí klíč odkazující se na trasu, do níž tato souřadnice patří.
- **latitudeE6** – Zeměpisná šířka vynásobená konstantou $1*10^6$ a uložená v datovém typu `integer`.
- **longtitudeE6** - Zeměpisná výška vynásobená konstantou $1*10^6$.
- **date** – Datum pořízení záznamu.
- **address** – Adresa zeměpisného bodu.



Obrázek 11: ER diagram databáze.

5.7 Obecnost systému

Jak již bylo uvedeno ve specifikaci, systém musí být schopen komunikovat s různými typy GSM alarmů. Jak jsme se však mohli dočíst v kapitole č. 3, alarmy nemají žádné standardizované rozhraní, prostřednictvím kterého by komunikovaly. Každý typ alarmu je jedinečný a používá vlastní komunikační protokol. Je tedy třeba vymyslet způsob, kterým popíšeme alarm aplikaci tak, aby poskytovala pouze a jen funkce, které alarm podporuje, a aby program uměla s alarmem komunikovat, tzn. používat jeho specifický komunikační protokol.

5.7.1 OOP a návrhový vzor „Adaptér“

Aby aplikace nebyla závislá na konkrétním alarmu, lze v objektově orientovaných programovacích jazycích využít rozhraní (interface) v kombinaci s návrhovým vzorem *Adaptér*. Tento vzor se používá jako prostředník mezi prostředím, které požaduje určité rozhraní, a třídou, která toto požadované rozhraní neposkytuje. V našem případě budeme chápat jako prostředí naší mobilní aplikaci a třídu bude reprezentovat konkrétní alarm. Dle návrhového vzoru je nutné nadefinovat obecné rozhraní, které bude obsahovat metody napříč všemi podporovanými funkcemi, a třídy pro všechny alarmy, které budou toto rozhraní implementovat. Metody, které nejsou alarmem podporovány, by byly prázdné anebo by vyvolaly příslušnou výjimku. Takovýmto způsobem by aplikace přesně věděla, které funkce jsou podporovány a které naopak ne.

Nevýhodou tohoto řešení je, že je potřeba vytvořit specifickou třídu pro všechny alarmy, což je však velmi obtížné. Na trhu se neustále objevují nové alarmy, bylo by nutné řešit, jakým způsobem tyto třídy dynamicky začlenit do aplikace, aby nebylo nutné ji opětovně kompilovat či komprimovat do spustitelného balíčku. Krom toho, většina uživatelů nemá dostatečné znalosti, aby byli schopni tuto novou třídu vytvořit. Bylo by nezbytné navázat spolupráci s výrobci a přimět je tyto třídy vytvářet, na což by však většina nemusela přistoupit. Aplikace by proto nebyla přístupná pro všechny typy alarmů, bylo by tak značně omezeno její použití.

5.7.2 XML

Snažili jsme se vymyslet způsob, který by umožnil i běžnému uživateli nadefinovat chování nového alarmu a snadno jej tak začlenit mezi alarmy, které jsou již aplikací podporovány. Začali jsme proto přemýšlet nad textovou specifikací alarmu, konkrétně nad specifikací zapsanou v XML, které umožňuje zapsat informace různého typu ve strukturované podobě, která je vhodná jak pro strojové zpracování, tak i pro přímé psaní rukou.

Naše představa je taková, že aplikace bude svázána s jedním či více XML konfiguračními soubory, které budou pomocí různých elementů a atributů specifikovat vlastnosti alarmu a jeho chování. Součástí této definice bude také popis jednotlivých SMS zpráv, kterými lze alarm vzdáleně konfigurovat a prostřednictvím kterých alarm komunikuje s uživatelem. Pro zpracování této konfigurace bude použito klasických nástrojů, které jsou pro práci s XML soubory poskytovány v programovacích jazycích. V úvahu připadalo i využití XLST transformace, která umožňuje na základě jednoduchých pravidel převést XML do jiné podoby, jako například do HTML nebo rovnou vygenerovat prostý text. Toto by bylo vhodné především pro generování SMS zpráv přímo z konfiguračního souboru, nicméně opačně zpracovat příchozí text a převést jej podle XML takto nelze. Proto bylo nakonec od tohoto upuštěno a ke zpracování bude využito pouze konečného automatu a klasického XML parseru.

V následujícím kusu kódu je zobrazena ukázka XML zápisu, jak by v tomto formátu mohla být zapsána zpráva pro uložení telefonních čísel, která budou sloužit pro zasílání upozornění z alarmu. Tento zápis koresponduje s příkazovou zprávou patřící alarmu GPS LOC (viz kapitola č. 3.3).

```
<message name="set_phone_number" label="Nastavení telefonních čísel"
         info="SMS pro nastavení telefonních čísel pro alarm">
     <word name="passwd" label="Heslo"/>
     <char value=" "/>
     <text>PHONE</text>
     <char value=" "/>
     <complex name="prvni_tel_cislo" label="První telefonní číslo">
          <phone name="cislo1" label="Telefonní číslo"/>
          <select required="0" name="volbaCislo1" label="Zabezpečení"
              info="Vyberte pro jake zabezpečení má být číslo aplikováno">
              <option info="Input alarm">I</option>
              <option info="GPS alarm">G</option>
          </select>
          <char value=" "/>
      </complex>
      <complex name="druhe_tel_cislo" label="Druhé telefonní číslo"
               required="0">
          <phone name="cislo2" label="Telefonní číslo"/>
          <select required="0" name="volbaCislo2" label="Zabezpečení"
              info="Vyberte pro jake zabezpečení má být číslo aplikováno">
              <option info="Input alarm">I</option>
              <option info="GPS alarm">G</option>
          </select>
          <char value=" "/>
      </complex>
      <complex name="treti_tel_cislo" label="Třetí telefonní číslo"
               required="0">
          <phone name="cislo3" label="Telefonní číslo"/>
          <select required="0" name="volbaCislo3" label="Zabezpečení"
              info="Vyberte pro jake zabezpečení má být číslo aplikováno">
              <option info="Input alarm">I</option>
              <option info="GPS alarm">G</option>
          </select>
          <char value=" "/>
       </complex>
</message>
```

Zdrojový kód 1: XML zápis příkazové zprávy pro uložení telefonních čísel alarmu GPS LOC.

Jak můžeme vyčíst ze zdrojového kódu, SMS zpráva je tvořena kořenovým elementem `message`, jejíž parametr `name` určuje, o jaký typ zprávy se jedná. V našem případě nabývá hodnoty `set_phone_number`, která značí, že jde o zprávu pro nastavení telefonních čísel alarmu. Tento kořenový element se poté skládá z dalších elementů, které tvoří jednotlivé části zprávy. Ty budou v aplikaci postupně nahrazovány skutečnými hodnotami, jako například heslem nebo telefonním číslem. Jak můžeme vidět, tak XML definuje, že lze uložit maximálně tři telefonní kontakty. Ty mohou být doplněny o volitelný parametr, který značí, zda má být odeslána zpráva při input alarmu, nebo až při alarmu, který je vyvolán pohybem vozidla.

5.7.3 XML schéma

Na základě komunikačních protokolů alarmů uvedených v kapitole č. 3 bylo vytvořeno XML schéma, které stanovuje formát konfiguračního XML souboru. Při tvorbě tohoto schématu jsme se snažili, aby bylo co nejvíce obecné, aby bylo možné prostřednictvím něho zapsat libovolný tvar SMS zprávy či zapsat, že komunikace probíhá přes telefonní hovor. Toto schéma je blíže popsáno v následujících odstavcích. Celková podoba XML schématu je součástí přílohy.

Definice vlastního datového typu pro atributy

Přestože samotný jazyk pro XML schéma definuje celou řadu jednoduchých i komplexních datových typů, v některých případech je třeba vytvořit si vlastní, který přiřadíme jednomu či více atributům, popřípadě elementům. V našem případě bude nutné vytvořit datový typ reprezentující jeden znak (`CHARACTER`) a výčtový typ pro čas, který bude definovat časové jednotky, ve kterých bude zadáván časový údaj. Povolenými časovými jednotkami budou milisekundy (*milisec*), sekundy (*sec*), minuty (*min*) a hodiny (*hour*). Definice těchto nových typů má následující tvar:

```
<xs:simpleType name="character">
      <xs:restriction base="xs:string">
            <xs:length value="1"/>
      </xs:restriction>
</xs:simpleType>
```

Zdrojový kód 2: Definice datového typu CHARACTER.

```
<xs:simpleType name="time_units">
      <xs:restriction base="xs:string">
            <xs:enumeration value="milisec"/>
            <xs:enumeration value="sec"/>
            <xs:enumeration value="min"/>
            <xs:enumeration value="hour"/>
      </xs:restriction>
</xs:simpleType>
```

Zdrojový kód 3: Definice datového typu s názvem TIME_UNITS.

Význam a popis atributů

Elementy, jejichž definice je popsána dále v textu, obsahují často atributy, které můžou být společné. Abychom tyto vlastnosti nemuseli neustále v textu popisovat, uvedeme si jejich popis již teď v následující tabulce.

Název	Datový typ	Popis
alert_with_location	xs:boolean	Příznak alarmu, který říká, zda poplašná zpráva obsahuje aktuální polohu.
info	xs:string	Dodatečný text, který do větších podrobností popisuje význam elementu.
label	xs:string	Popisek, který slouží pro stručnou charakteristiku elementu.
max	xs:unsignedInt	Atribut numerických prvků, který udává maximální povolenou hodnotu.
min	xs:unsignedInt	Atribut numerických prvků, který udává nejmenší povolenou hodnotu.
name	xs:string	Název elementů, který slouží k jejich identifikaci.
required	xs:boolean	Atribut, který určuje, zda při zpracování SMS musí být element v textu přítomen. Implicitně je tato hodnota nastavena na TRUE.
separator	xs:string	Atribut elementu FLOAT, který udává separátor desetinné části. Implicitně je jako separátor brán znak tečka ('.').
unit	time_units	Časová jednota reprezentující milisekundu, sekundu, minutu či hodinu. Defaultní časovou hodnotou je milisekunda.
value	character	Očekávaná hodnota elementu. Tato hodnota se vyskytuje pouze u elementu CHAR.
vendor	xs:string	Povinný atribut elementu ALARM, který obsahuje jméno výrobce alarmu.

Tabulka 9: Soupis všech atributů, které se mohou vyskytovat v elementech.

Definice jednoduchých elementů

V této části si nadefinujeme základní elementy, které nemohou obsahovat žádný jiný vnořený element, pouze obsahují atributy, které je blíže specifikují. Tyto elementy jsou navrženy co nejvíce obecně a kopírují podobu základních datových typů, se kterými se můžeme setkat v různých programovacích jazycích.

Nejjednodušším elementem konfiguračního XML souboru je element s názvem `CHAR`, který reprezentuje jeden libovolný či vybraný znak. Hodnota znaku, který je očekáván, může být zadán v atributu `value`.

```
<xs:element name="char">
      <xs:complexType>
            <xs:attribute name="value" type="character "/>
      </xs:complexType>
</xs:element>
```

Zdrojový kód 4: Element CHAR reprezentující jeden znak.

Druhým nejjednodušším elementem je `TEXT`, který je datového typu `xs:string` a nemá žádné atributy. Tento prvek poslouží k označení neměnného textu, jehož význam je irelevantní a můžeme jej při zpracování SMS zprávy klidně zahodit. Pokud však tento text nebude ve zprávě nalezen, bude zpracování ukončeno a vyhodnoceno jako nekorespondující s aktuálně zpracovanou konfigurací.

```
<xs:element name="text" type="xs:string"/>
```

Zdrojový kód 5: Definice základního elementu s názvem TEXT.

Dalším elementem, který reprezentuje textový řetězec, je `WORD`. Pokud se tento prvek objeví v konfiguraci, bude z SMS zprávy načteno jedno slovo obsahující znaky abecedy A-Z (malá i velká písmena), znaky podtržítko (‘_‘) nebo pomlčka (‘-’). Jelikož se očekává, že toto slovo bude mít určitý svůj význam, musí tento prvek obsahovat atributy `name` a `label`. Vlastnosti `info` a `required` jsou u tohoto elementu volitelné.

```
<xs:element name="word">
      <xs:complexType>
            <xs:attribute name="name" type="xs:string" use="required"/>
            <xs:attribute name="label" type="xs:string" use="required"/>
            <xs:attribute name="info" type="xs:string"/>
            <xs:attribute name="required" type="xs:boolean" default="1"/>
      </xs:complexType>
</xs:element>
```

Zdrojový kód 6: Definice prvku WORD.

Po uvedení základních elementů reprezentující text, si nyní můžeme představit elementy, které reprezentují číselné posloupnosti. Těmi jsou `NUMBER`, `FLOAT` a `TIME`, jež mají společné atributy `name`, `label`, `info`, `min`, `max` a `required`.

Prvky NUMBER a FLOAT mají stejný význam jako v klasických programovacích jazycích, proto je není třeba blíže specifikovat. Oproti tomu prvek `TIME` se v našem pojetí liší. Reprezentuje pouze jednu numerickou hodnotu, jež je zadána ve vybrané časové jednotce, která je specifikována v atributu `unit`.

```
<xs:element name="number">
      <xs:complexType>
            <xs:attribute name="name" type="xs:string" use="required"/>
            <xs:attribute name="label" type="xs:string" use="required"/>
            <xs:attribute name="info" type="xs:string"/>
            <xs:attribute name="min" type="xs:unsignedInt"/>
            <xs:attribute name="max" type="xs:unsignedInt"/>
            <xs:attribute name="required" type="xs:boolean" default="1"/>
      </xs:complexType>
</xs:element>
```

Zdrojový kód 7: Element NUMBER.

```
<xs:element name="float">
      <xs:complexType>
            <xs:attribute name="name" type="xs:string" use="required"/>
            <xs:attribute name="label" type="xs:string" use="required"/>
            <xs:attribute name="info" type="xs:string"/>
            <xs:attribute name="separator" type="xs:string" default="."/>
            <xs:attribute name="min" type="xs:unsignedInt"/>
            <xs:attribute name="max" type="xs:unsignedInt"/>
            <xs:attribute name="required" type="xs:boolean" default="1"/>
      </xs:complexType>
</xs:element>
```

Zdrojový kód 8: Element FLOAT.

```
<xs:element name="time">
      <xs:complexType>
            <xs:attribute name="name" type="xs:string" use="required"/>
            <xs:attribute name="label" type="xs:string" use="required"/>
            <xs:attribute name="info" type="xs:string"/>
            <xs:attribute name="unit" type="time_units" default="sec"/>
            <xs:attribute name="min" type="xs:unsignedInt"/>
            <xs:attribute name="max" type="xs:unsignedInt"/>
            <xs:attribute name="required" type="xs:boolean" default="1"/>
      </xs:complexType>
</xs:element>
```

Zdrojový kód 9: Element TIME.

Posledními dvěma prvky, které neobsahují žádné potomky, jsou `PHONE` a `OPTION`. Jak název prvního elementu napovídá, prvek reprezentuje telefonní číslo. Druhý zmíněný element pak reprezentuje jednu volbu při výběru 1 z N.

```
<xs:element name="phone">
      <xs:complexType>
            <xs:attribute name="name" type="xs:string" use="required"/>
            <xs:attribute name="label" type="xs:string" use="required"/>
            <xs:attribute name="info" type="xs:string"/>
            <xs:attribute name="required" type="xs:boolean" default="1"/>
      </xs:complexType>
</xs:element>
```

Zdrojový kód 10: Definice prvku PHONE.

```
<xs:element name="option">
      <xs:complexType>
            <xs:simpleContent>
                  <xs:extension base="xs:string">
                        <xs:attribute name="name" type="xs:string"/>
                        <xs:attribute name="info" type="xs:string"/>
                  </xs:extension>
            </xs:simpleContent>
      </xs:complexType>
</xs:element>
```

Zdrojový kód 11: Definice volby OPTION pro element SELECT.

