UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

DIPLOMOVÁ PRÁCE

2010 **Bc. Lukáš Cír**

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Systém pro sledování zásilek

Bc. Lukáš Cír

Diplomová práce

2010

Poděkování

Chtěl bych poděkovat RNDr. Davidu Žákovi Ph.D. za cenné rady, informace, připomínky a vedení celé mé práce.

Prohlašuji

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 20. 8. 2010

Bc. Lukáš Cír

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2009/2010**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Lukáš CÍR**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Systém pro sledování zásilek**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

- V teoretické části bude proveden rozbor problematiky komunikačních protokol GPS modulů (např. NMEA) a seznámení s knihovnou určenou pro práci s GPS modulem. Dále se bude teoretická část zabývat problémem zpracování geografických dat, přepočtem GPS souřadnic na obrazové souřadnice a způsoby síťové komunikace mezi klientem a serverem. - Primárním cílem diplomové práce je návrh a realizace aplikace pro sledování a evidenci vozidel dopravce a/nebo přepravovaných zásilek. Systém bude rozdělen na část uživatelskou (WWW rozhraní) a administrační. Obě části systému budou podporovat vizuální interpretaci umístění daného vozidla/zásilky na území České republiky. Aplikace umožní i ukládání projetých tras pro případné další zpracování. - Zvláštní pozornost při návrhu aplikace a datových struktur bude věnována jednoduché rozšiřitelnosti aplikace o poskytování dalších informací o vozidle či zásilce. - Diplomová práce se bude orientovat na využití kartografických dat z otevřených zdrojů, jako např. openstreetmap.org. - Součástí práce bude uživatelská dokumentace, dokumentace síťové komunikace, dokumentace fyzického databázového modelu.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

1. Kiszka, B.: Oracle ? 1001 Tipů a triků pro jazyk Java. Computer Press, 2007. 2. Jelínek, L.: Jádro systému Linux. Computer Press, 2008. 3. Urman , S., Hardman, R., McLaughlin, M.: Oracle ? Programování v PL/SQL. Computer Press, 2007. 4. http://www.openstreetmap.org/ 5. http://www.abclinuxu.cz/serialy/gps-a-komunikacni-protokol-nmea

Vedoucí diplomové práce: **RNDr. David Žák, Ph.D.**
Katedra informačních technologií

Datum zadání diplomové práce: **30. října 2009**

Termín odevzdání diplomové práce: **21. května 2010**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

doc. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 10. listopadu 2009

Anotace

Účelem diplomové práce je analýza, návrh a tvorba balíčku aplikací, sloužících pro sledování zásilek. Součástí práce je i seznámení s technologiemi použitými při procesu implementace. Analýza a návrh jsou směrovány na využití technologií, na které nejsou uvaleny komerční licence. Velký důraz je kladen na efektivitu zpracování, plánování a přenos dat.

První část práce informuje o funkcích, které bude navržený systém splňovat. Druhá část popisuje technologie, které jsou použité při tvorbě aplikací. Kapitoly třetí části obsahují popis struktury systému a jeho funkčnosti a analýzu klíčových komponentů a algoritmů.

KLÍČOVÁ SLOVA

GPS, Soap, webové služby, zásilka, NMEA, openstreetmap, swt, PostGIS.

TITLE

Consingment tracking system

ANNOTATION

The point of this thesis is the analysis, design and creation of a package of applications, used for tracking of parcels. The paper also introduces technologies used in the process of implementation. The analysis and design are directed to the use of technologies that are not liable to commercial licences. Strong emphasis is put on the effectiveness of processing, planning and data transmission.

The aim of the first part is to instruct you about the functions of designed system. The second section gives information about technologies that were used to create mentioned applications. Chapter three describes structure of the system, its functionality and also the analysis of crucial components and algorithms.

KEYWORDS

GPS, Soap, web services, consingment, NMEA, openstreetmap, swt, PostGIS.

Obsah

Úvod 10
1. Systém pro sledování zásilek 11
1.1 Výhody systému 13
2. Použité technologie 14
2.1 Global Position Systém 14
2.1.1 Přesnost GPS pozice 16
2.1.2 Komunikace SiRF III GPS čipsetu 17
2.1.3 NMEA Protokol 17
2.1.4 Knihovna jazyka C libgpsd 20
2.1.5 Systém GPSD a libgps 22
2.2 Simple Object Access Protocol 24
2.2.1 Klient a knihovna jazyka C libcsoap 25
2.2.2 Server AXIS2/c a jeho knihovny 26
2.2.3 SOAP knihovny programovacího jazyka Java 30
2.3 OpenStreetMap.org otevřené mapy 31
2.3.1 Aplikace pro OpenStreetMap.org 32
2.3.2 Formát souboru OSM 32
2.4 Databáze 34
2.4.1 PostGIS 35
2.4.2 Java knihovna PostGIS 38
2.5 The Standard Widget Toolkit 39
2.6 Ostatní technologie 40
2.6.1 Formát souboru GPX (GPS eXchange Format) 40
2.6.1 Knihovna pro komunikaci s Webkamerou libvideo0 42
3. Struktura systému sledování 43
3.1 GPS Klient 45
3.2 Autorizační server 52
3.3 Aplikační server 55
3.4 Aplikace operátora 58
3.5 Webové aplikace 61
3.5.1 Webová aplikace pro zákazníky 61
3.5.2 Webová aplikace pro řidiče a operátory 62

4. SOAP Komunikace 65
4.1 Bezpečnost komunikace 68
4.2 Použití jmen a následné šifrování 70
4.3 Komunikace mezi serverovými službami 72
5. Zobrazení kartografických dat 73
5.1 Efektivní vykreslování 75
5.2 Přichycení entity k cestě 81
5.3 Zobrazení satelitů 85
6. Výpočet optimální trasy 88
7. Kniha jízd 91
8. Závěr 96
9. Seznam použité literatury 98
10. Přílohy 100
10.1 Databázový trigger pro ukládání GPS bodů 100
10.2 Databázový trigger a funkce pro tvorbu projeté trasy 101
10.3 Nápověda aplikace osm2pgsql 106
10.4 Uživatelský manuál k aplikaci operátora 107
10.4.1 Základní ovládací prvky 107
10.4.2 Správa uživatelských dat 110
10.4.3 Práce s vozovým parkem 112
10.4.4 Sledování vozidla 116
10.4.5 Práce s mapovým podkladem 117
10.4.6 Kniha jízd 118
10.5 Obrázky webových aplikací 121
10.5.1 Webová aplikace pro zákazníky 121
10.5.2 Webová aplikace pro řidiče a operátory 123
10.6 Přepočet GPS souřadnic na obrazové souřadnice 125

Úvod

Účelem diplomové práce je analýza, návrh a tvorba balíčku aplikací sloužících pro sledování zásilek. Součástí práce je i seznámení s technologiemi použitými při procesu implementace. Analýza a návrh jsou směrovány na využití technologií, na které nejsou uvaleny komerční licence. Velký důraz je kladen na efektivitu zpracování, plánování a přenos dat. Jednou z podmínek je zajištění plné podpory alespoň v operačních systémech Microsoft Windows a GNU/Linux.

Součástí balíčku aplikací je klientská GPS aplikace instalovaná ve vozidle a serverová část, zpracovávající požadavky GPS klienta. Nedílnou součástí je i desktopová aplikace operátora (správce vozidel, balíků atd.) a webové rozhraní pro řidiče/depo, příjemce/odesílatele.

Prioritním požadavkem při návrhu komunikace bylo její důkladné zabezpečení před případnými útočníky. V práci je navržen efektivní a bezpečný způsob přihlašování operátorů do systému. Práce obsahuje i informaci o kontrole integrity SOAP zpráv.

Práce se skládá z několika částí. První oddíl informuje o funkcích, které bude navržený systém splňovat. Druhá část popisuje technologie, použité při tvorbě aplikací.

Nejzajímavější část diplomové práce představují kapitoly, popisující strukturu systému a analýzu a funkčnost klíčových komponentů a algoritmů. Je zde vysvětlen mechanismus komunikace, zobrazovací algoritmy, optimalizační způsoby a řada dalších součástí systému. Práce je navíc doplněna četnými ilustracemi které, napomohou v pochopení funkcí a procesů popisovaného systému.

Závěr diplomové práce obsahuje zhodnocení, návrh možných vylepšení a rozšíření systému. V příloze se nachází doplňkové informace jako jsou kódy databázových triggerů a funkcí pro práci s body cesty, nápověda k aplikaci osm2pgsql a také uživatelský manuál k aplikaci operátora.

1. Systém pro sledování zásilek

Systém pro sledování zásilek bude sofistikovaný systém, učený pro úzkou koncovou skupinu uživatelů. Uživateli by měly být společnosti podnikající v doručování zásilek: pošty, soukromé doručovatelské firmy, kurýři. Navržený systém by měl pomáhat výše jmenovaným zákazníkům snižovat provozní náklady a tím zvýšit efektivitu a výnos jejich podnikání.

Systém by měl nabízet zákazníkovi kombinaci sledování pohybu vozidel přepravy, vizualizace přepravy a evidence osob či balíků. Měl by umožnit vhodnou optimalizaci trasy vozidel podle rozmístění příjemců a odesílatelů a pomáhat mimo jiné i při tvorbě knihy jízd. Pro snadnou orientaci operátora je důraz kladen na přehlednou vizualizaci. Volitelnou možností bude například zachycování obrazu z kamery a jiných čidel.

Kromě aplikace operátora by se měl balík aplikací skládat i z webových rozhraní, sloužících ke zvýšení komfortu práce a její efektivity jak na straně příjemců/odesílatelů zásilek, tak i řidičů a pracovníků depa. Příjemci zásilek budou moci zjistit přes webové rozhraní informace o balíku, např.: zda je balík uložen v depu, nebo momentálně převážen. Pokud je balík rozvážen, bude možné podle potřebných indicií vypočítat přibližný čas doručení. Tato informace bude velmi přínosná pro příjemce, protože dnes některé doručovatelské firmy příjemce předem neinformují o čase příjezdu. Pokud ano, potom většinou pouze oznámí, že přijedou v daný den. Příslušné informace ve webovém rozhraní by měly být během cesty dynamicky aktualizovány.

Pro řidiče a pracovníky depa budou připraveny webové aplikace sloužící pro evidenci balíků. Každý dopravce v současné době využívá pro identifikaci balíků speciální čárové kódy. Pokud budou řidiči a pracovníci depa vybaveni čtečkami čárového kódu s připojením na systém, operace s balíky bude otázkou několika sekund. Kromě toho dostane řidič možnost exportovat optimální trasu do formátu GPX, který lze použít v GPS navigaci. Přesně tak bude vědět, kam má zajet a po jaké trase. Funkce export GPX ušetří čas, odstraní nebezpečné bloudění a tím snižuje přepravní náklady.