Definice komplexních elementů

Od základních prvků se nyní dostáváme k elementům komplexním, které obsahují jak atributy, tak i další vnořené elementy. Začněme prvkem, který reprezentuje výběr jedné hodnoty z více nabízených (`SELECT`). Touto nabízenou hodnotou může být slovo reprezentované elementem `OPTION`, který má své specifické jméno, které musí být v rámci celého výběru unikátní. Tato podmínka je ve schématu zapsána pomocí klíčového elementu `xs:unique` a pomocí výrazu v jazyce xpath. Schéma této části vypadá následně:

```
<xs:element name="select">
   <xs:complexType>
      <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="1" ref="option"/>
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" use="required"/>
      <xs:attribute name="label" type="xs:string" use="required"/>
      <xs:attribute name="info" type="xs:string"/>
      <xs:attribute name="required" type="xs:boolean" default="1"/>
   </xs:complexType>
   <xs:unique name="UniqueOptionName">
      <xs:selector xpath="./option" />
      <xs:field xpath="@name" />
   </xs:unique>
</xs:element>
```

Zdrojový kód 12: Definice elementu SELECT.

Dalším složeným elementem je `COMPLEX`, který funguje obdobně jako kontejner v programovacím jazyku. `COMPLEX` může obsahovat vnořené prvky, jenž uskupuje do určité skupiny, která může mít nastaven atribut `required`. Jestliže je tento atribut nastaven na hodnotu *false*, pak pokud alespoň jeden vnořený prvek není v textu obsažen a zároveň je tento prvek označen jako povinný, je celý blok ohraničený tímto elementem zahozen a nepoužit. Ukázka použití tohoto elementu je uvedena ve zdrojovém kódu č. 1.

Doposud jsme nadefinovali pouze prvky, které udávají sémantický význam jednotlivých částí textu SMS zprávy. Zbývají nám tedy vytvořit prvky, které je budou zabalovat a definovat vlastnosti a chování samotného alarmu. Začneme nejprve elementy, které definují komunikaci s alarmem. Těmi jsou `MESSAGE`, reprezentující komunikaci prostřednictvím SMS, a `CALLING`, který signalizuje, že komunikace bude probíhat přes telefonní hovor. Obě tyto možnosti se mohou vyskytovat v části popisující příchozí komunikaci (element `INCOMING_ACTIONS`).

V zápisu elementu reprezentující komunikaci přes textovou zprávu si můžeme všimnout, že je zde nastavené omezení, které hlídá unikátnost atributu `name` v rámci všech prvků (i vnořených v kontejneru `COMPLEX`). Tato podmínka musí být splněna, jelikož hodnoty získané ze zprávy musí být jednoznačně identifikovatelné a musí být možné je zpětně dosadit, pokud se jedná o příkazovou

SMS. Některé názvy jsou vyhrazené a mají specifický význam. Tyto hodnoty jsou uvedeny na závěr této kapitoly v tabulce č. 10.

```
<xs:element name="message">
   <xs:complexType>
      <xs:sequence maxOccurs="unbounded" minOccurs="0">
            <xs:choice>
                  <xs:element maxOccurs="unbounded" ref="word"/>
                  <xs:element maxOccurs="unbounded" ref="text"/>
                  <xs:element maxOccurs="unbounded" ref="char"/>
                  <xs:element maxOccurs="unbounded" ref="number"/>
                  <xs:element maxOccurs="unbounded" ref="float"/>
                  <xs:element maxOccurs="unbounded" ref="time"/>
                  <xs:element maxOccurs="unbounded" ref="phone"/>
                  <xs:element maxOccurs="unbounded" ref="select"/>
                  <xs:element maxOccurs="unbounded" ref="complex"/>
            </xs:choice>
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" use="required"/>
      <xs:attribute name="label" type="xs:string" use="required"/>
      <xs:attribute name="info" type="xs:string" use="required"/>
   </xs:complexType>
   <xs:unique name="UniqueMessageItemName">
      <xs:selector xpath="./* | ./complex/*" />
      <xs:field xpath="@name" />
  </xs:unique>
</xs:element>
```

Zdrojový kód 13: Element MESSAGE reprezentující komunikaci přes SMS.

```
<xs:element name="calling">
      <xs:complexType>
            <xs:attribute name="name" type="xs:string" use="required"/>
            <xs:attribute name="label" type="xs:string" use="required"/>
            <xs:attribute name="info" type="xs:string" use="required"/>
      </xs:complexType>
</xs:element>
```

Zdrojový kód 14: Tag CALLING signalizující komunikaci přes telefonní hovor.

Dostáváme se pomalu k závěru a tedy k definici kořenového elementu, který popisuje celý alarm. Ten se skládá ze čtyř po sobě jdoucích přímých potomků, které obsahují popis alarmu (`DESCRIPTION`), výpis externích zařízení (`DEVICES`) a v neposlední řadě popis příchozí (`INCOMING_ACTIONS`) a odchozí (`OUTGOING_ACTIONS`) komunikace z pohledu tohoto zařízení. U komunikace je třeba upozornit na omezení, které vyžaduje unikátní název akcí. Tento název poslouží k rozpoznání jednotlivých zpráv od sebe a také k identifikaci zpráv, které aplikace potřebuje pro vykonávání své činnosti. Tyto klíčové názvy jsou sepsány v tabulce č. 11.

```
<xs:element name="incoming_actions">
      <xs:complexType>
         <xs:sequence>
            <xs:sequence maxOccurs="unbounded" minOccurs="0">
                  <xs:choice>
                        <xs:element maxOccurs="unbounded" ref="message"/>
                        <xs:element maxOccurs="unbounded" ref="calling"/>
                  </xs:choice>
            </xs:sequence>
         </xs:sequence>
      </xs:complexType>
      <xs:unique name="UniqueIncomingActionsName">
            <xs:selector xpath="./*" />
            <xs:field xpath="@name" />
      </xs:unique>
</xs:element>
```

Zdrojový kód 15: Definice příchozí komunikace z pohledu alarmu.

```
<xs:element name="outgoing_actions">
      <xs:complexType>
            <xs:sequence>
                  <xs:element maxOccurs="unbounded" ref="message"/>
            </xs:sequence>
      </xs:complexType>
      <xs:unique name="UniqueOutgoingActionsName">
            <xs:selector xpath="./*" />
            <xs:field xpath="@name" />
      </xs:unique>
</xs:element>
```

Zdrojový kód 16: Definice odchozí komunikace z pohledu alarmu.

```
<xs:element name="alarm">
   <xs:complexType>
      <xs:sequence>
            <xs:element ref="description"/>
            <xs:element ref="devices"/>
            <xs:element ref="incoming_actions"/>
            <xs:element ref="outgoing_actions"/>
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" use="required"/>
      <xs:attribute name="vendor" type="xs:string" use="required"/>
      <xs:attribute name="alert_with_location" type="xs:boolean"
                    default="0"/>
   </xs:complexType>
</xs:element>
```

Zdrojový kód 17: Kořenový element s názvem alarm.

..

Název	Element	Význam
credit	**number, float**	Aktuální hodnota kreditu.
day	**number**	Den v měsíci (1-31).
device	**word, number**	Hodnotou tohoto elementu bude identifikátor externího zařízení.
device_time	**number, time**	Čas pro spuštění/vypnutí externího zařízení.
gmt_offset_hour	**number**	Element reprezentuje hodinový posun oproti GMT času.
gmt_offset_min	**number**	Element reprezentuje minutový posun oproti GMT času.
gmt_offset_minus	**option**	Značka reprezentuje záporný GMT offset.
gmt_offset_orientation	**select**	Výběr 1:N pro zjištění kladného či záporného časového posunu.
gmt_offset_plus	**option**	Značka reprezentuje kladný GMT offset.
hour	**number**	Počet hodin dne v časové známce (0-23).
latitude	**float**	Zeměpisná šířka zapsaná jedním desetinným číslem (pouze ve stupních).
latitude_degree	**number, float**	Hodnota stupňů zeměpisné šířky.
latitude_min	**number, float**	Hodnota minut zeměpisné šířky.
latitude_north	**option**	Orientace zeměpisné šířky (Sever).
latitude_orientation	**select**	Orientace zeměpisné šířky.
latitude_sec	**number, float**	Hodnota sekund zeměpisné šířky.
latitude_south	**option**	Orientace zeměpisné šířky (Jih).
longtitude	**float**	Zeměpisná výška zapsaná jedním desetinným číslem (pouze ve stupních).
longtitude_degree	**number, float**	Hodnota stupňů zeměpisné výšky.
longtitude_east	**option**	Orientace zeměpisné výšky (Východ).
longtitude_min	**number, float**	Hodnota minut zeměpisné výšky.
longtitude_orientation	**select**	Orientace zeměpisné výšky.
longtitude_sec	**number, float**	Hodnota sekund zeměpisné výšky.
longtitude_west	**option**	Orientace zeměpisné výšky (Západ).
min	**number**	Hodnota udávající počet minut (0-59).
month	**number**	Celočíselná hodnota reprezentující měsíc v roce (1-12).
passwd	**word**	Element reprezentuje heslo pro ovládání alarmu.
phone_for_credit	**phone**	Telefonní číslo pro zjištění kreditu.
sec	**number**	Počet sekund minuty (0-59).
year	**number**	Rok ve formátu YYYY nebo zkráceně YY.

Tabulka 10: Soupis všech vyhrazených názvů pro elementy tvořící zprávu.

Název	Typ	Význam
alarm_off	**příkazová**	Zpráva pro přepnutí alarmu do stavu OFF.
alarm_on	**příkazová**	Zpráva pro přepnutí alarmu do stavu ON.
alarm_start	**příkazová**	Spuštění ochrany vozidla.
alarm_stop	**příkazová**	Vypnutí ochrany vozidla.
credit	**příkazová**	Zpráva pro zisk kreditu.
credit_info	**informační**	Zpráva obsahující aktuální hodnotu kreditu.
device_off	**příkazová**	Vypnutí externího zařízení.
device_on	**příkazová**	Zapnutí externího zařízení na neomezenou dobu.
device_time_on	**příkazová**	Zapnutí externího zařízení na určitou dobu.
change_password	**příkazová**	Zpráva pro změnu hesla.
change_position_alert	**informační**	Poplašná zpráva vyvolána pohybem vozidla.
input_alert	**informační**	Zpráva informující o neoprávněném vniknutí.
position	**příkazová**	Zpráva pro získání aktuální pozice vozidla.
position_info	**informační**	Zpráva nesoucí informaci o aktuální pozici.
reset	**příkazová**	Reset do továrního nastavení.

Tabulka 11: Vyhrazené názvy pro komunikaci s alarmem.

6 Implementace

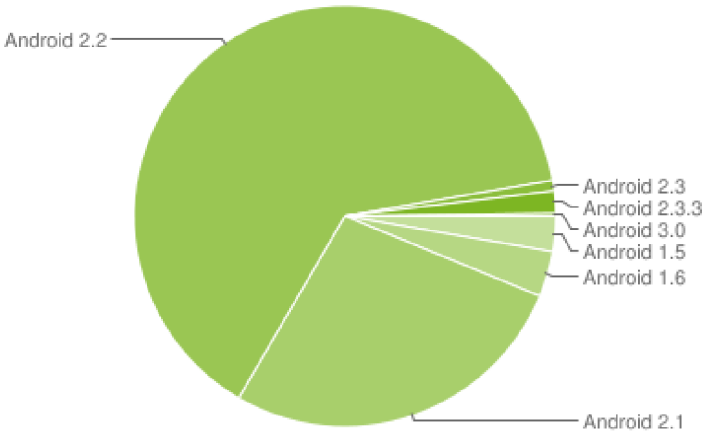
V této kapitole si blíže popíšeme implementaci námi navržené aplikace. Zaměříme se především na důležité aspekty tvorby tohoto programu, jako například na způsob, jakým byla implementována service, nebo jakým způsobem je zpracována komunikace mezi aplikací a alarmem. Než se však dostaneme k těmto konkrétním věcem, musíme si nejprve zvolit, pro jakou verzi SDK budeme tuto aplikaci vyvíjet.

6.1 Verze SDK

V současné době se operační systém Android vyskytuje v mnoha provedeních. Poměr zastoupení jednotlivých variant je zobrazen v obrázku č. 12, který zachycuje stav k datu 1.4.2011. Tato data vycházejí ze statistik Android Marketu, který tyto údaje sledoval po dobu 14 dní. [35]

Jak si můžeme všimnout z grafu, nejvíce zastoupenou verzí tohoto operačního systému je Android 2.1 a Android 2.2. Při výběru SDK se omezíme právě na tyto dvě varianty, abychom umožnili aplikaci používat co největšímu počtu uživatelů. Při konečném rozhodování sehrála důležitou roli knihovna `javax.xml.transform`, jež umožňuje provádět transformace mezi různými XML reprezentacemi (například DOM). Knihovnu použijeme pro generování XML, z tohoto důvodu budeme aplikaci vyvíjet pro Android 2.2 (API Level 8), protože až tato verze touto knihovnou disponuje.

Přestože se může zdát, že tímto rozhodnutím ztratíme téměř jednu čtvrtinu potencionálních uživatelů, nakonec tato ztráta nemusí být až tak veliká. V současné době lze operační systém bezplatně aktualizovat. Pokud uživatel se starším systémem bude chtít tuto aplikaci používat, stačí si jen zaktualizovat současnou platformu na námi požadovanou či vyšší verzi OS.



Obrázek 12: Statistika zastoupení jednotlivých verzí operačního systému Android na trhu s mobilními zařízeními k datu 1.4.2011 [35].

6.2 Service

Android SDK umožňuje implementovat service vícero způsoby. Nejčastěji používaným způsobem je implementace tzv. *Local Service*. Tento způsob umožňuje vytvořit komponentu programu, která běží současně s jinými částmi, avšak stále ve stejném procesu. Z tohoto důvodu je komunikace mezi klientem a komponentou velmi jednoduchá. Stačí pouze prostřednictvím objektu `IBinder` získat objekt služby a nad touto instancí volat jeho veřejné metody. Při této implementaci není třeba řešit žádnou mezi-procesovou komunikaci. Tento způsob však také přináší určitá omezení. Jelikož služba běží pod stejným procesem jako aplikace, která jí vytvořila, služba zaniká současně s aplikací.

Z tohoto důvodu je možné vytvořit tzv. *Remote Service*, tedy službu, která běží samostatně ve svém vlastním procesu. Komunikace v tomto případě probíhá mezi různými dvěma procesy. Protože obecně nelze přistupovat k paměti z jednoho procesu do paměti procesu jiného, je třeba všechny objekty nejprve dekomponovat na primitivní datové typy, kterým operační systém rozumí, a následně je opětovně složit na straně druhého procesu. Pro tento účel poskytuje Android SDK speciální nástroj AIDL[4], který toto obstarává. Služba nadefinuje veřejné rozhraní ve speciálním souboru s příponou `aidl`, na základě kterého je nástrojem vygenerována stub třída, která má nadefinované abstraktní metody tohoto rozhraní. Třídu je třeba v service vytvořit a dodefinovat chování všech metod. Klient objekt získá návratovou hodnotou metody `onBind()` a od této chvíle jej smí bezproblému využívat pro vzdálené volání metod služby.

Remote Service odpovídá všem požadavkům, které naše aplikace vyžaduje. Hlavní výhodou tohoto přístupu je, že service poběží i po ukončení grafické aplikace, což je pro naši službu nezbytné. Proto ji vytvoříme tímto způsobem.

6.2.1 Komunikace s klientem

Jak bylo řečeno v předešlé kapitole, aby mohl klient se službou komunikovat, je třeba nadefinovat rozhraní pro mezi-procesovou komunikaci. AIDL pro definování tohoto rozhraní umožňuje použít všechny primitivní datové typy z programovacího jazyka JAVA (`boolean`, `char`, `int`, `double`, …) a vybrané třídy, u kterých sám obstarává jejich dekompozici a jejich následné složení. Těmito třídami jsou `String`, `CharSequence`, `List` a `Map`. U posledních dvou zmíněných kolekcí musí být prvky datového typu, které AIDL podporuje, anebo datového typu, které implementuje rozhraní `Parcelable`. Pokud toto rozhraní naimplementujeme ve vlastní třídě, je možné tuto třídu v definici použít i mimo tyto kolekce.

V následujícím kódu je zobrazeno rozhraní, které bude pro komunikaci využívat námi implementovaná service. Jednotlivé metody jsou opatřeny komentáři, takže není třeba je blíže

[4] Android Interface Definition Language

specifikovat. Jediné, co je třeba vysvětlit, je klíčové slovo `oneway`, které je uvedeno ve všech metodách. Tento identifikátor mění chování při vzdálené komunikaci. Pokud není uveden, pak komunikace je prováděna synchronně a klient je po celou dobu vzdáleného zpracování blokován. V našem případě, kdy většina operací má pouze notifikační charakter, je čekání na provedení operace zcela zbytečné a proto komunikaci označíme jako jednosměrnou. V takovémto případě klient zašle data a okamžitě se navrátí k původní činnosti. Abychom však v tomto případě byli schopni komunikovat i opačným směrem, tzn. od service ke klientovi, zavedeme také rozhraní pro tuto komunikaci. Tento interface si však již blíže nebudeme popisovat, v případě zájmu bych odkázal na soubor `IControlServiceCallback.aidl`, umístěný ve složce se zdrojovým kódem programu.

```
/**
 * Metoda, která slouží pro registraci Callback klienta.
 * @param cb Callback klient
 */
oneway void registerCallback(IControlServiceCallback cb);

/**
 * Metoda, která slouží pro odregistraci Callback klienta.
 * @param cb Callback klient
 */
oneway void unregisterCallback(IControlServiceCallback cb);

/**
 * Metoda, která informuje o změně dat objektu alarmu.
 * @param idUserAlarm Identifikátor uživatelského alarmu
 */
oneway void changeUserAlarm(int idUserAlarm);
/**
 * Metoda, která informuje o smazání konkrétního alarmu.
 * @param idUserAlarm Identifikátor uživatelského alarmu
 */
oneway void deleteUserAlarm(int idUserAlarm);

/**
 * Metoda, která informuje o změně dat objektu třídy DeviceTaskData.
 * @param idUserAlarm Identifikátor uživatelského alarmu
 * @param taskId Identifikátor device task
 */
oneway void changeDeviceTask(int idUserAlarm, int taskId);

/**
 * Metoda, která informuje o smazání device úkolu.
 * @param idUserAlarm Identifikátor uživatelského alarmu
 * @param taskId Identifikátor device task
 */
oneway void deleteDeviceTask(int idUserAlarm, int taskId);

/**
 * Metoda, která požaduje zisk pozice vozidla.
 * @param idUserAlarm Identifikátor uživatelského alarmu
 */
oneway void getAlarmPosition(int idUserAlarm);
```

```
    /**
     * Metoda, která požaduje zisk aktuální hodnoty kreditu.
     * @param idUserAlarm Identifikátor uživatelského alarmu
     */
    oneway void getAlarmCredit(int idUserAlarm);

    /**
     * Požadavek na vypnutí poplachu.
     * @param idUserAlarm Identifikátor uživatelského alarmu
     */
    oneway void stopAlert(int idUserAlarm);

    /**
     * Potvrzení převzetí ALERT NOTIFIKACE uživatelem.
     * @param idUserAlarm Identifikátor uživatelského alarmu
     */
    oneway void pickUpAlertNotification(int idUserAlarm);

   /**
     * Metoda, která vyvolá provedení příkazové akce pro zvolený alarm.
     * @param idUserAlarm Identifikátor uživatelského alarmu
     * @param idAction Identifikátor akce
     * @param data Data, které se mají do příkazu zakomponovat
     */
    oneway void doAction(int idUserAlarm, int idAction,in
                         TransmittedAlarmData data);
```

Zdrojový kód 18: Interface pro vzdálenou komunikaci se service.

6.2.2 Komunikace s alarmem

Důležitou činností, kterou bude rovněž obstarávat kód v pozadí, je komunikace s alarmem. Ve většině případů půjde o komunikaci formou SMS zpráv, jejichž vzor je popsán v XML konfiguračním souboru (viz kapitola 5.7.3). Při generování či zpracování textových zpráv bude nutné využít některý z XML parseru. Android SDK nabízí celou řadu možností, jakým způsobem lze zpracovávat XML [36]. Můžeme například využít často používaný DOM[5], který funguje stejně jako v programovacím jazyku JAVA. Tento parser při zpracování nejprve načte celý obsah do paměti a vytvoří si z něj stromovou strukturu. K tomuto uskupení můžeme přistupovat prostřednictvím DOM API, které nám umožňuje získat libovolný element z celého dokumentu. Tento způsob zpracování je však velmi paměťově náročný a především v mobilních aplikacích, kde operační paměti je všeobecně nedostatek, může tento způsob vést k určitým problémům. Pokud i přesto bychom jej chtěli použít, musíme se nejprve ujistit, že zpracovávané dokumenty nebudou velmi obsáhlé.

Paměťovou náročnost DOM přístupu se snaží řešit SAX[6]. Jde o způsob, který postupně prochází všechny elementy a prostřednictvím handleru o těchto elementech informuje uživatelský kód. V programu musíme vytvořit vlastní handler, který je odvozen z bázové třídy `org.xml.sax.helpers.DefaultHandler`. Tato třída definuje základní metody,

[5] Document Object Model
[6] Simple API for XML

které jsou volány pro různé typy elementů. Pro počáteční značku dokumentu je volána metoda `startDocument()`, pro startovací značku metoda `startElement()` a pro koncovou `endElement()`. Jelikož elementy mohou obsahovat také vnořené textové řetězce (textové uzly), pro tato data je volána metoda `characters()`. Přestože SAX implementace představuje velmi rychlý a paměťově nenáročný způsob zpracování dokumentu, pro naše zpracování vzorů zpráv tento přístup není vhodný. SAX parser totiž vždy zpracuje celý dokument, nelze jej při zpracování pozastavit. U zpracování příchozí SMS zprávy, kdy se snažíme mapováním nalézt příslušný vzor, bychom byli nuceni projít celé XML, přestože bychom již věděli, že vzor neodpovídá textu zprávy. Tento způsob by zbytečně prodlužoval nalezení vzoru a zpracování textu SMS.

Proto v naší aplikaci využijeme raději XML pull parser, který pracuje podobně jako StAX[7]. Tento přístup umožňuje ve zdrojovém kódu řídit postupný průchod všemi elementy a tedy kdykoliv ukončit toto zpracovávání. V našem kódu budeme ve `while` cyklu procházet všechny značky dokud nedojdeme až do konce, nebo dokud logika nedojde do stavu `FAIL`. Jelikož XML pull parser přiřazuje typům elementů celočíselné konstanty, uvnitř cyklu využijeme `switch` konstrukci pro větvení, která bude obdobně jako v SAX handler volat metody pro počáteční a koncovou značku. Ukázka průchodu je zobrazena ve zdrojovém kódu č. 19.

Konkrétní postup, jakým budou jednotlivé příchozí a odchozí SMS zprávy zpracovány pomocí XML parseru, je popsán v následujících kapitolách.

```
                    …
int eventType = parser.getEventType();

while (eventType != XmlPullParser.END_DOCUMENT && !done){

    switch (eventType){
        case XmlPullParser.START_DOCUMENT:
            break;
        case XmlPullParser.START_TAG:
                startElement();
        case XmlPullParser.END_TAG:
                endElement();
    }

    eventType = parser.next();
}
```

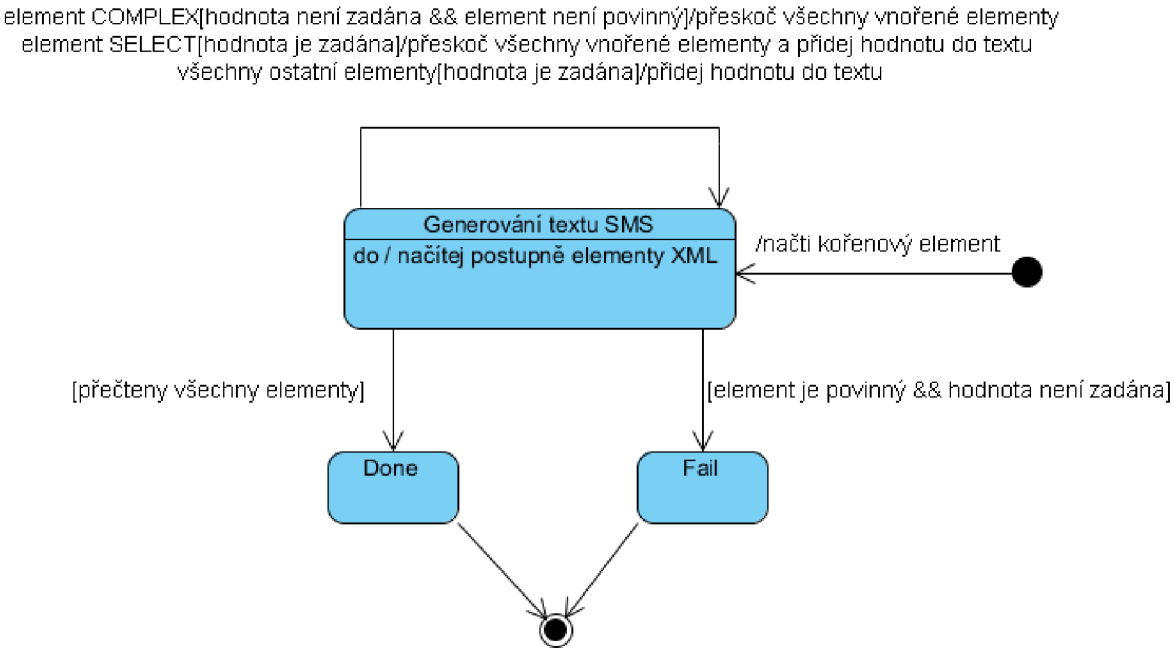
Zdrojový kód 19: Ukázka průchodu všemi elementy pomocí XML pull parseru.

Generování příkazové SMS

Pro odeslání příkazové zprávy je nutné XML šablonu projít a na základě definovaných značek postupně generovat zprávu. Tento vygenerovaný text musí obsahovat zadané hodnoty parametrů. Postup generování příkazové SMS znázorňuje diagram v obrázku č. 13. Jak můžeme vyčíst z popisu, u všech načtených elementů je požadována hodnota, která by měla být obsažena v objektu třídy

[7] Streaming API for XML

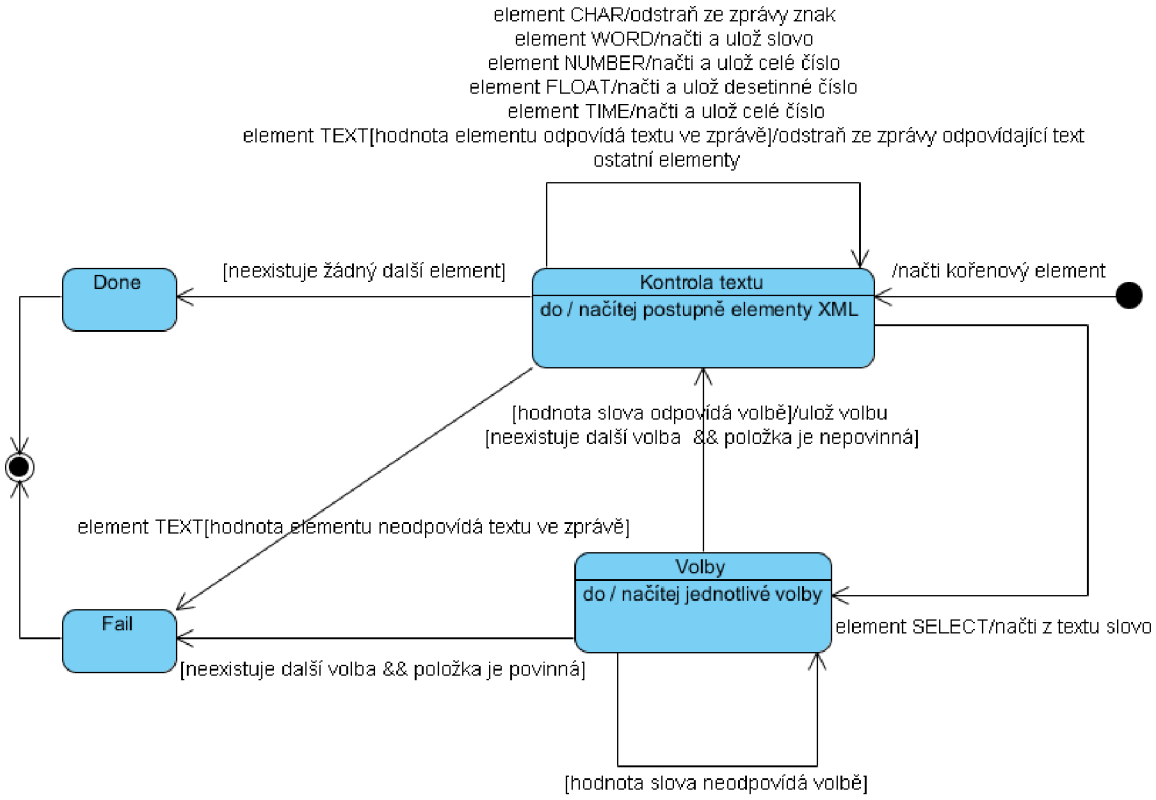
`TransmittedAlarmData`, který je třeba zadat při vytváření instance parseru. Pokud je hodnota zadána, tak je přidána do generovaného textu. V opačném případě je provedena kontrola, zda hodnota není povinná. Pokud ano, tak parser přechází do stavu `FAIL` a další elementy již nezpracovává.



Obrázek 13: Stavový diagram popisující generování textu SMS.