Systém bude obsahovat i zajímavou funkci pro registrované uživatele dopravní firmy (odesílatel) při odesílání balíku. Pomocí webového rozhraní bude odesílatel moci jednoduše vytvořit požadavek na vyzvednutí balíku řidičem. Tato registrace bude následně ihned zahrnuta do optimalizace cesty pro rychlé vyzvednutí balíku.

1.1 Výhody systému

Hlavním kladem systému bude jeho sofistikace. Systém bude kombinovat více služeb do jednoho jedinečného balíku. Další výhoda aplikace bude spočívat v podpoře více operačních systémů, nejen Microsoft Windows. Tento fakt umožní společnostem ušetřit náklady na nákup případných licencí.

Automatizace. Většina funkcí systému bude pomáhat automatizovat interní procesy dopravní firmy. Například při optimalizaci se trasa bude generovat podle počtu odesílatelů a příjemců v kvadrantu řidiče. Výpočty časů příjemců budou taktéž generovány automaticky.

Výhodu pro zákazníky bude i skutečnost, že zákazník bude platit vlastníkům systému pouze za využívané služby. Mezi služby bude patřit například pravidelná aktualizace aplikací, správa databáze, rozšiřování služeb a také úpravy GPS klientů. V základu balíku budou serverové služby s garancí 24/7 běhu online. Vlastník sytému bude poskytovatelem webových služeb. Každá webová služba bude moci být zákazníkovi upravena přesně na míru.

Systém bude navržen pro provoz více klientů najednou. Klientské údaje budou pro větší bezpečnost důkladně odděleny. Řešení oddělní databázových dat je náročnější na fyzické úložiště. Nevýhoda oddělení dat je zároveň i bezpečnostní výhodou.

Balíček aplikací bude zajišťovat velkou bezpečnost přenosu zákaznických dat. Aplikace operátora sama o sobě nebude vlastnit přihlašovací údaje, ty budou načteny až po ověření u autorizačního serveru. Veškerá komunikace v systému bude navíc přenášena v šifrované podobě, aby ji nebylo možné napadnout.

2. Použité technologie

2.1 Global Position Systém

Global Position System (GPS) – globální polohový systém slouží pro určení polohy kdekoliv na zemském povrchu bez ohledu na počasí a na dobu, kdy se poloha měří. GPS byl navržen jako vojenský navigační systém. Budování započalo roku 1973 Ministerstvem obrany Spojených států. Plně funkčním systémem se stal až na začátku 90. let minulého století. Od té chvíle je dostupný po celém světě. Své využití nalezl v mnoha oborech lidské činnosti.

Systém se ukázal být tak přínosný, že Kongres Spojených států odsouhlasil výnos o používání systému GPS i v civilní sféře. Z počátku však byla omezena jeho funkčnost z důvodu možné zneužitelnosti. GPS systém byl nastaven tak, aby prvořadým uživatelem byla armáda Spojených států. Mezi nastavená omezení patří například jeho selektivní dostupnost (Selected Availibility). Pod pojmem selektivní dostupnost se rozumí záměrné zhoršování přesnosti polohy. Pro vojenské účely bylo zavedeno šíření P/Y kódu, který byl mnohem přesnější. Tato omezení byla zrušena 1.5. 2000. Přesnost určení polohy pro civilní aplikace se tak zvýšila až desetinásobně.

GPS systém je možné rozčlenit do 3 segmentů:

- kosmický,
- řídící (kontrolní),
- uživatelský

Kosmický segment

V době psaní diplomové práce je GPS systém tvořen 24 družicemi. Tři z těchto družic jsou záložní. Satelity krouží okolo Země ve výšce cca 20 tis. km na šesti oběžných drahách nakloněných vždy o 60°.

Každý satelit je vybaven přijímačem, vysílačem, atomovými hodinami a dalšími přístroji sloužícími pro navigaci a jiné úkoly (např.: detekci výbuchu jaderných náloží). Satelit komunikuje s řídícím centrem. Přijímá a zpracovává informace z centra sloužící pro korigování dráhy a zpětně centrum informuje o stavu vlastních systémů. Každý satelit je vybaven záložními zdroji. Baterie jsou dobíjeny slunečními panely.

Určování polohy probíhá následujícím způsobem. Satelit vysílá složitý signál pro uživatele. V tomto signálu jsou zakódované informace o poloze satelitu a přibližných polohách zbývajících satelitů systému GPS. Pro výpočet polohy musí GPS přijímač vypočítat pseudovzdálenost (vzdálenosti mezi přijímačem a viditelnými satelity). K výpočtu je potřeba znát rychlost šíření družicového signálu a rozdíl času mezi vysláním a příjmem signálu.

Pro výpočet dvoj-dimenzionální polohy stačí příjem signálu z minimálně tří satelitů. Pro výpočet polohy a nadmořské výšky je zapotřebí minimálně čtyř satelitů. Čím je vyšší počet satelitů viditelných nad obzorem, tím menší je chyba výpočtu a tím vyšší přesnost určení polohy.

Řídící segment

Řídící segment slouží k monitorování funkcí družic. V tomto segmentu jsou zjišťovány informace jako například dráha, či komunikace družic. Zajišťuje se zde i přesný chod atomových hodin na družicích. Získané údaje se zpětně předávají družicím. Tento segment tvoří hlavní řídící stanice v Colorado Springs. Patří sem pět monitorovacích stanic a tři pozemní řídící stanice. Všechny tyto stanice spolupracují s hlavní řídící stanicí.

Existují i zvláštní monitorovací sítě umožňující přesnější určování polohy hlavně pro přesné aplikace (geodézie, geodynamika). Tyto sítě se nepodílejí na řízení a činnosti systému GPS.

Řídící a kontrolní segment komunikuje s uživateli také prostřednictvím zpráv *GPS NANU (Notice Advisory to NAVSTAR Users)*, kde zveřejňuje plánované odstávky družic, jejich stažení a uvedení do provozu.

Pokud by došlo ke zničení pozemních vojenských stanic řídícího a kontrolního segmentu, přecházejí družice do režimu *AUTONAV* (Autonomous Navigation Mode), ve kterém jsou schopny dále pracovat až 6 měsíců. V tomto režimu spolu družice komunikují a porovnávají vzájemně mezi sebou své efemeridy (přesná data o poloze dané družice) a stav palubních hodin. Výsledky poskytují uživatelskému segmentu v navigační zprávě. Tento režim však nikdy nenastal, nejsou ani známy výsledky jeho případných testů.

Uživatelský segment

Aby bylo možné přijímat a zpracovat GPS signál, byly vyvinuty speciální přijímače. Uživatelé pak pomocí GPS přijímače mohou přijímat signál z jakékoliv družice, která je v určitou chvíli nad obzorem. GPS přijímač následně podle přijatých dat a předem definovaných parametrů vypočítá polohu a nadmořskou výšku. Mnozí uživatelé si myslí, že GPS přijímač dokáže i odesílat informace o své poloze. To je ovšem omyl, přijímač je pouze pasivní tzn.: pouze přijímá data ze satelitů. [20]

Hodnota nadmořské výšky je častěji zatížena chybou, než informace o poloze. Proto se v dnešní době pro výpočet nadmořské výšky často využívá jiných, přesnějších komponent, například snímače tlaku.

2.1.1 Přesnost GPS pozice

Přesnost GPS signálu závisí na změřené době letu signálu, na počtu viditelných satelitů a jejich geometrické konfiguraci. V dvoj-dimenzionálním případě je výsledná pozice dána průnikem tři kružnic. Známé jsou středy (pozice satelitů) i relativní rozdíly poloměrů těchto kružnic.

Protože však ale tato měření nejsou přesná, byl k výpočtu polohy použit průnik tří mezikruží. Výsledek průniku má nenulový obsah. Velikost obsahu záleží na chybě měření a vzájemné poloze satelitů.

Matematicky se pro práci s popsanou geometrickou nepřesností použijí místo průniků kružnic první členy Taylorovy řady v okolí bodu blízkého očekávaného průsečíku R_0. Touto záměnou se nahradí průnik mezikruží průnikem pásů. [17]

2.1.2 Komunikace SiRF III GPS čipsetu

Aby mohly GPS moduly komunikovat pomocí sběrnice s mikroprocesorem, byly vytvořeny speciální způsoby komunikace (algoritmy, protokoly).

Čip SiRF III je jedním z oblíbených GPS čipů. Patří mezi velmi spolehlivé, rychlé a přesné. Diplomová práce je testována právě s GPS čipem SiRF III.

SiRF III čip umožňuje komunikovat dvěma způsoby: prostřednictvím NMEA protokolu nebo binárního SiRF módu. GPS čip je schopen volně se přepínat mezi těmito způsoby komunikace. Obvykle přepínání probíhá pomocí specializovaného softwaru určeného pro správu SiRF III čipu (přepnutí umožňuje i aplikace GPSD popsaná v diplomové práci). NMEA protokol je univerzální komunikační protokol, využívaný i v jiných zařízeních než jsou GPS čipy. Je mu proto věnována samostatná kapitola.

Druhým komunikačním způsobem je binární SiRF mód. Tento způsob není tolik oblíbený jako NMEA protokol, protože aby mohl být využit, je nutné mu aplikaci postavit přesně na míru. Tím ale aplikace ztrácí svoji univerzálnost oproti komunikaci přes protokol NMEA. Binární mód má ale zásadní výhodu. Tento mód lze na rozdíl od protokolu NMEA využívat i na různé nastavení čipu. Binární SiRF mód lze použít i pro lepší komunikaci s GPS čipem po sběrnici, protože obsahuje větší množství atributů, které je možné z GPS čipu získat. Je ale dobré upozornit, že nastavení GPS čipu je složitá činnost a při neopatrném nastavení hrozí riziko přerušení komunikace.

2.1.3 NMEA Protokol

National Marine Electronics Association (NMEA – národní asociace pro námořní elektroniku) stojí za vytvořením standardu, definujícím rozhraní mezi elektronickými zařízeními používanými pro lodní dopravu. NMEA slouží pro výměnu informací mezi počítačem a zařízeními, třeba GPS [17].

Protokol NMEA-0183 zasílá informace po řádcích. Jako první znak řádku je znak „$“, následován dvojící písmen GP označující „Global Position“. Zbylá tři písmena určují druh zasílané zprávy. Posledními znaky řádku jsou: hvězdička a kontrolní hexadecimální součet (XOR všech znaků na řádku mezi '$' a '*'). Řádek může být maximálně 80 znaků dlouhý. Oddělovacím znakem položek je čárka.

Protokol NMEA má definovaných několik vět. Pro GPS jsou nejzajímavější čtyři z nich:

- RMC – Základní informace o pozici,
- GGA – rozšířené informace o pozici (nadmořská výška, počet satelitů, odhadovaná chyba),
- GSA a GSV – informace o satelitech.

RMC (*Recommended minimum specific GPS/Transit data)*:

```
$GPRMC,181038.0,A,5006.3171,N,01425.6622,E,0.00,42.00,150606,,*37
```

181038.0	Čas, konkrétně 18:10:38.0 UTC.
A	Status (A=active, V=void platná či neplatná pozice)
5006.3171,N	Latitude: 50 stupňů, 6.3171 minut, severní šířky
01425.6622,	Longitude: 14 stupňů, 25.6622 minut, východní délky
0.00	Rychlost v námořních uzlech
42.00	Azimut ve stupních
150606	Datum 15. června 2006
	Chybí záznam o magnetickém rozptylu
	A jeho směru (W nebo E)

GGA (*Global Positioning System Fix Data)*:

```
$GPGGA,161016.000,5005.0334,N,01430.3369,E,1,06,2.1,242.9,M,45.5,M,,0000*5C
```

161016.000	Čas
5005.0334,N	Souřadnice, stejné jako RMC
01430.3369,E	Souřadnice, stejné jako RMC
1	Kvalita pozice (0 invalid, 1 GPS, 2 DGPS)
06	Počet viditelných satelitů
2.1	Relativní chyba v horizontálním směru (Horizontal Dilution of Precision)
242.9,M	Nadmořská výška v metrech
45.5,M	Výška geoidu nad WGS84 elipsoidem (45.5m)
	Chybí doba od poslední DGPS korekce
0000	Chybí ID referenční DGPS stanice

GSA (*GPS DOP and Active Satellites*) a GSV (*GPS Satellites in View*):

```
$GPGSA,A,3,26,08,18,29,28,09,,,,,,,2.9,2.1,1.9*3F
```

A,3	Automatický mód s 3D pozicí (jiné možnosti: M = Manual, 1 = chybí pozice, 2 = pouze 2D pozice)
26,08,18,29,28,09,,,,,,,	kódy viditelných satelitů (maximálně 12)
2.9	chyba pozice (Positional Dilution of Precision)
2.1	horizontální chyba (Horizontal Dilution of Precision)
1.9	vertikální chyba (Vertical Dilution of Precision)

```
$GPGSV,3,1,09,29,82,232,26,26,77,279,33,28,57,070,32,08,29,077,24*77
$GPGSV,3,2,09,18,22,314,25,09,21,269,18,17,20,128,11,10,16,197,*71
$GPGSV,3,3,09,27,06,085,11*4E
```

3,1	Celkový počet zpáv 3, toto je zpráva 1
09	Počet satelitů ve výhledu
29,82,232,26	PRN číslo satelitu, elevace ve stupních (90max), azimut 000..359, síla signálu dB 00..99
26,77,279,33	Stejný význam jako u předešlé čtveřice
28,57,070,32	Stejný význam jako u předešlé čtveřice
08,29,077,24	Stejný význam jako u předešlé čtveřice

2.1.4 Knihovna jazyka C libgpsd

Pro práci s GPS v GNU/Linux slouží systém GPSD a knihovna *libgpsd*. Jedná se o dvě podobná řešení. Knihovna *libgpsd* dává k dispozici přímý přístup k hardware GPS modulu, systém GPSD přidává určitou mezivrstvu.

V případě, kdy programátor chce přistupovat svým programem přímo k GPS modulu bez zmíněné mezivrstvy, pak by jeho volbou měla být právě knihovna *libgpsd*. GPSD je program napsaný právě pomocí knihovny *libgpsd*.

Aby bylo možné *libgpsd* použít je, třeba použít vkládací direktivu *#include <gpsd.h>* .

```
int gpsd_open_dgps (char * dgpsserver);
void gpsd_init (struct gps_device_t *session, struct * gps_context_t *,
char * device);
int gpsd_activate (struct gps_device_t *session, bool reconfigurable);
void gpsd_deactivate (struct gps_device_t * session);
gps_mask_t gpsd_poll (struct gps_device_t * session);
void gpsd_wrap (struct gps_device_t * session);
void gpsd_report (int d, const char * fmt, ...);
```

Jak už bylo uvedeno, *libgpsd* nabízí nízkou úroveň práce s GPS modulem. Nízká úroveň programování přináší výhody i nevýhody. Podstatnou výhodou je minimální režie na provoz, protože běží pouze jeden program. Mezi nevýhody patří uvolňování deskriptorů GPS modulů. Když dojde k násilnému odpojení GPS modulu od sběrnice, nedojde k uvolnění deskriptoru. Jakmile má deskriptor GPS modulu nějaká aplikace otevřený, pak je zamčený operačním systémem. A proto nelze deskriptor uzavřít.

Jakmile dojde k uvolnění deskriptoru aplikací, odemkne se, uzavře se a připraví na nové použití. Problém vzniká právě když je deskriptor uzamčen a zároveň dojde k opětovnému připojení GPS modulu ke sběrnici. Operační systém tento problém vyřeší přidělením nového deskriptoru, o kterém ale program neví.

Aby nedošlo k výše popsané situaci, je vhodné ihned po odpojení GPS modulu ukončit program. Třeba s chybou špatného čtení. Při ukončení dojde k odemčení deskriptoru a jeho připravení pro nové použití.

Celý tento proces je velkou nevýhodou použití nízké úrovně knihovny *libgpsd*. Řešení všech podobných výjimek, podobné té s deskriptory je ponecháno pouze na programátorovi. Pokud dochází k častému odpojování GPS modulu a problém je řešen ukončováním programu, zvyšuje se režie z důvodu opětovného startování aplikace voláním systémového volání *fork()*.

Problém je možné eliminovat použitím notifikačního procesu operačního systému. K opakované informaci o připojení/odpojení GPS modulu dodá notifikace informaci i o tom, na jaký deskriptor byl GPS modul přidělen. V této situaci se jedná o nejvhodnější, ale složitější, alternativu.

Knihovna *libgpsd* má ještě jednu nevýhodu. Pokud je k programu připojováno více druhů GPS modulů (komunikující jinými protokoly než NMEA), je třeba sestavit inicializační a konfigurační struktury pro každý specifický typ GPS modulu. Aplikace GPSD již naopak s tímto případem dopředu počítá.

2.1.5 Systém GPSD a libgps

GPSD je alternativou k předchozímu způsobu práce s GPS. Na rozdíl od *libgpsd* komunikuje přímo s GPS modulem pouze démon GPSD. Tento démon je podobný jakékoliv jiné systémové službě. Běží na pozadí a naslouchá GPS modulu a naslouchá na specifickém socketu. Programové rozhraní obsahuje v knihovna *libgps*.

Výhodou GPSD je, že tvoří určité abstraktní rozhraní pro různé typy GPS modulů. Aplikace zpracovávající GPS údaje pak lze úplně oprostit od problémů spojených se specifickými údaji a nastavením modulů. GPSD po spuštění komunikuje s programem napsaným pomocí knihovny *libgps* přes síť. Proto je možné GPSD využít i pro vzdálenou komunikaci.

Aplikace GPSD dokáže i vstřebat problémy s odpojením a následným připojením GPS modulů za běhu aplikace. Funguje na principu, že po odpojení modulu dojde k uvolnění deskriptoru zařízení v adresáři */dev*. Po připojení se opět tento deskriptor otevře. Inicializaci otevření dostane od systému notifikace. Problém nastane, pokud se náhodou nepodaří odpojit deskriptor. Pak dojde k inicializaci nového deskriptoru a program GPSD se o této změně nemusí dozvědět. To lze částečně eliminovat správným výběrem přepínačů služby.

```
usage: gpsd [-b] [-n] [-N] [-D n] [-F sockfile] [-G] [-P pidfile] [-S port]
[-h]
device...
  Options include:
  -b                        = bluetooth-safe: open data sources read-only
  -n                        = don't wait for client connects to poll GPS
  -N                        = don't go into background
  -F sockfile               = specify control socket location
  -G                        = make gpsd listen on INADDR_ANY
  -P pidfile                = set file to record process ID
  -D integer (default 0)    = set debug level
  -S integer (default 2947) = set port for daemon
  -h                        = help message
  -V                        = emit version and exit.
A device may be a local serial device for GPS input, or a URL of the form:
     {dgpsip|ntrip}://[user:passwd@]host[:port][/stream]
     gpsd://host[:port][/device][?protocol]
```

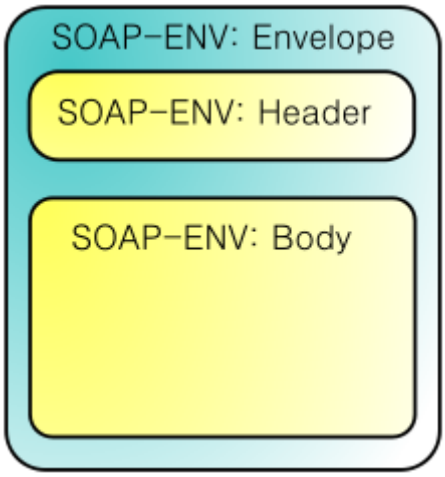
Podporované formáty a protokoly GPS: *Generic NMEA, Ashtech, Delorme TripMate, Delorme EarthMate (pre-2003, Zodiac chipset), Furuno Electric GH-79L4, Garmin NMEA, MTK-3301, San Jose Navigation FV18, AIVDM, EverMore bingy, Garmin USB binary, Garmin Serial bingy, iTalk bingy, oncore binary, Navcom binary, SiRF bingy, SuperStarII binary, Trimble TSIP, uBlox UBX bingy, Zodiac bingy, RTCM104V2, RTCM104V3, Garmin Simple Text*. Jak je vidět, GPSD podporuje velkou řadu formátů.

K tomu, aby bylo možné tento systém použít, je potřeba použít vkládací direktivu *#include <gps.h>*.

```
struct gps_data_t *gps_open (char *server, char * port);
int gps_query (struct gps_data_t *gpsdata, char *fmt...);
void gps_set_raw_hook (struct gps_data_t *gpsdata, void (*hook)(struct
gps_data_t *, char *buf));
int gps_poll (struct gps_data_t *gpsdata);
void gps_close (struct gps_data_t *gpsdata);
void gps_set_callback (struct gps_data_t *gpsdata, void (*callback)(struct
gps_data_t *sentence, char *buf), pthread_t *handler);
void gps_del_callback (struct gps_data_t *gpsdata, pthread *handler);
void rtcm_unpack (struct rtcm_t *rtcmp, char *buf);
```

2.2 Simple Object Access Protocol

Při rozhodování, jaký typ datového přenosu mezi GPS klientem -> serverem -> aplikací operátora použít, padlo rozhodní na protokol SOAP (Simple Object Access Protocolu). Tento protokol je založen na dnes velmi oblíbeném souboru XML. Díky tomu je struktura SOAP přenosu poslední dobou zvlášť mezi informačními systémy velmi rozšířená. Další výhodou protokolu je, že komunikace probíhá přes webový server. V řadě případů je možné použít speciální webový server, který je už upraven pro SOAP komunikaci. V jiném případě stačí sestavit a doinstalovat potřebný modul.



Obr. 1: Struktura SOAP zprávy [9]

Formát SOAP tvoří základní vrstvu komunikace mezi webovými službami a poskytuje prostředí pro tvorbu složitější komunikace.