Zpracování příchozí SMS

Příchozí zpráva je zpracována analogicky, avšak v obráceném sledu. Text zprávy je postupně načítán po částech, které jsou rozpoznány podle typu příslušné značky (`NUMBER`, `FLOAT`, `WORD`,…). Této načtené části je přiřazen sémantický význam a hodnota informace, kterou část reprezentuje, je uložena do pomocné datové struktury. Pokud element reprezentuje výběr 1 z N, jsou parserem zkontrolovány možné volby a porovnány s načteným slovem. Pokud slovo neodpovídá žádné volbě a zároveň element má nastaven atribut `required` na `true`, pak parser přechází do stavu `FAIL`, kde dojde k ukončení procesu zpracování. Celý proces načtení příchozí SMS zprávy je zobrazen v obrázku č. 14.



Obrázek 14: Stavový diagram popisující zpracování příchozí SMS.

6.3 Klient

Jak bylo uvedeno v kapitole č. 5.3.1, grafická část aplikace nebude mít na starosti téměř žádné operace. Klient bude sloužit pouze pro zobrazování dat a k provádění změn nastavení. Proto se v této kapitole budeme věnovat především rozhraní a implementaci této části podle návrhového vzoru MVC.

6.3.1 Implementace MVC

Přestože MVC architektura se začíná stále častěji promítat do všech novodobých platforem, vývoj aplikací pro systém Android je ve značné míře specifický a nelze na něj tento architektonický vzor bez menších kompromisů aplikovat.

V Android aplikacích bývá uživatelské rozhraní reprezentováno XML soubory, které specifikují jednotlivé prvky a jejich atributy. Tyto specifikační soubory uložené ve složce layout („`res/layout`“) budeme považovat za vrstvu View.

Controllery poté budou tvořit tzv. *aktivity*, které reprezentují obrazovku obsahující uživatelského rozhraní. V těchto třídách budeme načítat jednotlivé prvky z XML souboru a budeme zde odchytávat jejich události. Aktivity tedy musí implementovat potřebné rozhraní, které jim toto chování umožní. Nejčastějším rozhraním, které bude třeba implementovat, bude rozhraní `OnClickListener()`

z SDK balíčku `android.view.View`, které nám umožní reagovat na kliknutí uživatele na daný ovládací prvek.

Většina akcí uživatele bude mít za následek změnu hodnot dat, aktivita tedy bude muset komunikovat s modelem. Proto je nezbytné, aby v aktivitě byly vytvořeny objekty, které budou pracovat s daty a budou je měnit. Nejvhodnějším místem, kde je dobré tyto objekty vytvářet, je metoda `onCreate()`, která je volána vždy při vytvoření aktivity v paměti. Bude tak zaručeno, že data bude možné okamžitě měnit po vyvolání obrazovky v programu.

Protože velká část dat je společná pro více aktivit, docházelo by při přechodu z jedné obrazovky do druhé k opakovanému načítání hodnot z databáze a změny provedené v aktuální obrazovce by se neprojevily v těch předešlých. Z tohoto důvodu budou objekty modelu, jejichž data budou sdílena více aktivitami, implementovány dle návrhového vzoru *Singleton*. Data tak budou načtená pouze jednou a změny se projeví vždy a všude najednou.

Zbývá vyřešit, jakým způsobem budeme provádět aktualizaci pohledu. Jelikož uživatelské rozhraní je popsáno značkovacím jazykem XML, nelze z této vrstvy aktualizovaná data získat. K této činnosti potřebujeme prostředníka, který bude informován o změně dat, data načte a zobrazí. Tímto prostředníkem bude opět aktivita, jelikož již má vazbu na prvky View a tedy nedojde k vytvoření nové závislosti na tuto vrstvu. Bude však třeba vytvořit nový kanál, přes který bude aktivita informována modelem, že došlo ke změně. K tomu využije dalšího návrhového vzoru, *Observer*.

Observer řeší problém, kdy vícero různých objektů (pozorovatelé) se zajímají o změnu stavu či události jiného objektu (vydavatele), přičemž nechceme, aby sledovaný objekt byl s těmito pozorovateli přímo svázán. Tohoto stavu se docílí tak, že je vytvořeno obecné rozhraní pro pozorovatele, přes které jsou objekty informovány o změně. Sledovaný objekt pouze musí umožnit pozorovatelům se dynamicky registrovat a při změně dat je informovat. Knihovna `java.util` poskytuje předpřipravené rozhraní a třídu, prostřednictvím kterých lze toto chování snadno naimplementovat. Stačí pouze, aby třída modelu byla odvozená z třídy `Observable` a aktivita implementovala rozhraní `Observer`.

6.3.2 Zpracování konfigurace alarmu

Úkolem klientské části aplikace, kromě zobrazování dat, bude také zpracování konfigurace alarmu. Tato konfigurace, jak je popsáno v kapitole č. 5.7, bude zapsána v XML souboru. Pro toto zpracování si tedy budeme muset opět vybrat jeden z XML parseru. Než se k tomuto výběru dostaneme, řekněme si nejdříve postup, jakým budou data zpracována a kde budou uložena.

Pokud uživatel bude chtít naimportovat novou konfiguraci, musí umístit konfigurační soubor do složky `AlarmController/config` umístěného na SD kartě. Pouze z této lokace bude možné soubor s nastavením do aplikace načíst. Aby nebylo třeba mít konfiguraci alarmu neustále k dispozici

a při potřebě získat z ní data ji opětovně procházet a zpracovávat, bude konfigurace načtená pouze jednou a získané informace se budou pro další použití ukládat do databáze. K ukládání informací poslouží databázová tabulka `alarm`. Pokud si však vzpomeneme na kapitolu, kde je popsán způsob zpracování a generování SMS zpráv, zjistíme, že tento proces je na XML popisu závislý. Z tohoto důvodu budeme muset vzory jednotlivých zpráv ukládat do databáze ve zdrojové podobě XML. Musíme tedy být schopni jak XML číst, tak i jej generovat.

Android SDK pro generování XML moc voleb nenabízí. Pokud zavrhneme manuální generování s využitím třídy `StringBuilder`, pak jedinou přívětivou možnost pro přímé generování poskytuje třída `XmlSerializer`, která je obsažena, stejně jako XML pull parser, v knihovně `org.xmlpull.v1.XmlSerializer`. Tato třída poskytuje základní metody pro přidávání elementů, přidávání atributů a v neposlední řadě taky pro ukončení celého dokumentu. V našem případě však nemusíme generovat XML ručně, jelikož jej již budeme mít napsané. Musíme však jednotlivé části najít a oddělit je od zbytku.

V tuto chvíli se již pomalu dostáváme do fáze, kdy si krom způsobu generování vybereme i způsob zpracování XML dokumentu, jelikož tyto dvě volby spolu velmi úzce souvisí. Od SDK verze 8 bylo v Android aplikacích umožněno využít třídu `TransformerFactory`, která dokáže vygenerovat XML z uzlu DOMu. Tuto třídu tedy využijeme pro generování a pro parsování vstupního souboru využijeme implementaci DOM.

6.3.3 Export tras

Další a zároveň poslední činností, kterou bude klientská část provádět, bude export GPS tras. Uživatel tak získá možnost přenést zaznamenané trasy do jiného softwaru, popřípadě si je takto zálohovat. Pro export budou k dispozici dva základní formáty.

Prvním bude formát GPX[8] [37], který se používá pro přenos geografických dat mezi různými zařízeními či aplikacemi. Pomocí tohoto typu souboru jsme schopni zaznamenat samostatné body (například města), trasy (uspořádaná posloupnost bodů s časovými známkami) nebo cesty (uspořádaná posloupnost bodů bez časových známek). Tyto zaznamenané údaje lze poté zobrazit například v mapových podkladech od společnosti Google (Google Earth) [38].

O export do tohoto souboru se bude starat třída `GPXWriter`, která je odvozena od abstraktní třídy `AbstractExporter`. Jelikož GPX vychází z XML, budeme tento soubor generovat pomocí již dříve zmíněné třídy `XmlSerializer`. Abychom zaznamenali trasu v tomto formátu, je třeba zaznamenat všechny pozice i s časovou známkou do track segmentu (element `trkseg`), který vložíme do elementu `trk`. Takto vygenerujeme všechny trasy, které umístíme do kořenového prvku `gpx`. Jednotlivé exporty se pak budou v aplikaci ukládat na SD kartu ve složce `AlarmController/export/`.

[8] GPS Exchange Format

Druhý formát poslouží k zobrazení nasbíraných dat v programu MS Excel či OpenOffice, což uživateli umožní je statisticky či graficky zpracovat. Tato volba se může hodit především v kombinaci s tvorbou knihy jízd, kdy získaná data bude takto možné snadno přepracovat do oficiálního výkazu. Data tedy umožníme ukládat do souboru XLS. Tento export bude mít na starosti třída `XLSWriter`, která je opět odvozená z třídy `AbstractExporter`. V této třídě pro zápis do tohoto formátu využijeme knihovnu *JExcelApi* [39], která umožňuje číst, měnit a vytvářet tyto soubory. Knihovna je dostupná pod licencí open-source, můžeme jí tedy volně použít.

6.3.4 Uživatelské rozhraní

Intuitivní uživatelské rozhraní tvoří základ úspěchu všech grafických aplikací. Programy pro mobilní platformy jsou v rámci tohoto návrhu do značné míry specifické. Při rozvržení prvků na obrazovce musíme brát ohled na různé aspekty. Musíme si například uvědomit, jakým způsobem uživatel mobil ovládá, zda používá stylus nebo jej ovládá dotyky prstů. Protože všechny nové smartphony s operačním systémem android jsou ovládány bez stylusu, uživatelské rozhraní jsme se snažili vytvořit tak, aby bylo vždy snadné prstem vybrat konkrétní volbu. Při implementaci rozhraní bylo také nutné si uvědomit, že mobilní zařízení mohou disponovat displeji s různými rozlišeními. Místo nastavení pevné šířky a výšky jsme tedy u jednotlivých prvků raději používali hodnoty `WRAP_CONTENT` nebo `FILL_PARENT`, které způsobily, že se prvek vykreslil jen v takové velikosti, aby byl schopen zobrazit obsah (`WRAP_CONTENT`), nebo že se zobrazil v maximální velikosti, kterou mu povolil nadřazený layout (`FILL_PARENT`). Jelikož je v aplikaci použita spousta ikon, bylo nutné i je přizpůsobit různým rozlišením. Všechny použité ikony jsou tedy v aplikaci obsaženy ve třech různých velikostech, pro rozlišení *hdpi*[9], *mdpi*[10] a *ldpi*[11].

V aplikaci byly použity speciální prvky uživatelského rozhraní a další podpůrné prostředky, které zajišťují validaci vstupních polí. Použité prvky si nyní blíže popíšeme.

ActionBar

Do záhlaví všech aktivit jsme umístili tzv. *ActionBar*, jenž se často používá jako náhrada pro zastaralý *TitleBar*. Tento panel obsahuje text, který reflektuje účel aktivity, nebo obsahuje název aktuálně zvoleného alarmu. Krom tohoto statického textu, panel bude také obsahovat speciální tlačítka, tzv. *ActionItems*. Ty lze použít buď pro navigaci mezi aktivitami nebo také pro provádění různých operací. V aplikaci tedy nebudeme mít *option menu*, jelikož tlačítka panelu plně nahrazují jeho funkci.

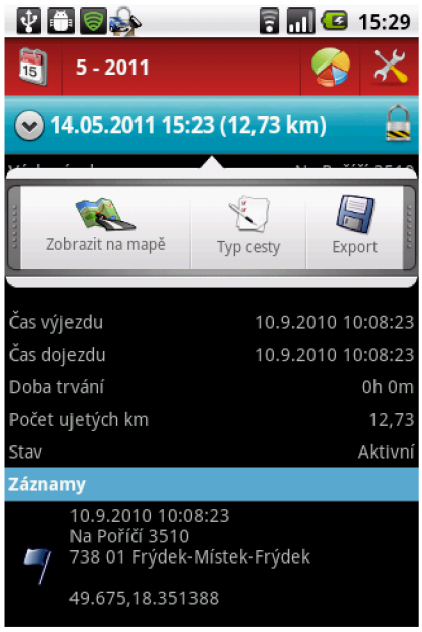
[9] High dots per inch
[10] Medium dots per inch
[11] Low dots per inch

ActionBar je obsažen v Android SDK až od verze 11 (Android 3.0), musíme si tedy tento prvek buď sami vytvořit, nebo použít externí knihovnu. Pro tento panel jsme nakonec použili knihovnu nesoucí název *Android ActionBar* [40].

QuickAction

QuickAction je nová komponenta, která se používá pro zobrazování kontextového menu mimo dialog, přímo v hlavní obrazovce. Výhodou tohoto prvku je, že uživatel stále ví, ke které položce se volby vážou. Nemůže tak dojít k omylu.

Přestože se s tímto prkem můžeme setkat jak ve standardním panelu pro práci s kontakty, tak i v aplikaci Twitter, komponenta není obsažena v Android SDK. Opět tedy použijeme externí kód, konkrétně projekt uvedený v článku zabývající se touto problematikou [41].



Obrázek 15: Ukázka komponenty QuickAction v seznamu ujetých tras.

MapView

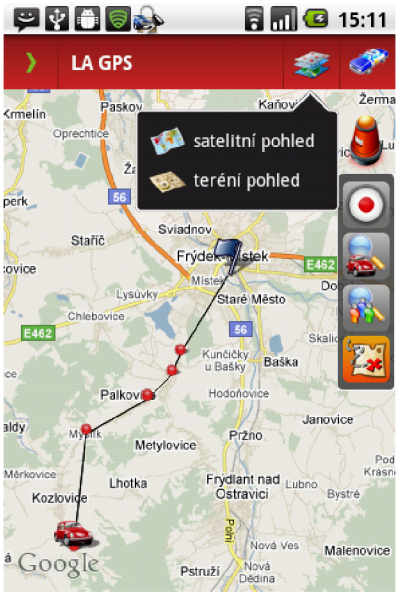
Jedním z hlavních rozšíření, kterým se aplikace snaží ulehčit uživateli práci s GPS alarmem, je zobrazení aktuální polohy vozidla na mapě. V programu tedy musí být obsažena mapová komponenta. V tomto případě však nemusíme používat žádný kód třetích stran, můžeme rovnou použít prvek s názvem *MapView*, který je obsažen v rozšíření Google API.

Abychom tuto komponentu mohli v programu použít, musíme se odkázat na onu knihovnu v manifestu projektu a povolit přístup k internetu. Tohoto docílíme přidáním následujících dvou řádků do souboru `AndroidManifest.xml`:

```
<uses-library android:name="com.google.android.maps" />
<uses-permission android:name="android.permission.INTERNET" />
```

Protože komponenta je propojena s Google Maps, je třeba vytvořit Maps API klíč, abychom mohli mapy používat. Tento klíč je nutné uvést při použití mapové komponenty v layoutu.

Na této mapě budeme zobrazovat jak aktuální polohu vozidla, tak i trasy, které byly aplikací zaznamenány. K zobrazení jednotlivých značek na mapě bylo využito třídy `OverlayItem`, ze které jsme vytvořili třídu `MapOverlayItem`. Tyto objekty jsou pro zobrazení umístěny do instance třídy `MapItemizedOverlay`, která se stará o vykreslování značek na zadané GPS pozici a také o zpracování různých událostí souvisejí s těmito značkami, jako například `onTap`. Zachycení této události jsme v aplikaci využili pro zobrazování detailních informací o zvolené značce. Uživatel takto může kdykoliv zpětně zjistit, kdy a na jaké adrese se vozidlo během trasy nacházelo.