Existuje několik různých druhů šablon pro komunikaci na protokolu SOAP. Nejznámější z nich je RPC šablona, kde jeden z účastníků komunikace je klient a na druhé straně je server. Server ihned odpovídá na požadavky klienta.

SOAP je nástupce XML-RPC. A zapůjčuje si jeho způsob přenosu dat a další vlastnosti. Obálka, hlavička a tělo komunikace je ale pravděpodobně z WDDX.

Původně ho navrhli Dave Winer, Don Box, Bob Atkinson a Mohsen Al-Ghosein v roce 1998 za podpory firmy Microsoft (kde tou dobou Atkinson a Al-Ghosein pracovali). Dnes je SOAP specifikace držena XML skupinou tvořící internetové protokoly z W3C konsorcia. [9]

Příklad požadavku protokolu SOAP:

```
<SOAP-ENV:Envelope
xmlns:SOAPENV="http://schemas.xmlsoap.org/soap/envelope/">
 <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
   <GetPosition xmlns="urn:position">
    <id>db0597abc29271f2a2f4db6981a810a6dc610fe8</id>
   </GetPosition>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Z požadavku je zřejmé, že se jedná o volání metody GetPosition. Tato metoda přijímá parametr id a vrací zpět pozici GPS klienta tohoto id. Z příkladu vyplývá, že se jedná o požadavek s XML strukturou.

Vše ale není tak pozitivní, jak se na první pohled zdá. Mince má dvě strany. Na jedné straně je XML struktura velice flexibilní a přehledná, ale na druhé straně probíhá komunikace v podobě čistého textu. Útočníkovi tím poskytuje možnost velmi zajímavého počtení. Proto W3C konsorcium navrhlo standardizaci speciálního XML zabezpečení, známého pod spojením XMLSec (XML Security). Tento systém funguje na principu transformace těla SOAP dokumentu pomocí klíče. Jedná o šifrování, dešifrování a podepisování. Programátor však naráží na problém různých implementací v různých knihovnách.

2.2.1 Klient a knihovna jazyka C libcsoap

V aplikaci GPS klienta byla použita knihovna *libcsoap*. Byla vybrána z důvodu dostatečné dokumentace a zároveň logičtějšího používání než konkurenční knihovny, spojené se serverem AXIS2.

Knihovna *libcsoap* je na GNU/Linux obvykle přítomna přímo v repositářích. Škoda, že obvykle není k dispozici v nejnovější verzi (novější verze mají větší funkcionalitu: disponuje například šifrovacím standardem *xmlsec*). Je dobré uvést, že knihovna *libcsoap* potřebuje ke své funkci také knihovny *nanohttp*, *libxml2* a *libssl*.

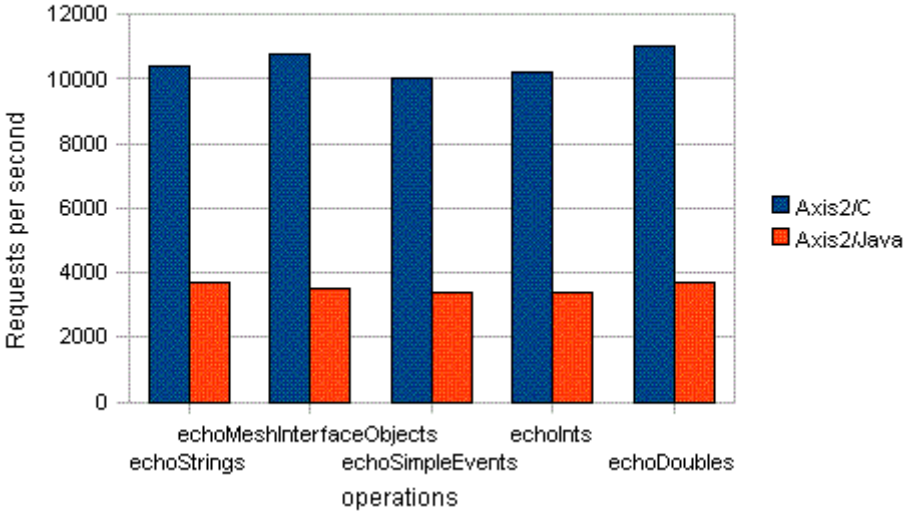
Přes tuto knihovnu je možné zasílat kromě klasických informací i přílohy jako soubory. Problém spočívá v tom, že konkurenční knihovny SOAP protokolu mají tuto funkci implementovanou trochu jinak.

2.2.2 Server AXIS2/c a jeho knihovny

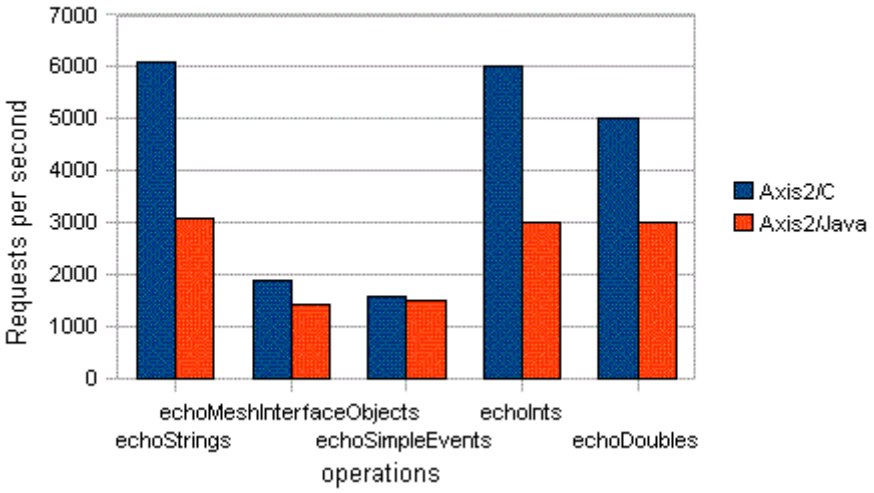
Přestože knihovna *libcsoap* má možnost tvořit SOAP služby pomocí CGI skriptů, není tato volba optimální. V dnešní době je velmi oblíbený server webových služeb AXIS2. AXIS2 je postaven na upraveném webovém serveru Apache 2, rozšířeném o moduly serveru AXIS2.

Projekt serveru AXIS2 se ještě dělí na dva dost odlišné druhy:

- Prvním druhem je server postavený na programovacím jazyce Java Axis2/Java. Kód služby pro tento typ serveru je logicky uspořádaný, knihovny jsou dobře popsané. Ovšem při výměně velkého množství dat nejsou moc optimální, protože programovací jazyk Java nepatří mezi nejrychlejší.
- Druhý typ serveru AXIS2 je Axis2/C. Jak už písmeno za lomítkem napovídá, jedná se o server postavený na programovacím jazyce C. Tento programovací jazyk je oproti Javě mnohem rychlejší. Proto je rychlejší i celý Axis2/C server. Server má ale jednu zásadní nevýhodu. Knihovny pro psaní služeb serveru nemají téměř žádnou dokumentaci. To je avšak malá daň za podstatně vyšší rychlost.

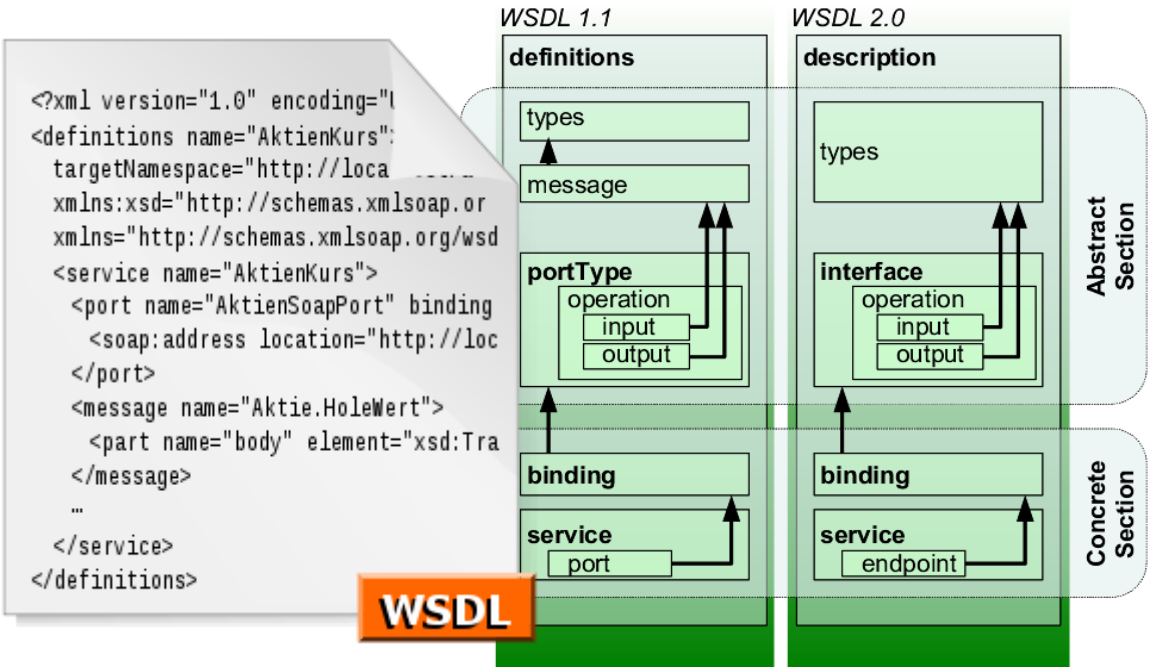


Obr. 2: Porovnání rychlostí Axis2/C a Axis2/Java [11]



Obr. 3: Porovnání rychlostí Axis2/C a Axis2/Java [11]

Jak je z obrázků vidět, až na volání určitých metod je rozdíl rychlostí opravdu vysoký. A přesně to je pro efektivnost programu o velké vytíženosti rozhodující. Axis2/C disponuje malým problémem oproti Axis2/Java. Problém spočívá v automatickém generování souboru *wsdl* (Web Services Description Language) definujícím služby serveru. U Axis2/C je programátor omezen pouze na základní informace o službě. *Wsdl* je možno manuálně vygenerovat, ale to není tak rychlé. Vzhledem k tomu, že nikdo nemusí vidět detailní informace o službách, není dobré *wsdl* generovat. V tomto případě by byl soubor *wsdl* spíš ke škodě, než k prospěchu.



Obr. 4: Schéma WSDL souboru [10]

Při nasazení Axis2/C serveru existuje možnost kompilace celého serveru ze zdrojových kódů, nebo sestavení speciálního modulu pro již funkční Apache 2 server. Pokud nasazujeme pouze novou službu na funkční server, představuje druhá možnost nejlepší volbu. Instalaci popisuje následující skript:

```
#!/bin/bash

apt-get install apache2 apache2-dev
wget http://trixes.net/pub/ws/axis2/c/1_6_0/axis2c-src-1.6.0.tar.gz
tar -xzf axis2c-src-1.6.0.tar.gz

cd axis2c-src-1.6.0/
```

```
./configure --enable-openssl=yes --with-apache2="/usr/include/apache2/"  --
with-apr="/usr/include/apr-1.0/" && make && make install


ln -s /usr/local/axis2c/lib/libmod_axis2.so
/usr/lib/apache2/modules/mod_axis2.so


echo "LoadModule axis2_module /usr/lib/apache2/modules/mod_axis2.so
Axis2RepoPath /usr/local/axis2c
Axis2LogFile /usr/local/axis2c/axis2.log
Axis2MaxLogFileSize 1024
Axis2LogLevel LOG_LEVEL
<Location /axis2>
SetHandler axis2_module
</Location>" >> /etc/apache2/httpd.conf
```

Ve zkratce - nejdříve se nainstalují potřebné knihovny a stáhnou se zdrojové kódy SOAP serveru. Následně se zkonfigurují tak, aby se kompiloval pouze modul. Sestavený modul se aplikuje na správné místo a posledním krokem je konfigurace modulu do souboru `httpd.conf`. Tento soubor je v distribuci Ubuntu považován za sekundární konfigurační soubor.

Axis2/C se tímto způsobem nainstaluje do `/usr/local/axis2c`. V tomto adresáři se nachází konfigurační soubor `axis2.xml`, složka services. Do složky services se umísťují služby serveru. Pod pojmem služba serveru se rozumí speciální knihovna, která implementuje potřebné rozhraní. Hierarchie pokračuje složkou `/usr/local/axis2c/nazev_sluzby` s názvem služby. V této složce se nacházejí soubory `services.xml` s nastavením služby a knihovny služby.

Služby je možné kompilovat např. takto:

```
gcc -Wall -fPIC -g -D_REENTRANT -shared -olib$JMENO.so.0.0.0 -
I../../include/axis2-1.6.0/ -L../../lib -laxutil -laxis2_axiom -
laxis2_parser -laxis2_engine -lpthread -laxis2_http_sender -
laxis2_http_receiver -lpq $JMENO.c camera_skel.c
../../build/gscrypt/Debug/libgscrypt.a ../../build/gsfile/Debug/libgsfile.a
```

Soubor services.xml má následující strukturu:

```
<service name="camera">
    <parameter name="ServiceClass" locked="xsd:false">camera</parameter>
   <description>
        Service which sets Camera Information.
   </description>
    <operation name="SetCamera">
    </operation>
</service>
```

Když je vytvořena služba, stačí restartovat webový server pro aplikaci služby. Na odkazu http://www.example.ltd/axis2/services/ se po restartu nachází seznam vytvořených služeb.

2.2.3 SOAP knihovny programovacího jazyka Java

Programovací jazyk Java je velmi oblíbený pro komunikaci přes protokol SOAP, proto obsahuje už výchozí integrované mechanismy pro tento druh komunikace.

```
import javax.xml.soap.*;
```

Po importu je možné snadno odeslat zprávu:

```
SOAPConnectionFactory soapConnFactory =
SOAPConnectionFactory.newInstance();
SOAPConnection connection = soapConnFactory.createConnection();

MessageFactory messageFactory = MessageFactory.newInstance();
SOAPMessage message = messageFactory.createMessage();

SOAPPart soapPart =     message.getSOAPPart();
SOAPEnvelope envelope = soapPart.getEnvelope();

SOAPBody body =         envelope.getBody();
SOAPElement bodyElement =  body.addChildElement(METODA ,"", URN));

bodyElement.addChildElement("id").addTextNode(gpsHash);

message.saveChanges();
message.writeTo(System.out);

SOAPMessage reply =  connection.call(message,URL);
```

Z odpovědi reply je možné sestavit seznam uzlů, který se následně zpracovává. Předchozí kód zabalí zprávu o jediném elementu id a odešle zprávu na URL voláním metody METODA.

Je zřejmé, že integrované mechanismy programovacího jazyka Java jsou velmi intuitivní a logicky uspořádané. Opět má knihovna integrovanou možnost přijímat a odesílat přílohy. Synchronizace příloh s Axis2/C byla přesto z důvodu nekompatibility implementace neúspěšná.

2.3 OpenStreetMap.org otevřené mapy

Pro mapový podklad pro aplikaci byla použita data z webové aplikace openstreetmap.org. Tato data patří pod jednu z liberálních multimediálních licencí Creative Commons Attribution-ShareAlike 2.0 license. API aplikace je pod licencí GNU General Public License. Obě licence dovolují volné použití.

Aplikace openstreetmap.org slouží k tvorbě celosvětové mapy přímo uživateli webu. Proto je přesnost mapy úměrná počtu kvalitních přispěvovatelů v oblasti. Ve městech se přesnost mapy dá srovnat s komerčními mapovými podklady, kterými disponuje například firma Seznam na webu mapy.cz.

K nasazení na vlastní server ale pouze liberální licence nestačí. Kvalitní API systému dovoluje provádět denní snapshot databáze do speciálního souboru XML s koncovkou *osm*. Soubory s koncovkou *osm* lze pak využít pro import mapových podkladů do vlastní databáze. K ulehčení práce s obrovskými XML soubory denních snapshotů je možné najít na Internetu i různé skupiny, které rozčlení celkový XML na menší snapshoty týkající se určitého území. V případě diplomové práce území České Republiky. Tento předpoklad je důležitý, protože snapshot Evropy měl před rokem cca 20GB! Současný snapshot České Republiky má velikost okolo jednoho gigabyte.

Oproti konkurenci, např. firmě Garmin, obsahuje mapa pouze dvou-dimenzionální data. Databázový systém je připraven i na vkládání dat ve třech dimenzích. Zajímavé by bylo, pokusit se zkombinovat openstreetmap.org mapy s mapovým podkladem určeným pro zobrazení členitosti terénu. Kombinace mapových podkladů by tak byla vhodná pro zjišťování doplňujících informací o trase a vozidle, třeba závislosti spotřeby na členitosti terénu. Zjištěná data poskytují možnost vylepšit generování optimální trasy z hlediska spotřeby v závislosti na profilu trasy.

2.3.1 Aplikace pro OpenStreetMap.org

Nejdůležitější aplikací pro openstreetmap.org je *osm2pgsql*. Tato utilita dokáže importovat a exportovat údaje z/do XML souboru do/z databázového systému PostgreSQL (PostGIS). Nápověda k aplikaci je v diplomové práci přiložena jako příloha 6.2.

2.3.2 Formát souboru OSM

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="OpenStreetMap server">
  <bounds minlat="51.5073601795557" minlon="-0.108157396316528"
maxlat="51.5076406454029" maxlon="-0.107599496841431"/>

  <node id="275452090" lat="51.5075933" lon="-0.1076186" version="3"
changeset="2980587" user="nickb" uid="1697" visible="true" timestamp="2009-
10-29T12:14:35Z">
    <tag k="name" v="Jam's Sandwich Bar"/>
  </node>

  <way id="27776903" visible="true" timestamp="2009-05-31T13:39:15Z"
version="3" changeset="1368552" user="Matt" uid="70">
    <nd ref="275452090"/>
    <nd ref="275452091"/>
    <tag k="access" v="private"/>
  </way>
</osm>
```

Z výpisu struktury souboru *osm* je na první pohled vidět, že se jedná o speciální XML soubor obsahující specifické elementy.

První řádek souboru je deklarace, že se jedná o XML soubor. Jako atributy jsou uvedeny: verze a kódování.

Osm – element se nachází na druhém řádku. Tento element deklaruje, o jaký typ souboru *osm* se jedná. Atributy elementu: první atribut říká že se jedná o verzi 0.6 (tato verze odpovídá verzi použitého API), druhý atribut informuje o způsobu, jak byl soubor generován.

Bounds – znamená hranice. Tato hranice ohraničuje kvadrant, který obsahuje všechny body v souboru *osm*. Atributy znázorňují dva hraniční vrcholy kvadrantu, geometricky tvořící úhlopříčku hranice.

Node – tento element je hlavním stavebním kamenem struktury. Geometricky je to znázornění určitého uzlu (bodu), ze kterého se mohou, ale nemusí, stavět složitější útvary, jako třeba polygon. Mezi nejdůležitější atributy elementu patří *id*, *lon* a *lat*. *Id* určuje jedinečné identifikační číslo uzlu, *lon* a *lat* znázorňují GPS polohu uzlu. Ostatní atributy jsou pouze informační: kdo a kdy vytvořil daný uzel.

Tag – je doplňkovým elementem elementů *node* a *way*. *Tag* doplňuje hlavní elementy o informace. Například, pokud je uzel městem, doplňuje jej o název. *Tag* je složen z dvojic atributů *k* a *v* (klíč a hodnota).

Way – tento element zapouzdřuje sekvenci uzlů. Toto zapouzdření tvoří složitější geometrické struktury. Nejdůležitějším atributem elementu je id. Stejně jako u elementu node se jedná o jedinečné identifikační číslo. Ostatní atributy jsou opět pouze informační.

Nd – odkazuje pomocí atributu *ref* na *id* elementu *node*. Sekvence elementů *nd* tvoří složitější strukturu, jako například polygon.

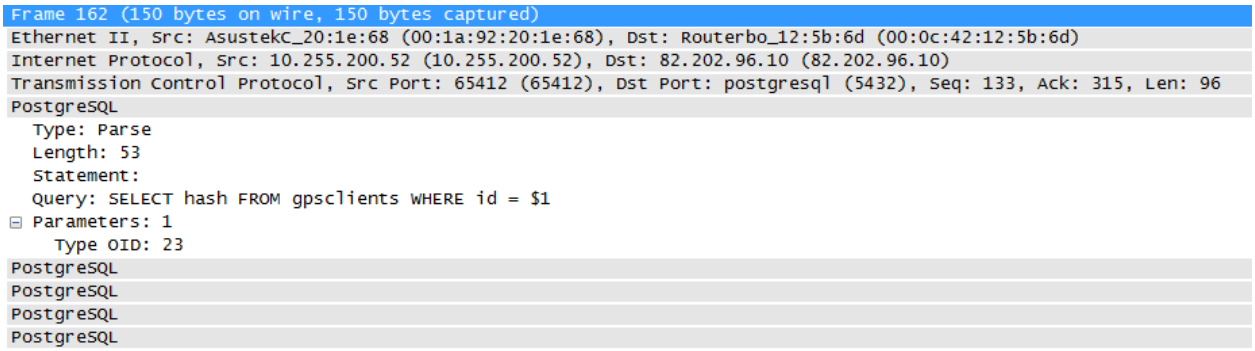
2.4 Databáze

Jako databázový systém projektu je použit PostgreSQL. PostgreSQL podporuje jazyk PL/SQL, který dává databázovému systému lepší flexibilitu. PostgreSQL je navíc pod licencí GNU General Public License.

Alternativou při výběru databázového systému pro server byl databázový systém firmy Oracle, který je výkonnější. Protože se ale nejedná o bezplatný systém, tak byl zamítnut.