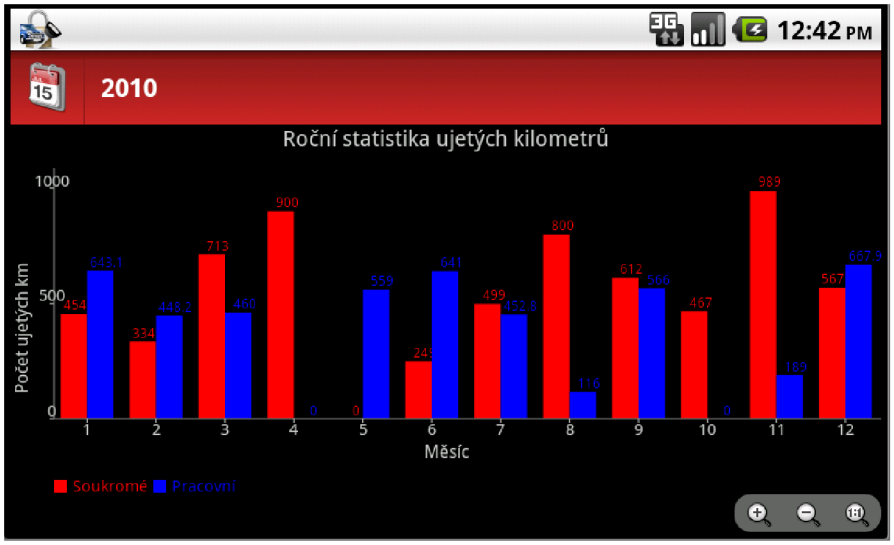
Obrázek 16: Mapa zobrazující trasu vozidla během poplachu.

Grafy

Přestože aplikace umožňuje vyexportovat trasy do formátu XLS, který lze zpracovat v tabulkovém editoru MS Office či v bezplatném balíčku OpenOffice, bude aplikace umožňovat zobrazit alespoň jeden graf, který bude zachycovat celkový počet ujetých kilometrů v jednotlivých měsících v roce. Tento údaj bude navíc rozdělena na soukromé a pracovní cesty, uživatel tak bude moci získat rychlý přehled o tom, kolik a za jakým účelem v daném měsíci ujel kilometrů.

Na internetu lze nalézt mnoho knihoven, které umožňují vykreslovat různé grafy. Nejpropracovanější a také nejobsáhlejší, co se do počtu grafů týče, je knihovna *AChartEngine* [42]. Tento projekt umožňuje vytvořit více než deset různých druhů grafů, jako například koláčový, sloupcový či spojnicový. V našem případě jsme tuto roční statistiku vynesli do grafu sloupcového, který se bude skládat ze dvou sérií.

Pro vytvoření grafu prostřednictvím této knihovny se používají dvě základní třídy. První třída obsahuje informace týkající se vzhledu grafu, kde lze například navolit velikost písma nebo barvy popisků. V našem případě jsme využili speciální třídu `XYMultipleSeriesRenderer`, která se používá pro 2D grafy s více datovými sériemi. Druhou důležitou třídou je `XYMultipleSeriesDataset`, která uchovává hodnoty grafu. Pro sloupcový graf musíme tyto hodnoty vložit přes instanci třídy `CategorySeries`. Ukázku vytvořeného grafu můžete vidět na následujícím obrázku.



Obrázek 17: Roční statistika ujetých kilometrů pro rok 2010.

Validace vstupních polí

Stávající verze SDK neposkytuje téměř žádné prostředky, kterými by se daly data ze vstupních polí validovat. U textových vstupních polí můžeme pouze nastavit typ klávesnice, který bude zobrazen, pokud toto textové pole bude mít nastaven fokus. Protože v našem případě budeme data potřebovat validovat i podle jiných kritérií, jako například, zda položka je povinná, nebo zda číslo leží v povoleném intervalu, byly v aplikaci vytvořeny jednoduché třídy, které tuto validaci obstarávají.

Jako bázová třída pro tyto validátory slouží `AbstractValidator`, která spravuje výpis chybových zpráv a také definuje společné rozhraní, aby nebylo při validaci nutné rozpoznávat, o jaký konkrétní typ se jedná. Nejdůležitějším potomkem této třídy je `ComplexValidator`, který jako jediný může obsahovat jiné vnořené prvky. Tato třída slouží pro seskupování prvků uživatelského rozhraní a jejich validačních objektů do skupiny, které lze nastavit atribut `required`. Pokud tento atribut je nastaven na hodnotu `false` a zároveň alespoň jeden prvek ze skupiny nemá zadanou hodnotu, přestože by tato hodnota měla být zadána, nedojde k validační chybě. Tímto způsobem budou validovány prvky příkazové SMS zprávy, které obsahují ve své XML definici element `COMPLEX` (viz kapitola 5.7.3).

Dalším validátorem, který je třeba zmínit, je `TextValidator`. Instance této třídy má na starosti validaci hodnot, které se mohou vyskytovat v textových polích, jako například klasický text, či celé nebo desetinné číslo. U číselných hodnot v této třídě dochází ke kontrole, zda jde skutečně o textový řetězec tvořený čísly a také zda číslo je větší než minimální povolená hodnota a zároveň menší než maximální povolené číslo. Pokud číslo neodpovídá nastaveným kritériím, pak validace selže.

Přestože zbývající validační třídy již neposkytují žádnou rozšiřující validaci podle jiných kritérií, jejich výskyt je důležitý, jelikož zpřístupňují hodnoty z konkrétních prvků uživatelského rozhraní, jako například z prvku *Spinner*, či ze *SeekBaru*, který v naší aplikaci používáme pro zadávání časových jednotek. Získané hodnoty jsou pak použity při testu, zda je daný prvek zadán, či nikoliv.

6.4 Android oprávnění

Android platforma klade velký důraz na bezpečnost, proto aplikace nemohou vykonávat některé operace bez povolení uživatele. Pokud program chce tyto operace provádět, musí nejprve nadefinovat, jaké oprávnění pro svůj běh potřebuje. Tyto údaje musí být uvedeny v souboru `AndroidManifest.xml`, ze kterého jsou tyto informace získány a zobrazeny uživateli při instalaci aplikace. Pokud je uživatel nepotvrdí, pak aplikace nemůže být nahrána do mobilního přístroje.

Naše aplikace bude pro svou činnost potřebovat celou řadu oprávnění, bez kterých by jinak nemohla plně fungovat. Význam některých z nich si zde nyní blíže popíšeme.

Nejdůležitější oprávnění, které vychází ze způsobu komunikace s alarmem, je požadavek na čtení a odesílání SMS zpráv. Z tohoto důvodu musíme získat `android.permission.RECEIVE_SMS` a `android.permission.SEND_SMS`. Jelikož však některé alarmy komunikují také prostřednictvím telefonního hovoru, musíme také získat oprávnění pro tuto operaci. K tomuto využijeme `android.permission.CALL_PHONE`, jenž nám umožňuje inicializovat odchozí hovor bez žádné interakce s uživatelem. Budeme tedy moci kdykoliv kontaktovat alarm, aniž by o tomto uživatel musel vědět.

Abychom mohli v programu načítat XML konfigurace a také provádět export tras do souborů XLS a GPX, musíme získat přístup k externímu úložišti (k SD kartě). Z tohoto důvodu aplikace vyžaduje `android.permission.WRITE_EXTERNAL_STORAGE`, jenž tento přístup garantuje.

Funkce sledování vozidla může pro svou činnost využívat GPS souřadnic získaných z mobilního telefonu. Abychom mohli tuto polohu získávat, opět potřebujeme mít přidělena určitá oprávnění. Tím je `android.permission.ACCESS_FINE_LOCATION`, které nám umožňuje získat polohu jak prostřednictvím GPS modulu, tak i pomocí BTS stanic, či WI-FI sítí. V našem případě však budeme používat výhradně souřadnic z GPS, jelikož jejich přesnost je největší.

Poslední dvě oprávnění, která si zde uvedeme, nesouvisí přímo s vykonávání určité funkce, nýbrž slouží především k zajištění většího komfortu uživatele při ovládání bezpečnostního zařízení, či k lepšímu upozornění uživatele při vyvolaném poplachu. Těmito oprávněními jsou `android.permission.READ_CONTACTS` a `android.permission.VIBRATE`. První zmíněné umožní aplikaci přistupovat ke kontaktům uložených v paměti telefonu. Uživateli tak v aplikaci zpřístupníme telefonní seznam, z něhož bude moci vybrat kontakty, jejichž čísla budou automaticky nahrány do příslušných vstupních polí. Usnadníme a zrychlíme tak zadávání vstupních dat pro příkazové SMS zprávy. Druhé oprávnění slouží k zapnutí či vypnutí vibrací v mobilu. Můžeme tak během poplachu využít všechny možnosti, jak uživatele upozornit na poplach či událost, kterou je potřeba co nejdříve obsloužit.

7 Testování

Důležitou etapou vývoje každého softwaru je testování a ani v našem případě tomu nebude jinak. Činnost naší aplikace musíme ověřit v praxi a to v kombinaci s reálným zařízením. Pro ověření, že aplikace se chová korektně, jsme využili autoalarm LA GPS [43] prodávaný společností Levnealarmy.cz. Toto bezpečnostní zařízení se chová a komunikuje stejným způsobem jako konkurenční alarm TS GPS (viz kapitola č. 3.2). Pomocí námi užitého alarmu tedy můžeme otestovat veškerou funkčnost aplikace. Pro komplexní ověření správného chodu jsme vytvořili celkem deset případových studií.

7.1 Společné předpoklady testování

Abychom mohli aplikaci otestovat pro výše uvedené bezpečnostní zařízení, aplikace musí mít uloženou konfiguraci tohoto alarmu. Ta byla pro naše testování vytvořená a stala se součástí aplikace. Pro provedení všech případových studií se očekává, že uživatel zná heslo a telefonní číslo SIM karty alarmu, aby mohl v aplikaci vytvořit účet pro tento přístroj. V našem případě je alarm stále ve výchozím nastavení, jako přístupové heslo použijeme číselný kód „4321“.

7.2 Nastavení telefonních čísel pro poplašné zprávy

Popis:

Uživatel chce navolit nová autorizovaná telefonní čísla, na které budou alarmem zasílány poplašné zprávy. Aplikace pro tuto změnu poskytuje uživatelské rozhraní, které umožňuje zadat až tři čísla, přičemž u každého lze zvolit, zda číslo má být pouze pro poplach, který je vyvolán vloupáním do vozidla, nebo pro případ, kdy vozidlo je v pohybu. Pro otestování budeme chtít uložit telefonní číslo +420777010030 pro vstupní alarm a číslo +420777020040 pro GPS poplach.

Předpoklady:

1. Aplikace je spuštěná na popředí.
2. V programu je zvolen alarm LA GPS.

Očekávaný výsledek:

Po zadání a potvrzení formuláře aplikace odešle alarmu zprávu s následujícím textem:

```
4321 PHONE +420777010030I +420777020040G
```

Alarm na základě této zprávy zruší doposavad uložená telefonní čísla a místo nich uloží čísla +420777010030 a +420777020040.

Postup:

1. Uživatel v hlavním panelu aplikace zvolí volbu „Ovládání alarmu“.
2. V seznamu nalezne a zvolí operaci „Nastavení telefonních čísel“.
3. Do polí s názvem „Telefonní číslo“ zapíše nebo vybere prostřednictvím telefonního seznamu čísla +420777010030 a +420777020040.
4. U volby „Zabezpečení“ zvolí písmeno I („Input alarm“) pro první číslo a pro druhé nastaví volbu G („GPS alarm“).
5. V dolní části displeje tento příkaz potvrdí tlačítkem s textem „Potvrdit“.
    a. Program neprovede operaci, protože nejsou zadány všechny povinné položky.
    b. Program odešle SMS zprávu autoalarmu a vrátí se na seznam všech operací.

Závěr:

Alarm obdržel příkazovou SMS zprávu, uložil si nová telefonní čísla a potvrdil provedení operace zasláním zprávy `PHONE OK`. Ověření, zda na čísla bude zaslána poplašná zpráva, bude provedeno v případové studii uvedené v kapitole č. 7.9.

7.3 Kontrola kreditu

Popis:

Jelikož SIM karta v alarmu nevyužívá žádného paušálního tarifu, nýbrž přednabitého kreditu, uživatel chce kontrolovat aktuální výši této částky, aby jí mohl vždy zavčas dobít. Minimální hodnota kreditu musí být 100 Kč, přičemž cena jedné SMS je 1 Kč.

Předpoklady:

1. Aplikace je spuštěná na popředí.
2. V programu je zvolen alarm LA GPS.
3. SIM karta alarmu využívá pro telefonní služby přednabitý kredit.

Očekávaný výsledek:

Jakmile částka klesne pod nastavenou mez, aplikace vyvolá tzv. bar notifikaci, ve které upozorní na tuto skutečnost.

Postup:

1. Uživatel z hlavní části aplikace zvolí volbu Nastavení→Kontrola kreditu.
2. V zobrazeném formuláři uživatel povolí službu, zadá telefonní číslo pro zjištění kreditu, nastaví současnou hodnotu kreditu a vyplní limit a částku jedné SMS.
3. Jakmile přijde SMS zpráva z alarmu, aplikace dekrementuje hodnotu a provede porovnání:
    a. Pokud aktuální hodnota je nižší než nastavený limit, vyvolá se notifikace upozorňující na tuto skutečnost.
    b. V opačném případě uživatel není nijak upozorněn na příchozí SMS ani na aktuálně přednabitou částku.

Závěr:

Tato kontrola byla aktivována na počátku testování. Při každé příchozí SMS ze zvoleného alarmu program snížil aktuální částku. Jakmile tato hodnota klesla pod hranici 100 Kč, program vyvolal notifikaci. Test proběhl úspěšně.

7.4 Vyžádání si aktuální polohy vozidla

Popis:

V aplikaci jsou zobrazeny mapové podklady, na kterých chceme zobrazit aktuální polohu vozidla. Program musí na požádání zjistit polohu vozidla a získané souřadnice zanést na mapu.

Předpoklady:

1. Aplikace je spuštěná na popředí.
2. V programu je zvolen alarm LA GPS.
3. Uživatel má v aplikaci zobrazeny mapové podklady.
4. Alarm je vybaven GPS modulem.

Očekávaný výsledek:

Bude zjištěna aktuální poloha vozidla, která se zobrazí na mapě ve formě ikony. Při vykreslování polohy musí mapa automaticky zobrazit oblast, ve které se tento bod nachází.

Postup:

1. V právě části mapy je umístěn panel, který obsahuje ovládací tlačítka. Uživatel zvolí „Získání polohy vozidla“.
2. Z grafické části aplikace je vyvolán požadavek na service, aby zjistila polohu automobilu. Po dobu, než je poloha navrácena, je uživateli zobrazen progres dialog.
3. Service vytvoří příkazovou SMS zprávu a odešle jí na telefonní číslo alarmu.
4. Alarm obdrží zprávu a zpět posílá svou aktuální pozici.
5. Service přijme SMS zprávu, analyzuje jí a pokud se jedná o zprávu s polohou, informuje o ní grafickou část aplikace.
6. Na mapě je zobrazena aktuální poloha vozidla.

Závěr:

Tento test proběhl úspěšně. Service odeslala alarmu zprávu pro zjištění pozice, na základě které byla alarmem odeslána odpověď. Aplikace na tuto zprávu s pozicí zareagovala korektně a to tak, že zrušila progres dialog, zobrazila ikonu na dané pozici a v mapě zobrazila oblast, jejímž středem je získaná GPS pozice.

7.5 Sledování trasy vozidla pomocí alarmu

Popis:

Uživatel si přeje mít informace o trasách a o ujetých kilometrech automobilu. Proto aktivuje službu „Kniha jízd“, která zaznamenává pravidelně polohu vozidla po celou dobu zapnutého sledování.

Předpoklady:

1. Služba „Kniha jízd“ je povolená.
2. Alarm je vybaven GPS modulem.

Očekávaný výsledek:

Veškerý pohyb vozidla během sledování je zaznamenáván a uložen. Pro jednotlivé trasy jsou uvedeny informace, jako například výchozí adresa, doba sledování či počet ujetých kilometrů. Průběh celé jízdy lze zobrazit na mapě.

Postup:

1. Sledování vozidla je manuálně či automaticky (pomocí bluetooth zařízení) spuštěno.
2. Service v pravidelných intervalech zasílá alarmu zprávu pro zjištění GPS souřadnic.
3. Alarm na tyto zprávy odpovídá zasláním polohy.

4. Získaná poloha je uložená do databáze a v grafické části aplikace je zobrazená na mapě.
5. Jakmile je sledování deaktivováno, service se přestane dotazovat alarmu na polohu.

Závěr:

Během tohoto testu bylo celkem zaznamenáno pět tras, během kterých nedošlo ani k jednomu selhání či nekorektnímu chování. Tyto všechny záznamy obsahovaly jak výchozí tak i cílovou adresu, počet ujetých kilometrů i celkovou dobu sledování. Na základě tohoto testu tedy můžeme říci, že aplikace je schopná bez problému provádět toto sledování vozidla.

7.6 Sledování trasy pomocí souřadnic z mobilu

Popis:

Obdobně jako v předcházejícím testu, uživatel chce zaznamenávat ujetou trasu vozidla. Protože však automobil využívá výhradně on a vždy při této jízdě má u sebe mobilní telefon, chce ušetřit za SMS komunikaci mezi telefonem a alarmem, a tak pro zjištění polohy bude využívat GPS modulu z mobilního přístroje.

Předpoklady:

1. Služba „Kniha jízd" je povolená.
2. Mobilní telefon je vybaven GPS modulem.

Očekávaný výsledek:

Prostřednictvím mobilního přístroje budou zaznamenány trasy ujeté automobilem.

Postup:

1. Sledování vozidla je manuálně či automaticky (pomocí bluetooth zařízení) spuštěno.
2. Service si vyžádá získávání souřadnic z GPS modulu v mobilu.
    a. Pokud je GPS nedostupná, tak aplikace upozorní uživatele na tuto skutečnost a dočasně pozastaví sledování.
3. Získaná poloha je uložená do databáze a v grafické části aplikace je zobrazená na mapě.
4. Sledování polohy je deaktivováno a s ním i získávání GPS souřadnic z mobilu.

Závěr:

Při sledování polohy prostřednictvím GPS modulu z mobilního telefonu nebyly nalezeny žádné nedostatky programu. V průběhu testu byl modul opětovně aktivován a deaktivován, aplikaci se však s touto situací poradila. Jakmile došlo k vypnutí, program tuto událost zdetekoval a upozornil

na ní uživatele. Jakmile byla GPS opětovně aktivována, program pokračoval ve sledování. Trasa vozidla byla úspěšně zaznamenána.

7.7 Export tras do formátu XLS při mazání

Popis:

Ujeté trasy jsou pro majitele vozidla velmi důležité, proto si je chce zálohovat ve formátu XLS. Pokud aplikace v rámci údržby bude mazat staré data, musí je nejprve vyexportovat do tohoto souboru a uložit jej na SD kartu. Stejný postup musí být aplikován i v případě, kdy jsou záznamy trasy smazány uživatelem.

Předpoklady:

1. V databázi jsou uloženy trasy ke zvolenému alarmu.
2. Mobilní přístroj disponuje externím úložištěm (SD kartou), které umožňuje zapsat soubor.
3. Program je nastaven, aby všechny mazané záznamy byly exportovány do formátu XLS.

Očekávaný výsledek:

Všechny trasy automobilu budou před smazáním uloženy ve formátu XLS do složky `AlarmController/config`. Všechny mazané záznamy jsou odstraněny z databáze.

Postup:

1. Uživatel vybere trasu, kterou chce smazat.
2. Program zobrazí dialog pro potvrzení operace.
    a. Pokud uživatel tuto operaci zamítne, tak operace mazání není provedena.
    b. Pokud je operace potvrzena, tak program vyexportuje záznam a smaže jej.
3. Jsou zobrazeny všechny záznamy tras pro aktuální měsíc a alarm.

Závěr:

Vybraná trasa byla úspěšně z databáze odstraněna a již nadále nebyla zobrazena v seznamu ujetých tras, ani na mapě. V adresáři `AlarmController/config` byl vytvořen soubor s názvem `LA_GPS_trasa_10_05_2011.xls`, který obsahuje tabulku s jednotlivými trasami (v našem případě pouze jednu trasu) a tabulku se všemi GPS souřadnicemi, které byly v rámci dané cesty zaznamenány.

7.8 Automatická aktivace/deaktivace alarmu

Popis:

Aplikace disponuje mechanismem, který umožňuje automaticky aktivovat a deaktivovat alarm, jakmile je uživatel poblíž svého vozidla. Pro detekci, zda se má provést daná operace se využívá bluetooth zařízení, které když se připojí k mobilnímu telefonu, tak aplikace automaticky vypne alarm a naopak, jakmile se zařízení odpojí, tak je alarm zapnut. Tuto funkcionalitu otestujeme s handsfree sadou *Parrot CK3100* [44] v kombinaci se speciální ochranou GPSS, která neustále kontroluje polohu vozidla. Pokud je tato ochrana aktivní a vozidlo je v pohybu, pak alarm detekuje poplach a zasílá poplašnou zprávu s GPS souřadnicí. Tuto ochranu lze vypnout a zapnout pouze pomocí telefonního hovoru.

Předpoklady:

1. Mobilní přístroj podporuje technologii bluetooth.
2. Mobilní přístroj je spárován s handsfree sadou Parrot CK3100.
3. Služba „Automatické řízení alarmu“ je aktivována a vázána na výše uvedené bluetooth zařízení.

Očekávaný výsledek:

Pokud se handsfree sada připojí k mobilnímu přístroji, aplikace deaktivuje alarm a naopak jakmile se přístroj odpojí od telefonu, program opětovně alarm aktivuje.

Postup:

1. Uživatel odemkne vozidlo klíčem a zapne zapalování.
2. Handsfree sada se připojí k telefonu. Na základě této události aplikace deaktivuje alarm.
3. Uživatel vypne zapalování.
4. Parrot CK3100 se odpojí od telefonu a alarm je aplikací opětovně aktivován.

Závěr:

Vytáčení telefonního čísla alarmu proběhlo okamžitě, jakmile mobilní přístroj zaznamenal spojení s bluetooth zařízením a také jakmile toto spojení ztratil. Prostřednictvím tohoto volání byla ochrana alarmu deaktivována a následně aktivována.

7.9 Upozornění na vyvolání poplachu

Popis:

Pokud alarm detekuje vloupání či pohyb vozidla bez sepnutého zapalování, pak odesílá na autorizovaná čísla poplašnou zprávu. Na tuto zprávu je třeba zareagovat co nejdříve, proto aplikace vyvolá notifikaci a dokud tato zpráva není vyzvednuta, tak neustále přehrává vyzváněcí tón, popřípadě vyvolává vibrace telefonu. Pokud i přesto nedojde k převzetí zprávy do nastaveného intervalu, pak je poplašná zpráva přeposlána na jiné vybrané číslo (číslo, které bylo uživatelem navoleno).

Předpoklady:

1. Alarm zaznamená bezpečnostní incident, na základě kterého odesílá poplašnou zprávu na autorizovaná čísla.
2. Mobilní telefon uživatele je zapnutý a je v dosahu GSM sítě.

Očekávaný výsledek:

Aplikace detekuje poplašnou zprávu, na základě které vyvolá varovné oznámení. Uživatel na toto oznámení zareaguje do nastaveného časového limitu, jinak je zpráva přeposlána.

Postup:

1. Alarm zaznamená vloupání do vozidla a zasílá poplašnou SMS.
2. Service zpracuje příchozí SMS a rozezná, že jde o poplach.
    a. Pokud ještě nebyl vyvolán poplach, tak je zobrazeno oznámení a nastaven časovač, po jehož uplynutí je tato zpráva přeposlána na přednastavené telefonní číslo.
3. Uživatel si všimne oznámení a potvrdí jej. Tato činnost automaticky spustí grafickou část aplikace a deaktivuje časovač pro přeposílání.

Závěr:

Aplikace správně zdetekovala poplašnou zprávu a okamžitě vyvolala notifikaci upozorňující na tuto událost. Informace o poplachu byla dokonce signalizována dříve, než byla ohlášena nová příchozí SMS zpráva. Upozornění na poplach naší aplikací je tedy rychlejší než by uživatel byl schopen zareagovat na příchozí SMS.

7.10 Automatické zjišťování polohy během poplachu

Popis:

Aplikace umožňuje během poplachu zaznamenávat trasu odcizeného vozidla. Některé autoalarmy zasílají GPS souřadnice automaticky s poplašnou zprávou, jiné však vyžadují, aby se uživatel na polohu vždy dotázal. Aby trasa vozidla byla během poplachu vždy správně zaznamenána, aplikace bude obstarávat dotazování sama. Tuto činnost v tomto testu ověříme.

Předpoklady:

1. Alarm je vybaven GPS modulem.
2. Služba „Automatické zjišťování polohy“ je aktivní.

Očekávaný výsledek:

Během poplachu bude zaznamenána ujetá trasa vozidla, přestože poplašné zprávy neobsahují GPS souřadnice.

Postup:

1. Alarm odešle poplašnou zprávu bez aktuální polohy vozidla.
2. Service detekuje tento poplach a aktivuje automatické získávání polohy vozidla.
3. V pravidelných intervalech je alarmu zaslána příkazová zpráva pro zjištění GPS souřadnic.
4. SMS obsahující pozici je v service zpracována a získané souřadnice jsou uloženy do databáze.
5. Uživatel deaktivuje poplach, na základě čehož dojde k ukončení periodického získávání polohy automobilu.

Závěr:

Během všech poplašných incidentů aplikace zaznamenávala polohu vozidla. Automatické získávání polohy tedy pracuje správně.

7.11 Spuštění externího zařízení pomocí plánovače

Popis:

Některá bezpečnostní zařízení umožňují připojit k alarmu externí spotřebiče, které lze vzdáleně prostřednictvím alarmu spouštět. Naše aplikace proto poskytuje plánovač, prostřednictvím kterého lze navolit, kdy a na jak dlouho dobu se má spustit daný spotřebič. V tomto testu ověříme, zda tato

činnost funguje správně a také zda nedojde ke komplikacím, pokud alarm neumožňuje spustit zařízení po dobu, kterou požaduje uživatel.

Při tomto testu budeme chtít, aby se zařízení spustilo ve 12:00 na 1 minutu. Alarm však neumožňuje spustit tento přístroj po delší dobu, než 30 sekund. Aplikace tedy bude muset pro tento běh zaslat vícero zpráv.

Předpoklady:

1. Alarm umožňuje připojit a vzdáleně ovládat externí zařízení.

Očekávaný výsledek:

Externí zařízení bude spuštěno ve 12:00 a jeho činnost bude ukončena ve 12:01.

Postup:

1. Uživatel v nabídce „Zařízení“ přidá nový úkol pro spotřebič s označením „A“. Tento úkol bude mít nastaven čas na 12:00 a dobu běhu na 1 minutu.
2. Service pro tento úkol vytvoří časovač, který vyvolá událost při dosažení času 12:00.
3. Jakmile je vyvolána událost od časovače, service vytvoří zprávu pro spuštění zařízení s označením „A“.
4. Jelikož je čas běhu větší než maximálně přípustná hodnota, tento čas je rozdělen na menší rámce. Aplikace postupně odešle příkazové SMS obsahující jednotlivé časové fragmenty, přičemž následující zpráva je odeslána až tehdy, když uplyne čas předcházející SMS.

Závěr:

Pro spuštění spotřebiče byly odeslány celkem dvě zprávy, které postupně spustily toto zařízení po dobu 30 sekund. Díky prodlevě při odesílání druhé zprávy jsme docílili tíženého efektu a to že zařízení ve skutečnosti běželo celou 1 minutu.

8 Možná rozšíření do budoucna

Přestože aplikace razantním způsobem rozšiřuje stávající možnosti bezpečnostních zařízení, daly by se do budoucna některé funkce ještě vylepšit či doplnit o další možnosti. V této kapitole si tedy v bodech nastíníme, jakým směrem by se mohl ubírat další vývoj tohoto programu.

8.1 Automatická aktivace/deaktivace alarmu na základě GPS

Jednou z funkcí, kterou aplikace nabízí, je automatická aktivace a deaktivace alarmu. V současné době je tato funkce závislá na existenci bluetooth zařízení (například hands free sadě), které musí být součástí hlídaného objektu. Jakmile se toto externí zařízení připojí k telefonu, alarm je deaktivován a naopak, jakmile se zařízení odpojí, tak se alarm opětovně aktivuje. Funkce pracuje velmi spolehlivě, avšak díky tomu, že je závislá na dalším přístroji, je její použití v praxi do značně míry omezené.

Toto omezení by se dalo eliminovat, pokud bychom pro zjišťování, zda je uživatel u vozidla, používali GPS souřadnice z mobilního přístroje a zároveň GPS souřadnice z bezpečnostního zařízení. Na základě nich bychom poté mohli rozhodnout, zda alarm má běžet, či jeho činnost má být pozastavena.

Tento způsob řešení však také není bez chyby a přináší celou řadu problémů, které jsou v dnešní době velmi obtížně řešitelné. Aby byla služba spolehlivá a nedocházelo k případům, kdy by se automobil nesprávně odemknul, museli bychom velmi často zjišťovat pozice a neustále kontrolovat, zda vzdálenost nepřekročila určitou mez. Toto by však vedlo k velmi razantnímu navýšení počtu odeslaných zpráv, což by funkci velmi prodražilo, nemluvě o tom, že neustálé získávání souřadnic v mobilním přístroji je velmi energeticky náročné, mobilní přístroj by se tedy mohl velmi rychle vybít. Museli bychom tedy vytvořit algoritmus, který by efektivně měnil periodu získávání a tím snížil energetickou i finanční náročnost této funkce. Pro toto řízení by se dalo například využít informací z BTS stanic, ze kterých bychom, na základě identifikátoru CID[12] rozpoznávali, zda je uživatel přihlášen pod stejnou BTS jako hlídané vozidlo, přičemž pokud by tomu tak bylo, bylo by zahájeno získávání pozic. Tento postup by velmi dobře fungoval ve městech, kde velikost BTS buněk je velmi malá a tedy by k získávání souřadnic docházelo velmi málo. V oblastech s menším pokrytím by však tento způsob nevedl k větším úsporám, bylo by tedy nutné tento algoritmus ještě optimalizovat.