Databázový systém PostgreSQL (aktuální verze 8.4) je pro potřeby programu dostatečně podporován. V programovacím jazyce C knihovnou *libpq* a v programovacím jazyce Java knihovnou *jdbc4* ve verzi 8.4.

Určitým nedostatkem PostgreSQL je komunikace, která probíhá ve výchozím nastavení nešifrovaně. Pro vyšší bezpečnost komunikaci je možné nastavit pomocí SSL vrstvy. Komunikace pod SSL vrstvou je dostatečně bezpečná pro přenos informací sítí Internet.



Obr. 5: Odchycení komunikace s PostgreSQL nešifrovanou cestou aplikací *Wireshark*

2.4.1 PostGIS

Databázový systém sám o sobě není vhodný pro uchovávání GIS (*Geographic information system*) dat, jako jsou např. silnice. Z tohoto důvodu bylo pro databázové systémy navrženo několik balíků (*spatial*) funkcí pro práci s geografickými daty. Firma Oracle má k dispozici příslušný balík pro svůj databázový systém, stejně tak existuje podobný balík i pro PostgreSQL.

V případě databázového systému PostgreSQL se jedná o balík jménem PostGIS. Tento byl vybrán hlavně proto, že je využit přímo na openstreetmap.org. Jejich databázové API je na už PostGIS připraveno.

Moderní GNU/Linux distribuce mají PostGIS mezi svými balíčky. Stejně jako balík PostGIS je potřeba nainstalovat i doplňkové aplikace, které zajistí nahrání geografických dat z exportovaného XML souboru přímo do databáze. K nim patří např. *osm2pgsql*. K dnešnímu dni jsou aktuální verze aplikací:

- PostGIS – postgresql-8.4-postgis_1.5.1-5,
- osm2pgsql – osm2pgsql_0.66.20090526-1ubuntu1.

Postup instalace a nahrání dat do systému PostgreSQL s balíkem PostGIS:

```
#!/bin/bash

wget http://download.geofabrik.de/osm/europe/czech_republic.osm.bz2
sudo apt-get install postgis osm2pgsql

droplang plpgsql osm
dropdb osm
createdb osm
createlang plpgsql osm


psql -dosm -f /usr/share/postgresql/8.4/contrib/postgis.sql
psql -dosm -f /usr/share/postgresql/8.4/contrib/spatial_ref_sys.sql
psql osm < /usr/share/postgresql/8.4/contrib/_int.sql

osm2pgsql -s -U peny -d osm -p osm -l czech_republic.osm.bz2
```

Pomocí výše uvedeného skriptu se docílí vytvoření databáze *osm*. V této databázi se aplikuje nastavení nadstavby PostGIS (vytvoření funkcí, struktur). V posledním kroku se spustí aplikace *osm2pgsql* pro nahrání XML souboru geografických dat do databáze. Mezi nejdůležitější přepínače aplikace patří přepínač „s“. Volbou „s“ se program přepne do režimu, který nevyžaduje velké množství paměti. Aplikace nahrává data do databázového systému po menších blocích. Bez tohoto přepínače dojde i na serveru o 2GB RAM velmi rychle volná paměť. Druhou zajímavou volbou je „*l*“. To zajistí, aby se do databáze vkládaly souřadnice ve stejném formátu jako v GPS. Bez přepínače „l“ by se souřadnice ukládaly ve zvoleném typu geografické projekce. Souřadnice ve formátu GPS představují nejlepší a nejuniverzálnější možnost.

Pokud je potřeba pro kompatibilitu XML souboru `czech_republic.osm.bz2` mít nejnovější verzi aplikace *osm2pgsql*, je možné ji nainstalovat ze zdrojových kódů:

```
sudo apt-get autoconf install subversion libxml2-dev libbz2-dev libproj-dev
libgeos-dev build-essential

svn co http://svn.openstreetmap.org/applications/utils/export/osm2pgsql/

cd osm2pgsql/
./autoconf.sh
./configure
make
sudo make install
```

Je vidět, že instalace ze zdrojových kódů není složitá. Někdy je opravdu potřeba takovou instalaci podstoupit, protože verze API XML souboru *osm* se vyvíjí rychleji, než distribuční balíček aplikace.

Po nahrání geografických údajů do databáze se vytvoří tři tabulky, osm_point, osm_line, osm_polygon. Všechny mají podobnou strukturu:

oms_id	ID určitého uzlu
parametry jako name, highway	Parametry určující typ uzlu
z_order	Index zobrazení
way	Datový typ *geometry* která má zakódované veškeré body uzlů (line a polygon jich mají více, point pouze jeden)

Tabulka *point* struktura *way* (*geometry*) obsahuje pouze jednu souřadnici. Tabulka *line* obsahuje více bodů a stejně tak souřadnic. Tyto body spolu geometricky udávají lomenou čáru. Tabulka *polygon* obsahuje více souřadnic bodů, které tvoří uzavřený x-úhelník. Tento způsob uchovávání dat je možný pouze s balíkem PostGIS. Už z popisu je jasné, že složit a rozložit strukturu *way* (*geometry*) není složité. K tomuto účelu obsahuje PostGIS řadu předem definovaných funkcí.

Je dobré uvést konkrétní příklad výhody systému PostGIS. Je jím rozdíl mezi starším openstreetmap.org API, které využívalo jako databázový systém MySQL bez jakéhokoliv GIS balíku a současným API verze 0.6. Nahrání cca stejného množství dat do MySQL systému trvá zhruba čtyři hodiny. Kdežto do současného API, postaveném na PostgreSQL a s nadstavbou PostGIS, zhruba jednu hodinu. Z příkladu je vidět, že nárůst efektivity novějšího API je zřejmý.

2.4.2 Java knihovna PostGIS

Pro geografický balík databázového systému PostGIS existuje doplňková knihovna pro programovací jazyk Java. Knihovna definuje nové datové typy *PGgeometry* a umožní snadnou konverzi datových typů GIS nadstavby do grafických datových typů vestavěných v grafických knihovnách jazyka Java.

```
import org.postgis.*;

((org.postgresql.Connection)conn).addDataType("geometry","org.postgis.PGgeo
metry");
((org.postgresql.Connection)conn).addDataType("box3d","org.postgis.PGbox3d"
);
```

Nejprve se importuje potřebný namespace. Ihned po vytvoření spojení *conn* je zapotřebí zajistit, aby spojení počítalo s existencí datových typů se strukturami *PGgeometry* a *PGbox3d*.

```
PGgeometry geom = (PGgeometry)r.getObject(1);
```

Výpis zobrazuje načtení PGgeometry z dotazu v databázovém systému. Následně je možné pracovat s *PGgeometry* stejně jako s vestavěnými datovými typy programovacího jazyka Java. Výhodná je následná detekce typu geometrie. Lze takto zjistit, o jaký druh databázového datového typu *geometry* se jedná. Podle typu se pak vybírá správné chování geometrického tvaru.

V programu byla použita knihovna development snapshot verze 2.0.0.

2.5 The Standard Widget Toolkit

Při výběru hlavní grafické knihovny programovacího jazyka Java, pomocí které je postavena celá aplikace operátora systému, se nabízely možnosti: SWING, AWT a SWT. Ten, kdo někdy programoval grafické prostředí (GUI) v programovacím jazyce Java ví, že hlavní nevýhodou grafického rozhraní v Javě je rychlost. To je způsobeno tím, že veškeré grafické operace se zpracovávají na úrovni JVM (Java Virtual Machine).

Knihovna, která by byla provázána přímo s grafickým rozhraním operačního systému a tím částečně obcházela JVM, by umožnila celé grafické prostředí výrazně urychlit. S tímto nápadem přišla firma IBM a vytvořila knihovnu SWT. Nyní je SWT ve správě nadace Eclipse Foundation.

K vykreslovaní prvků GUI používá SWT nativní knihovny operačních systémů prostřednictvím JNI (Java Native Interface). Bohužel využití nativních knihoven, zajišťující rychlost SWT, je v jiném případě nevýhodou. Při namapování knihovny na nativní knihovny operačního systému dochází k situaci, kdy je SWT plně závislá na dané platformě. V případě GNU/Linux je i potřeba mít nainstalované grafické knihovny GTK+ a X server.

Knihovna je licencovaná pod Eclipse Public License.

Podle Eclipse Foundation, "SWT a Swing jsou různé nástroje vytvořené s různými cíli. Cílem SWT je poskytovat API pro přístup k nativním grafickým prvkům napříč širokým spektrem platforem. Hlavními cíli návrhu jsou: velká rychlost, nativní vzhled a hluboká platformní integrace. Swing je naproti tomu navržen tak, aby umožňoval vysoce přizpůsobitelný vzhled, který je pak stejný na všech platformách." [5]

SWT patří mezi jednodušší grafické knihovny. Nejsou zde obsaženy žádné nadstandardní funkce. Někdo by mohl říci, že SWT postrádá oproti Swingu nějakou funkčnost. Však právě odlehčení je jednou z výhod SWT, které bylo navrženo tak, aby bylo velmi výkonné, rychlejší, lehčí a s lepší schopností využití systémových prostředků, než Swing. SWT používá k vykreslování nativní systémové knihovny, proto komponenty mají stejný vzhled jako nativní grafické prvky sytému.

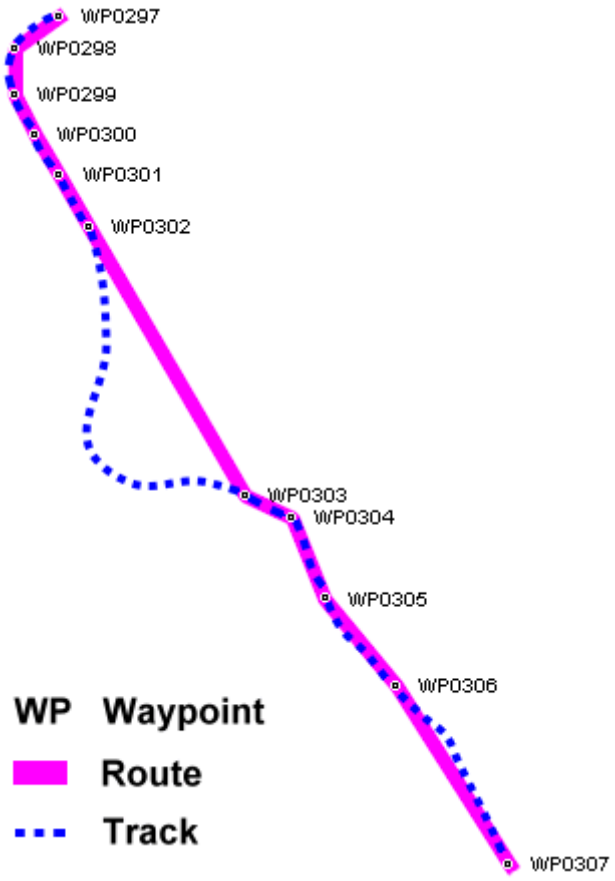
2.6 Ostatní technologie

2.6.1 Formát souboru GPX (GPS eXchange Format)

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

<gpx xmlns="http://www.topografix.com/GPX/1/1"
xmlns:gpxx="http://www.garmin.com/xmlschemas/GpxExtensions/v3"
xmlns:gpxtpx="http://www.garmin.com/xmlschemas/TrackPointExtension/v1"
creator="Oregon 400t" version="1.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.topografix.com/GPX/1/1
http://www.topografix.com/GPX/1/1/gpx.xsd
http://www.garmin.com/xmlschemas/GpxExtensions/v3
http://www.garmin.com/xmlschemas/GpxExtensionsv3.xsd
http://www.garmin.com/xmlschemas/TrackPointExtension/v1
http://www.garmin.com/xmlschemas/TrackPointExtensionv1.xsd">