Existuje ještě jeden technický problém, který nelze ze strany programátora nijak vyřešit. Současné GPS moduly, které jsou zabudovány v mobilních telefonech, mají velmi malou

[12] Cell ID

přesnost (cca. 30 m). Tato dosti značná odchylka by mohla způsobit, že automobil zůstane nechráněn, aniž by uživatel byl v jeho bezprostřední blízkosti. Z tohoto důvodu nebyl tento způsob do současné verze aplikace naimplementován a necháváme jej raději až jako možné rozšíření do doby, než souřadnice získávané z mobilu budou mít dostatečně velkou přesnost.

8.2 Generátor konfigurace

Přestože konfigurace naší aplikace je jak dobře čitelná, tak i lze snadno vytvořit bez hlubších znalostí IT, bylo by do budoucna vhodné vytvořit desktopovou aplikaci, která by tento proces ještě více usnadnila. Tento program by pracoval s XML schématem konfigurace a s načteným formátem SMS zprávy. Uživatel by v textu označil jednotlivé podřetězce, kterým by přiřadil typ elementu (`NUMBER`, `TEXT`, `PHONE`, ...) a atributy, které aplikace nabídne uživateli na základě vybraného prvku. K vlastnostem, které mohou nabývat pouze určitých hodnot, či k vlastnostem, které mohou obsahovat i vyhrazené hodnoty (klíčová slova), by program nabízel jednotlivé volby s nápovědou, které by uživateli umožnili snáze hodnotu vyplnit. Jakmile by byly takto zpracovány všechny komunikační zprávy a byly vyplněny všechny povinné informace týkající se alarmu, bylo by možné konfiguraci vyexportována do XML souboru a následně použít v mobilní aplikaci.

8.3 Propojení s webovou aplikací

Současná podoba aplikace je vhodná především pro běžné majitele vozu, kteří chtějí mít svůj automobil pod neustálou kontrolou a sledování trasy vozidla používají především pro osobní účely. Pokud bychom chtěli, aby aplikace pronikla i do komerční sféry, bylo by vhodné vytvořit webovou aplikaci, která by umožňovala centralizovanou správu všech vozidel. Aplikace by pracovala stejným způsobem jako teď, avšak komunikace mezi alarmem a uživatelem by probíhala přes webovou službu, která by tvořila jakéhosi prostředníka. Uživatel by standardním způsobem ovládal alarm přes svůj mobilní telefon, avšak mobilní přístroj by nekomunikoval s alarmem prostřednictvím SMS, nýbrž komunikoval s webovou službou, která by toto nastavení zpracovala a operaci delegovala alarmu přes vlastní GSM bránu. Veškerá komunikace by tak byla zálohována a kdykoliv zpětně dohledatelná.

Hlavním úkolem této webové služby by však bylo sledovat neustále polohu vozidla. Na základě těchto dat by poté bylo možné kdykoliv zjistit, kde a v kolik hodin se vozidlo nacházelo a jak dlouhou dobu zde bylo. Zaměstnavatel by takto získal cenné informace, které by se dali využít pro zefektivnění práce, popřípadě ke kontrole, zda zaměstnanec nezneužívá firemní vozidlo pro své osobní účely. Všechny ujeté trasy by se samozřejmě uchovávaly a na konci každého měsíce by z nich byly generovány knihy jízd.

9 Závěr

Cílem této diplomové práce bylo navrhnout a vytvořit mobilní aplikaci, která bude mít za úkol spravovat příchozí hlášení od bezpečnostních alarmů a svou funkcionalitou rozšíří stávající možnosti těchto přístrojů. Abychom mohli takovýto program vytvořit, bylo nezbytné seznámit se s novodobými mobilními platformami, na nichž by tato aplikace mohla fungovat, a také se blíže obeznámit se specifikací moderních bezpečnostních GSM a GPS alarmů. Zde byla sledována především funkcionalita a způsob jejich komunikace, konkrétně komunikace prostřednictvím příkazových SMS zpráv. Tato problematika byla nastudována a v této práci podrobně popsána. V rámci studie bylo zjištěno, že alarmy neposkytují stejnou funkčnost a vůbec nekomunikují stejným protokolem. Formát zpráv je u většiny alarmů zcela odlišný, bylo tedy třeba vymyslet způsob, který vyřeší právě tyto rozdíly. Aplikace musí být obecná, nesmí být svázána pouze s určitými typy nebo modely.

Tento problém byl řešen nejprve pomocí návrhového vzoru Adaptér, od čehož bylo posléze upuštěno. Zmiňovaný způsob by vyžadoval vytvořit pro každý model vlastní třídu, která by definovala jak chování alarmu, tak i jeho komunikaci. Místo toho byl vymyšlen jiný postup, který pro popis alarmu používá XML soubor s předem definovanými značkami. Tento prostředek je vhodný především z důvodu, že zápis v XML je velmi dobře čitelný a lze snadno vytvořit v jakémkoliv XML editoru bez znalosti programovacího jazyka či samotného programu. Aby konfigurace šla jednoduše vytvořit a také aby bylo možné správnost tohoto zápisu formálně ověřit, bylo vytvořeno XML schéma, které strukturu zmíněného dokumentu popisuje. Jednotlivé části tohoto schématu byly popsány v kapitole č. 5.7.3, přičemž celý popis je přiložen v příloze této práce.

Jelikož aplikace neměla předem definovanou požadovanou funkčnost, bylo nutné jí navrhnout. Snažili jsme se při tom maximálně využít schopností moderních chytrých telefonů, aby tak mohly být stávající možnosti alarmů obohaceny o nové funkce a operace. Aplikace pro svou činnost například používá GPS modul pro získání aktuálních souřadnic, které jsou potřebné pro sledování pohybu vozidla. Pro zobrazení těchto tras bylo využito mapových podkladů od společnosti Google, uživatel tak nemusí tyto údaje zadávat do jiných aplikací. Celkem bylo navrženo přes osm takovýchto rozšíření.

Na základě vypracované analýzy a návrhu byla aplikace naimplementována. Jako cílovou platformu jsme zvolili operační systém Android, který patří mezi nejrozšířenější mobilní OS a zároveň poskytuje celou řadu operací a služeb, bez kterých by bylo velmi obtížné program vytvořit. Především bylo důležité, aby se dalo programově přistoupit k přijatým SMS zprávám a také aby bylo možné je v kódu generovat a odesílat. Všechny tyto uvedené operace Android podporuje, při vývoji jsme tedy nenarazili na žádný závažnější problém, který by znemožnil program dokončit.

Aplikace byla úspěšně vytvořena a také otestována v kombinaci s alarmem LA GPS. Celkem bylo v provozu provedeno deset testů, které proběhly bez výjimky úspěšně.

Tato verze je zcela funkční a připravená pro použití běžnými uživateli. Protože program není vázán pouze na jeden typ bezpečnostního zařízení, dá se použít v kombinaci s alarmy od různých výrobců. Program disponuje intuitivním uživatelským rozhraním, uživatel si tedy ovládání velmi brzy osvojí. Přestože se během vývoje nepodařilo navázat spolupráci ani s jedním výrobcem, můžeme očekávat, že se program velmi rozšíří a stane se mezi uživateli velmi oblíbený. Firmy by tak mohly dodatečně projevit zájem a aplikaci začít dodávat současně s alarmy.

Literatura

[1] Operating System Share. In *AdMob Mobile Metrics: Metrics Highlights* [online]. [s.l.]: Admob, 2010 [cit. 2010-10-19].
Dostupné z WWW:
<http://metrics.admob.com/wp-content/uploads/2010/06/May-2010-AdMob-Mobile-Metrics-Highlights.pdf>.

[2] *Apple: Developer* [online]. 2010 [cit. 2010-10-19].
Dostupné z WWW: <http://developer.apple.com/>.

[3] *Android: Developers* [online]. 2010 [cit. 2010-10-19]. What is Android?.
Dostupné z WWW: <http://developer.android.com/guide/basics/what-is-android.html>.

[4] *Open Source Iniative* [online]. 2010 [cit. 2010-12-30]. The Open Source Definition.
Dostupné z WWW: <http://www.opensource.org/docs/osd>.

[5] *Symbian: Blog* [online]. 2010 [cit. 2010-12-30]. Symbian Foundation is completing its transition to a licensing body.
Dostupné z WWW:
<http://blog.symbian.org/2010/12/17/symbian-foundation-is-completing-its-transition-to-a-licensing-body/>.

[6] *Symbian: Blog* [online]. 2010, [cit. 2010-12-30].
Dostupné z WWW: <http://blog.symbian.org/>.

[7] FITZEK, F.; REICHERT, F. *Mobile phone programming: and its Application to Wireless Networking*. Vyd. 1. Dordrecht: Springer, 2007. 473 s. ISBN 978-1-4020-5968-1.

[8] FLING, B. *Mobile Design and Development: Practical Techniques for Creating Mobile Sites and Web Apps*. Vyd. 1. United States of America: O'Reilly Media, 2009. 336 s.
ISBN 978-0-596-15544-5.

[9] POULÍČEK, Z. *Windows Mobile* [online]. 2010. [cit. 2010-10-19].
Dostupné z WWW:
<https://www.fit.vutbr.cz/study/courses/TAM/private/lect/TAM-WindowsMobileI.pdf>.

[10] *Microsoft* [online]. 2010 [cit. 2010-12-30]. Windows CE 5.0.
Dostupné z WWW:<http://www.microsoft.com/cze/windows/embedded/WindowsCE.mspx>.

[11] *Microsoft* [online]. 2010 [cit. 2010-12-30]. .NET Framework Developer Center.
Dostupné z WWW: <http://msdn.microsoft.com/en-us/netframework/>.

[12] *W3schools.com* [online]. 2010 [cit. 2010-12-30]. AJAX Introduction.
Dostupné z WWW: <http://www.w3schools.com/Ajax/ajax_intro.asp>.

[13] *W3schools.com* [online]. 2010 [cit. 2010-12-30]. JavaScript Introduction.
Dostupné z WWW: <http://www.w3schools.com/js/js_intro.asp>.

[14] *W3schools.com* [online]. 2010 [cit. 2010-12-30]. XML DOM Introduction. Dostupné z WWW: <http://www.w3schools.com/dom/dom_intro.asp/>.

[15] *Windows Phone* [online]. 2010 [cit. 2010-12-30]. 10 hlavních vylepšení ve Windows Mobile 6.5. Dostupné z WWW: <http://www.microsoft.com/cze/windowsphone/meet/wm65-top-ten-features.aspx>.

[16] *Microsoft Silverlight* [online]. 2010 [cit. 2010-12-30]. Microsoft Silverlight powers engaging, interactive user experiences wherever the web works. Dostupné z WWW: <http://www.microsoft.com/silverlight/what-is-silverlight/>.

[17] *MSDN: Blogs* [online]. 2006-08-25 [cit. 2010-12-30]. What is the XNA Framework. Dostupné z WWW: <http://blogs.msdn.com/b/xna/archive/2006/08/25/724607.aspx>.

[18] *Windows Phone* [online]. 2010 [cit. 2010-12-30]. Přehled systému Windows Phone 7. Dostupné z WWW: <http://www.microsoft.com/cze/windowsphone/meet/windows-phone-7.aspx>.

[19] *Jablotron* [online]. 2008 [cit. 2010-11-04]. GSM alarm CA-1803 BT "ATHOS". Dostupné z WWW: <http://www.jablotron.cz/cz/Katalog/autotechnika/gsm+autoalarmy/autoalarm+ca1803+bt+athos/>.

[20] Uživatelský manuál. In *Inteligentní GSM/GPS autoalarmy: CA-1802 a CA-1803 Athos* [online]. [s.l.] : JABLOTRON ALARMS a.s., 2008 [cit. 2010-11-04]. Dostupné z WWW: <http://www.jablotron.cz/upload/download/CA-180x_CZ_MHF57800.pdf>.

[21] Návod na použití. In *TS GPS* [online]. [s.l.] : [s.n.], 2010 [cit. 2010-11-04]. Dostupné z WWW: <http://www.9000.cz/pdf/ts-gps.pdf>.

[22] *Keetec : The best security and car accessories solutions* [online]. 2010 [cit. 2011-01-01]. Dostupné z WWW: <http://www.keetec.cz/>.

[23] Návod na použití. In *GPS LOC* [online]. [s.l.] : [s.n.], 2010 [cit. 2010-11-04]. Dostupné z WWW: <http://www.levnealarmy.cz/get_file.php?did=836&lang=cz&download=0>.

[24] *Satmaps.net* [online]. 2011 [cit. 2011-04-22]. Dostupné z WWW: <http://www.satmaps.net/>.

[25] Návod k použití. In *SEL01 – GSM alarm* [online]. [s.l.]: Selax Electronics s.r.o., 2010 [cit. 2010-11-04]. Dostupné z WWW: <http://www.selax.cz/domain/selax/files/gsm-alarmy/sel-01-manual.pdf>.

[26] *Flops* [online]. 2011 [cit. 2011-03-03]. Nokia hodila Symbian přes palubu, vítr do plachet má přinést Windows Phone 7.
Dostupné z WWW:
<http://www.flops.cz/aktuality/nokia-hodila-symbian-pres-palubu-vitr-do-plachet-ma-prinest-windows-phone-7>.

[27] *IOS Developer library* [online]. 2010, poslední změna 2010-05-11 [cit. 2011-04-17].
Map Kit Framework Reference.
Dostupné z WWW:
<http://developer.apple.com/library/ios/#documentation/MapKit/Reference/MapKit_Framework_Reference/_index.html>.

[28] *Windows Phone* [online]. 2010 [cit. 2011-05-04]. No way to programmatically access SMS?.
Dostupné z WWW:
<http://social.msdn.microsoft.com/Forums/en-US/windowsphone7series/thread/2738361c-734c-48f4-8451-3f02e4654cfc/>.

[29] *Stackoverflow* [online]. 2011 [cit. 2011-05-04]. How to programmatically send SMS on the iPhone?
Dostupné z WWW:
<http://stackoverflow.com/questions/4653615/how-to-read-sms-message-on-ios>.

[30] *Unified Modeling Language: Infrastructure* [online]. Vyd. 2.: OMG, 2006 [cit. 2010-12-30].
Dostupné z WWW: <http://www.omg.org/spec/UML/2.0/Infrastructure/PDF/>.

[31] MEIER, R. *Professional Android 2 Application Development*, Wrox, 2010,
ISBN 978-0-470-56552-0

[32] ZENDULKA, J. *UP - rozpracování : Návrh architektury* [online]. 2011 [cit. 2011-03-08].
Dostupné z WWW:
<https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/AIS-IT/lectures/7_SWarchitecture.pdf>.

[33] *SQLite* [online]. 2011 [cit. 2011-04-17]. SQLite Foreign Key Support.
Dostupné z WWW: <http://www.sqlite.org/foreignkeys.html>.

[34] *Android: Developers* [online]. 2011 [cit. 2011-04-17]. Data Storage.
Dostupné z WWW: <http://developer.android.com/guide/topics/data/data-storage.html>.

[35] *Android: Developers* [online]. 2011, poslední změna 2011-04-01 [cit. 2011-05-02].
Platform Versions.
Dostupné z WWW:
<http://developer.android.com/resources/dashboard/platform-versions.html>.

[36] GALPIN, M. *IBM: Build Java applications for mobile devices* [online]. 2009-06-23
[cit. 2011-04-26]. Working with XML on Android.
Dostupné z WWW: <http://www.ibm.com/developerworks/opensource/library/x-android/>.

[37] *GPX* [online]. 2004 [cit. 2011-05-08]. The GPS Exchange Format. Dostupné z WWW: <http://www.topografix.com/gpx.asp>.

[38] *Google earth* [online]. 2011 [cit. 2011-04-27]. Importing Global Positioning Systems (GPS) data in Google Earth. Dostupné z WWW: <http://earth.google.com/outreach/tutorial_importgps.html>.

[39] *Java Excel API* [online]. 2009 [cit. 2011-04-27]. A Java API to read, write, and modify Excel spreadsheets. Dostupné z WWW: <http://jexcelapi.sourceforge.net/>.

[40] *Github* [online]. 2010 [cit. 2011-04-27]. Android-actionbar. Dostupné z WWW: <https://github.com/johannilsson/android-actionbar>.

[41] *Lorenz's Blog* [online]. 2010 [cit. 2011-04-27]. How to Create QuickAction Dialog in Android. Dostupné z WWW: <http://www.londatiga.net/it/how-to-create-quickaction-dialog-in-android/>.

[42] *AChartEngine* [online]. 2009 [cit. 2011-04-28]. Dostupné z WWW: <http://www.achartengine.org/>.

[43] Uživatelský manuál. In *GPS lokalizátor: LA GPS lokalizátor* [online]. [s.l.] : [s.n.], 2010 [cit. 2010-11-04]. Dostupné z WWW: <http://www.levnealarmy.cz/get_file.php?did=860&lang=cz&download=0>.

[44] Instalační příručka. In *Parrot CK 3100/3300* [online]. [s.l.] : [s.n.], 2010 [cit. 2011-05-11]. Dostupné z WWW: <http://www.mobiland.cz/gsm/documents/parrot-ck3100.pdf>.

Seznam příloh

Příloha A: Schéma XML konfigurace

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- definition of attributes -->

  <xs:simpleType name="character">
    <xs:restriction base="xs:string">
      <xs:length value="1"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="time_units">
      <xs:restriction base="xs:string">
            <xs:enumeration value="milisec"/>
            <xs:enumeration value="sec"/>
            <xs:enumeration value="min"/>
            <xs:enumeration value="hour"/>
      </xs:restriction>
  </xs:simpleType>

<!-- definition of simple elements -->

   <xs:element name="text" type="xs:string"/>
   <xs:element name="word">
      <xs:complexType>
            <xs:attribute name="name" type="xs:string" use="required"/>
            <xs:attribute name="label" type="xs:string" use="required"/>
            <xs:attribute name="info" type="xs:string"/>
            <xs:attribute name="required" type="xs:boolean" default="1"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="option">
      <xs:complexType>
            <xs:simpleContent>
                  <xs:extension base="xs:string">
                        <xs:attribute name="name" type="xs:string"/>
                        <xs:attribute name="info" type="xs:string"/>
                  </xs:extension>
            </xs:simpleContent>
      </xs:complexType>
   </xs:element>
   <xs:element name="char">
      <xs:complexType>
            <xs:attribute name="value" type="character"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="number">
      <xs:complexType>
            <xs:attribute name="name" type="xs:string" use="required"/>
            <xs:attribute name="label" type="xs:string" use="required"/>
            <xs:attribute name="info" type="xs:string"/>
            <xs:attribute name="min" type="xs:unsignedInt"/>
            <xs:attribute name="max" type="xs:unsignedInt"/>
            <xs:attribute name="required" type="xs:boolean" default="1"/>
      </xs:complexType>
   </xs:element>
```

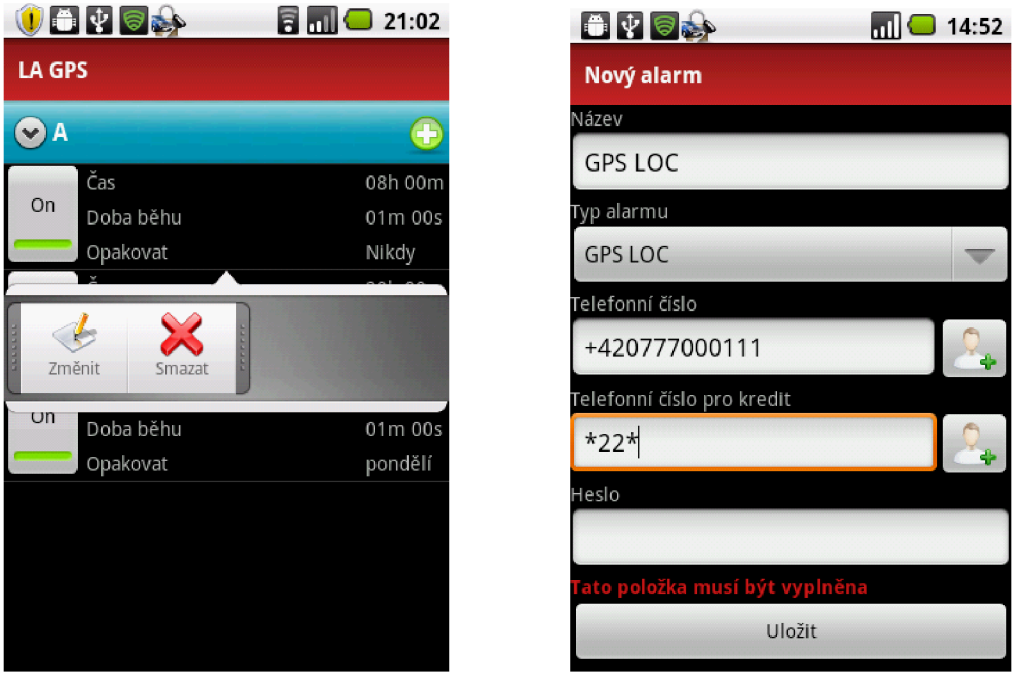
```
    <xs:element name="float">
       <xs:complexType>
             <xs:attribute name="name" type="xs:string" use="required"/>
             <xs:attribute name="label" type="xs:string" use="required"/>
             <xs:attribute name="info" type="xs:string"/>
             <xs:attribute name="separator" type="xs:string" default="."/>
             <xs:attribute name="min" type="xs:unsignedInt"/>
             <xs:attribute name="max" type="xs:unsignedInt"/>
             <xs:attribute name="required" type="xs:boolean" default="1"/>
       </xs:complexType>
    </xs:element>
    <xs:element name="time">
       <xs:complexType>
             <xs:attribute name="name" type="xs:string" use="required"/>
             <xs:attribute name="label" type="xs:string" use="required"/>
             <xs:attribute name="info" type="xs:string"/>
             <xs:attribute name="unit" type="time_units" default="sec"/>
             <xs:attribute name="min" type="xs:unsignedInt"/>
             <xs:attribute name="max" type="xs:unsignedInt"/>
             <xs:attribute name="required" type="xs:boolean" default="1"/>
       </xs:complexType>
    </xs:element>
    <xs:element name="phone">
       <xs:complexType>
             <xs:attribute name="name" type="xs:string" use="required"/>
             <xs:attribute name="label" type="xs:string" use="required"/>
             <xs:attribute name="info" type="xs:string"/>
             <xs:attribute name="required" type="xs:boolean" default="1"/>
       </xs:complexType>
    </xs:element>

<!-- definition of complex elements -->

    <xs:element name="select">
       <xs:complexType>
          <xs:sequence>
             <xs:element maxOccurs="unbounded" minOccurs="1" ref="option"/>
          </xs:sequence>
          <xs:attribute name="name" type="xs:string" use="required"/>
          <xs:attribute name="label" type="xs:string" use="required"/>
          <xs:attribute name="info" type="xs:string"/>
          <xs:attribute name="required" type="xs:boolean" default="1"/>
       </xs:complexType>
       <xs:unique name="UniqueOptionName">
             <xs:selector xpath="./option" />
             <xs:field xpath="@name" />
       </xs:unique>
    </xs:element>
    <xs:element name="complex">
       <xs:complexType>
          <xs:sequence maxOccurs="unbounded" minOccurs="0">
             <xs:choice>
                   <xs:element maxOccurs="unbounded" ref="word"/>
                   <xs:element maxOccurs="unbounded" ref="text"/>
                   <xs:element maxOccurs="unbounded" ref="char"/>
                   <xs:element maxOccurs="unbounded" ref="number"/>
                   <xs:element maxOccurs="unbounded" ref="float"/>
                   <xs:element maxOccurs="unbounded" ref="time"/>
                   <xs:element maxOccurs="unbounded" ref="phone"/>
```

```
                <xs:element maxOccurs="unbounded" ref="select"/>
            </xs:choice>
        </xs:sequence>
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="label" type="xs:string" use="required"/>
        <xs:attribute name="required" type="xs:boolean" default="1"/>
    </xs:complexType>
</xs:element>
<xs:element name="message">
    <xs:complexType>
          <xs:sequence maxOccurs="unbounded" minOccurs="0">
                <xs:choice>
                      <xs:element maxOccurs="unbounded" ref="word"/>
                      <xs:element maxOccurs="unbounded" ref="text"/>
                      <xs:element maxOccurs="unbounded" ref="char"/>
                      <xs:element maxOccurs="unbounded" ref="number"/>
                      <xs:element maxOccurs="unbounded" ref="float"/>
                      <xs:element maxOccurs="unbounded" ref="time"/>
                      <xs:element maxOccurs="unbounded" ref="phone"/>
                      <xs:element maxOccurs="unbounded" ref="select"/>
                      <xs:element maxOccurs="unbounded" ref="complex"/>
                </xs:choice>
          </xs:sequence>
          <xs:attribute name="name" type="xs:string" use="required"/>
          <xs:attribute name="label" type="xs:string" use="required"/>
          <xs:attribute name="info" type="xs:string" use="required"/>
    </xs:complexType>
    <xs:unique name="UniqueMessageItemName">
          <xs:selector xpath="./* | ./complex/*" />
          <xs:field xpath="@name" />
    </xs:unique>
</xs:element>
<xs:element name="calling">
    <xs:complexType>
          <xs:attribute name="name" type="xs:string" use="required"/>
          <xs:attribute name="label" type="xs:string" use="required"/>
          <xs:attribute name="info" type="xs:string" use="required"/>
    </xs:complexType>
</xs:element>
<xs:element name="incoming_actions">
    <xs:complexType>
          <xs:sequence>
             <xs:sequence maxOccurs="unbounded" minOccurs="0">
                <xs:choice>
                      <xs:element maxOccurs="unbounded" ref="message"/>
                      <xs:element maxOccurs="unbounded" ref="calling"/>
                </xs:choice>
             </xs:sequence>
          </xs:sequence>
    </xs:complexType>
    <xs:unique name="UniqueIncomingActionsName">
          <xs:selector xpath="./*" />
          <xs:field xpath="@name" />
    </xs:unique>
</xs:element>
<xs:element name="outgoing_actions">
    <xs:complexType>
          <xs:sequence>
                <xs:element maxOccurs="unbounded" ref="message"/>
          </xs:sequence>
    </xs:complexType>
```

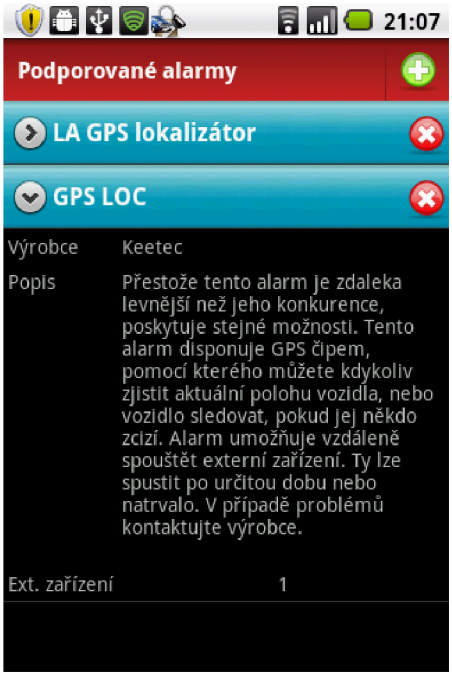
```
      <xs:unique name="UniqueOutgoingActionsName">
            <xs:selector xpath="./*" />
            <xs:field xpath="@name" />
      </xs:unique>
   </xs:element>
   <xs:element name="device">
      <xs:complexType>
            <xs:attribute name="name" type="xs:string" use="required"/>
      </xs:complexType>
   </xs:element>
   <xs:element name="devices">
      <xs:complexType>
            <xs:sequence>
                  <xs:element maxOccurs="unbounded" ref="device"/>
            </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="description" type="xs:string"/>
   <xs:element name="alarm">
      <xs:complexType>
            <xs:sequence>
                  <xs:element ref="description"/>
                  <xs:element ref="devices"/>
                  <xs:element ref="incoming_actions"/>
                  <xs:element ref="outgoing_actions"/>
            </xs:sequence>
      <xs:attribute name="name" type="xs:string" use="required"/>
      <xs:attribute name="vendor" type="xs:string" use="required"/>
      <xs:attribute name="alert_with_location" type="xs:boolean"
                    default="0"/>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Příloha B: Ukázka vzhledu aplikace

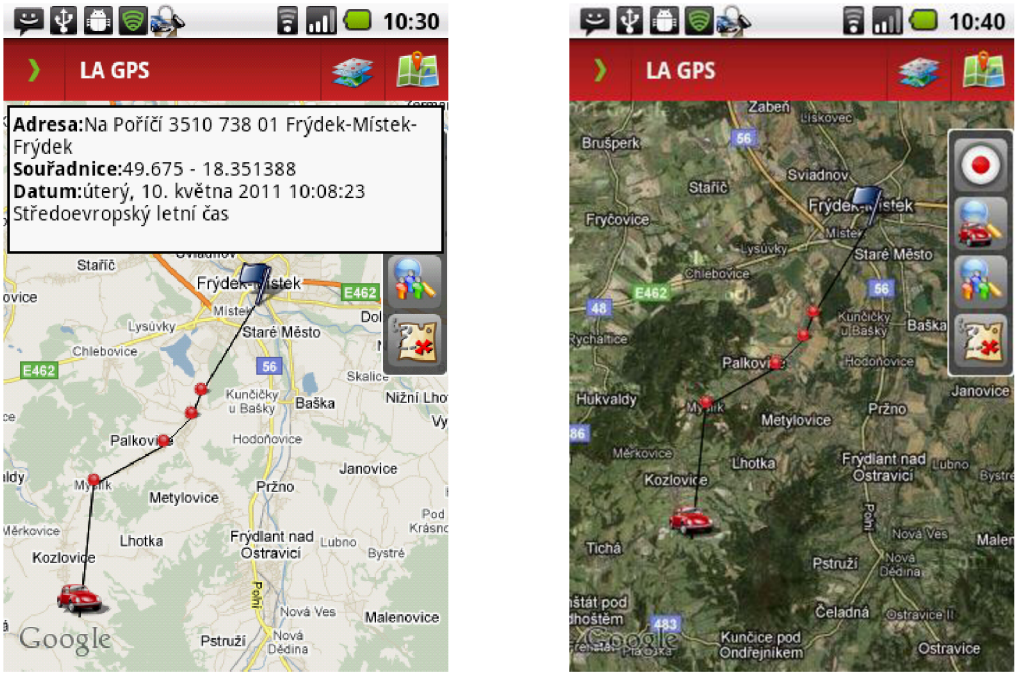
Obrázek B.1: Hlavní panel aplikace.



Obrázek B.2: Správa úkolů externího zařízení a formulář pro vytvoření nového alarmu.



Obrázek B.3: Správa konfigurací a správa ujetých tras.



Obrázek B.4: Zobrazení trasy na mapě.