  <metadata>
    <link href="http://www.garmin.com">
      <text>Garmin International</text>
    </link>
    <time>2009-10-17T22:58:43Z</time>
  </metadata>
  <trk>
    <name>Example GPX Document</name>
    <trkseg>
      <trkpt lat="47.644548" lon="-122.326897">
        <ele>4.46</ele>
        <time>2009-10-17T18:37:26Z</time>
      </trkpt>
      <trkpt lat="47.644548" lon="-122.326897">
        <ele>4.94</ele>
        <time>2009-10-17T18:37:31Z</time>
      </trkpt>
      <trkpt lat="47.644548" lon="-122.326897">
        <ele>6.87</ele>
        <time>2009-10-17T18:37:34Z</time>
      </trkpt>
    </trkseg>
  </trk>
</gpx>
```

Soubor GPX (GPS eXchange Format) je v dnešní době jedním z nejrozšířenějších souborů určených pro uchování informací o GPS poloze a trase. Stejně jako u souboru *osm* se jedná o speciální typ XML souboru. Tento formát podporuje většina moderních GPS navigací a proto se dobře hodí k přenosu naplánovaných tras mezi aplikací operátora systému a navigací řidiče. Uložená data v GPX mohou být tří typů:

- waypoint – element *wpt*, definuje bod, atributy elementu udávají polohu
- route – element *rte*, definuje trasu sekvencí bodů *rtept*
- track – *trk* element, podobný route, ale jedná se o projetou trasu, definovanou sekvencí bodů *trkpt*



Obr. 6: Typy uložených dat v souboru GPX [4]

Příklad souboru GPX uvedený na začátku kapitoly informuje o trase *track*. Jak je z příkladu vidět, soubor se skládá z tří hlavních částí. První částí je definice typu GPX souboru, informace o používaných rozšířeních a dalších důležitých informacích. Druhou částí souboru jsou uložená metadata, informace o tvůrci, doplňující texty a jiné další elementy. Někdy tato metadata, na rozdíl od uveřejněného příkladu, nejsou uzavřena do elementu. Poslední částí a také tou nejdůležitější je struktura samotné cesty skládající se z elementů:

- **Trk** – element zapouzdřuje celou strukturu trasy

- **Name** – tento element informuje o názvu trasy
- **Trkseq** – když se objeví v souboru tento element, znamená to, že zapouzdřuje *trkpt* body do sekvence trasy
- **Trkpt** – stavební kámen trasy. Jedná se o jeden bod trasy. Element obsahuje atributy *lat* a *lon*, popisující souřadnice bodu
    - **Ele** – jedním z elementů bodu *trkpt*. Element zaznamenává elevaci bodu
    - **Time** – druhým z elementů bodu *trkpt*, zaznamenávající čas průchodu

V případě jiného typu dat než je *track*, se struktura datové části GPX souboru změní. Popis jednoho z typů je ovšem pro představu dostačující. GPX soubory mají také možnost definovat mnohá rozšíření. Těchto rozšíření se využívá například ve hře Geochaching pro definování nových elementů potřebných přesně pro tuto hru.

2.6.1 Knihovna pro komunikaci s Webkamerou libvideo0

Jednou z doplňkových služeb, kterou může využívat GPS klient, je například zachycování obrazu z kamery. Zachycování informací z kamery je užitečná věc pro analýzu současné situace vozidla. Nastane-li situace, kdy nejsou k dispozici žádné informace z GPS modulu, třeba v případě velkého zarušení signálu, umožní obraz z kamery vizuálně zanalyzovat vzniklou situaci. Někdy i v případě dopravní nehody je možné data z kamery využít. Jedná se pouze o volitelnou součást systému GPS klientské aplikace.

K tomu, aby bylo možné zachytit obrazová data z kamery, musí být k dispozici potřebná knihova. V programu byla použita knihovna *libvideo0*. Knihovna je součástí projektu *V4L4J* (Video 4 Linux 4 Java). Projekt je hostován na *googlecode*. *V4L4J* zpřístupňuje *V4L* subsystém programovacímu jazyku Java v operačním systému GNU/Linux právě přes zmíněnou knihovnu *libvideo0*.

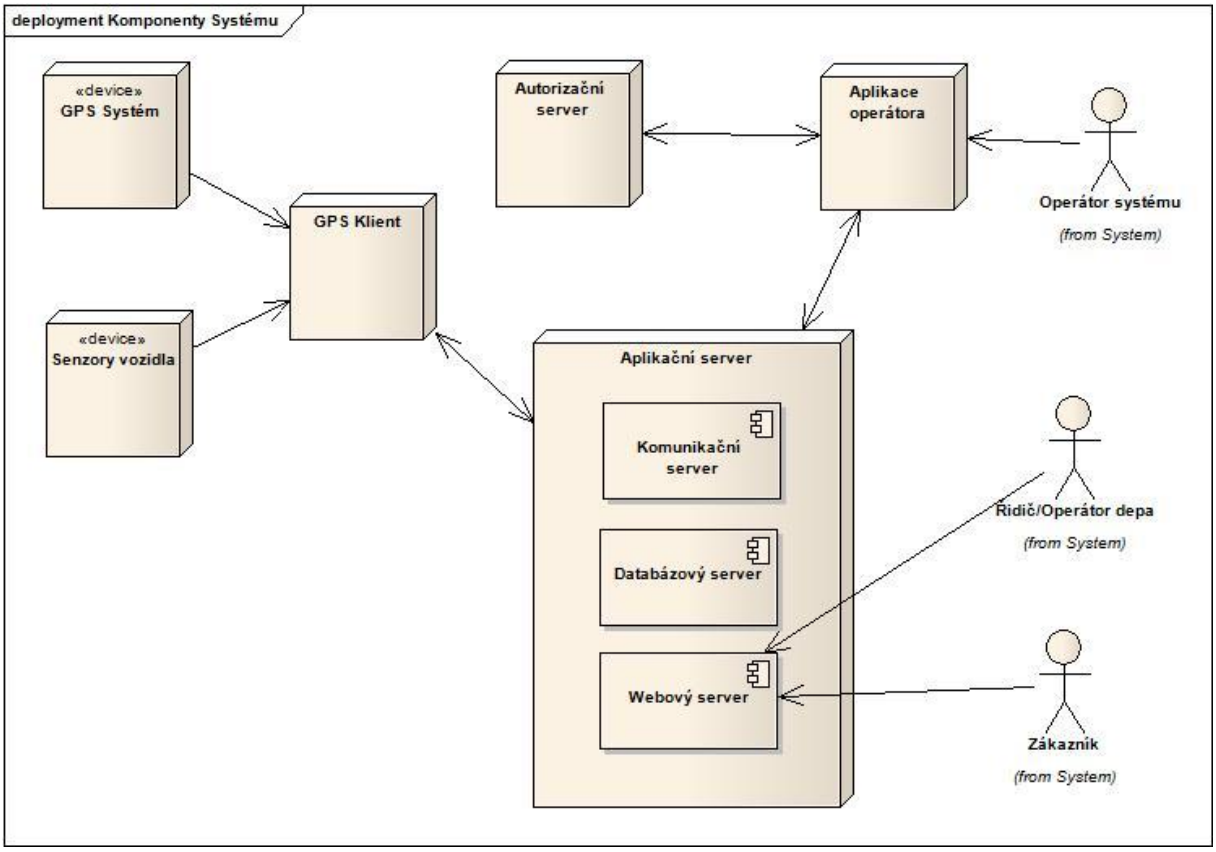
3. Struktura systému sledování

Tato kapitola je věnována části analýzy, návrhu a implementace systému sledování. Je zde vysvětlen základní princip a pochody, které dopomohou k představě o aplikaci. Klíčové části systému jsou popsány v dalších podkapitolách, ve kterých je upřesněna funkce jednotlivých komponentů systému.

První kapitola práce uvádí, že účelem systému je sledování pohybu zásilek a operace s nimi. Tato kapitola se věnuje popisu, řešení části systému.

Systém se skládá z následujících částí:

GPS Klient	GPS klient je aplikace nainstalovaná v operačním systému GNU/Linux uvnitř sledovacího zařízení ve vozidle.
Autorizační server	Autorizační server je služba, která se stará o autorizaci aplikací operátora podle zvolených kritérii.
Aplikační server	Aplikační server poskytuje komunikační rozhraní mezi GPS klientem a aplikací operátora. Aplikační server poskytuje i jiné služby.
Aplikace operátora	Aplikace operátora je aplikací administrační. Disponuje vizualizací pohybu vozidel a umožňuje mnohé nastavení systému.
Webové aplikace	Webové aplikace slouží jako rozhraní pro práci se zásilkou. Existují webová rozhraní pro řidiče, zákazníky a další.



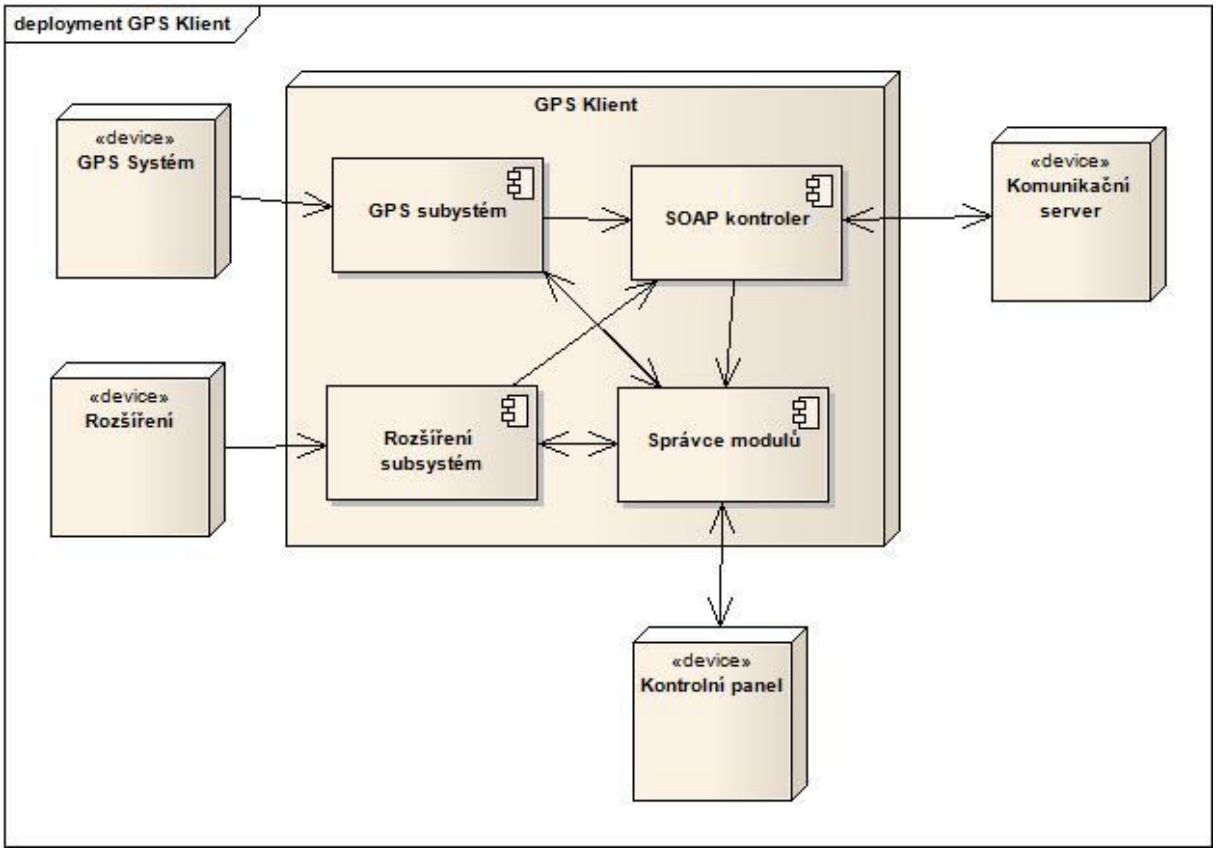
Obr. 7: Grafické znázornění složení systému

Každá součást systému je na obrázku 7 zobrazena jako uzel systému. Webové aplikace jsou na obrázku zastoupeny komponentou *webový server* v uzlu *aplikační server*.

3.1 GPS Klient

GPS klient je aplikace, která je nainstalována v přístroji *GPS tracking* ve vozidle. Aplikace slouží primárně pro komunikaci s GPS modulem. Ihned po získání informací z GPS modulu se tyto odesílají pomocí protokolu SOAP na komunikační server.

Aplikace je stavěna tak, aby umožňovala přidání dalších podpůrných modulů. Mezi podpůrné moduly patří například modul pro zachycovávání obrazu z kamery a jeho následné odesílání na komunikační server. Dalším možným zajímavým rozšířením, které nebylo v rámci práce řešeno, ale pouze navrženo, je rozšíření o komunikaci s různými senzory ve vozidle. Po doplnění by systém byl schopen zaznamenávat například spotřebu vozidla.



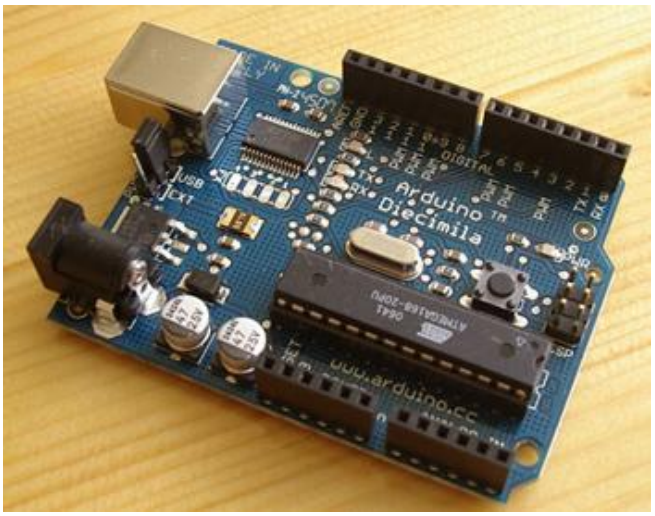
Obr. 8: Schéma částí GPS klienta

Správce modulů	Kontrolní část, která se stará o načtení konfigurace aplikace a správu chodu všech dostupných modulů.
SOAP kontrolér	Slouží ke komunikaci s komunikačním serverem prostřednictvím SOAP protokolu.
GPS subsystém	Subsystém na jedné straně komunikuje s GPS modulem a na druhé s SOAP kontrolérem.
Rozšíření subsystém	Podobný subsystému jako GPS subsystém, pouze komunikuje s jinými zařízeními, například s kamerou.

Je zřejmé, že celý GPS klient je stavěn modulárně, tak aby bylo možné ho dále rozšiřovat v případě budoucí potřeby. Při rozšiřování se musí dbát na vytíženost operačního systému. Běh každého subsystému je zajištěn speciálním vláknem. Při rozšiřování doporučuji nějaké subsystémy spojit, aby GPS klient neměl zbytečně mnoho čekajících vláken. Tyto vlákna čekají proto, aby nedošlo k přetížení mobilního Internetového spojení na komunikační server.

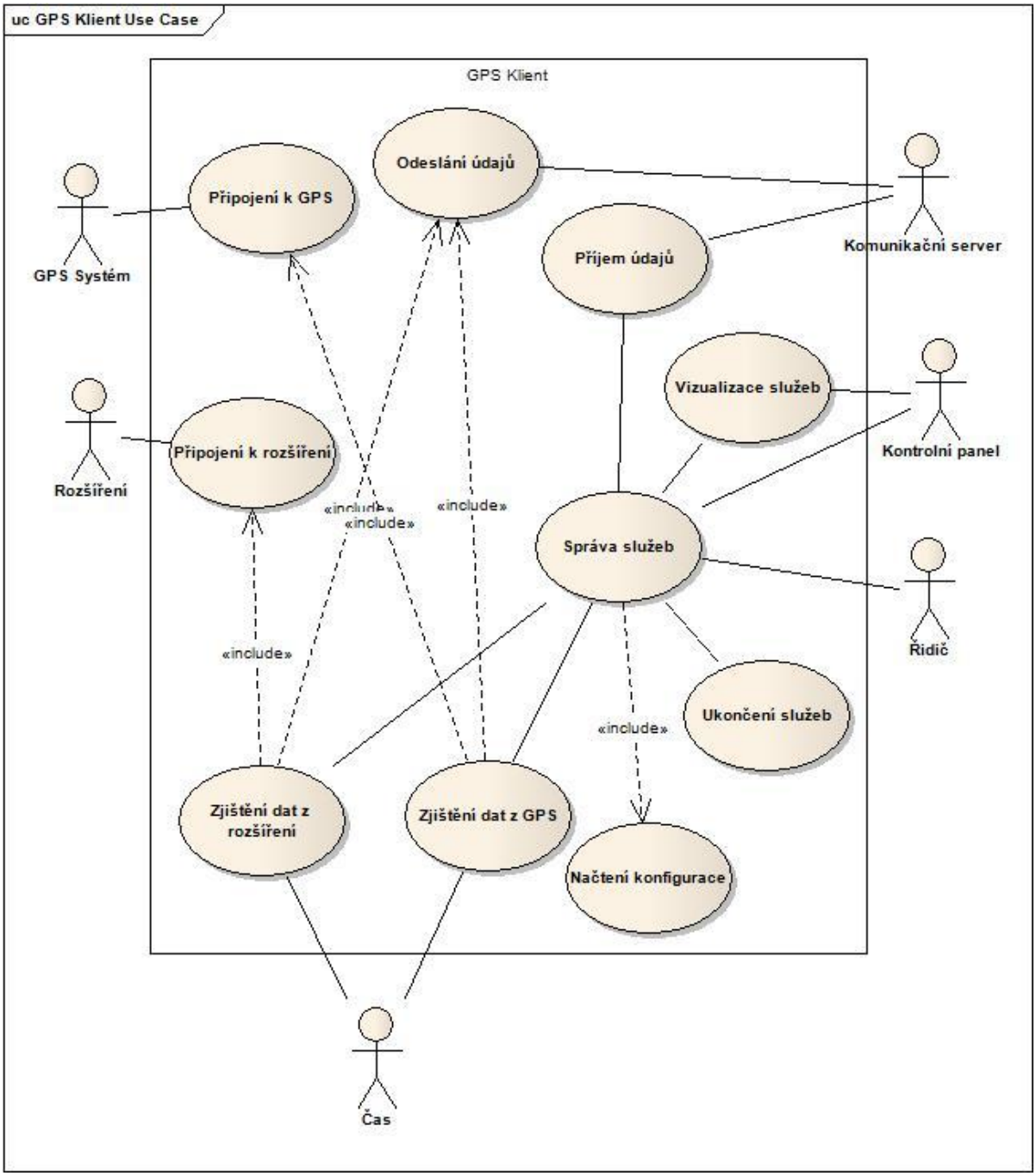
Na schématu popisujícím strukturu GPS klienta se nachází i dvě zajímavá zařízení: *komunikační server* a *kontrolní panel*. Tato zařízení tvoří také aktéry v diagramu případů užití.

Komunikační server je internetový server, který poskytuje SOAP služby pro komunikaci s GPS klienty. Přesný popis je uveden v následující podkapitole *Aplikační server*.



Obr. 9: Programovatelný modul *Arduino* [13]

Druhým aktérem je zmíněný *kontrolní panel*. V případě diplomové práce byl použit programovatelný modul *Arduino* s procesorem Atmel. Tento modul komunikuje s procesorem pomocí sběrnice USB. Na modul je možné připojit digitální/analogové vstupy i výstupy. Byl využit k napojení několika stavových LED, které indikují funkci GPS klienta a přepínače druhu cesty: soukromá/služební. Modul je programovatelný pomocí speciálního programovacího prostředí, které je přímo určeno pro modul *Arduino*.



Obr. 10: Případy užití GPS klienta

Diagram na obrázku 10 znázorňuje, jaké případy použití GPS klienta mohou nastat. Popis diagramu je zbytečný, mluví sám za sebe. Pouze aktéři *řidič* a *čas* si zaslouží bližší vysvětlení.

Aktér *řidič* je v diagramu uveden proto, že právě řidič aktivuje nastartováním motoru GPS klienta. Kromě toho řidič zajišťuje přepínání mezi módy cesty: soukromá/služební. Aktér *čas* slouží pro uspání vláken po předem specifikovaný čas, slouží jako časový spínač akce. Nebýt aktéra čas, došlo by k zahlcení Internetové linky.

Pro představu o komunikaci GPS klienta je v textu jako názorný příklad uvedena funkce subsystému pro zachycování obrazu z kamery. Jedná se o doplňkový subsystém. Pro grafickou interpretaci pomocí diagramu aktivit byl tento druh komunikace vybrán pro jeho vyšší složitost než u ostatních subsystémů.

Složitost komunikace *subsystému kamery* je vyšší než u ostatních subsystémů, protože pomocí protokolu SOAP komunikuje pouze inicializační část algoritmu. Inicializační část algoritmu, zasílání informací protokolem SOAP, je nutná, protože mezi odeslanými informacemi se také nachází informace o velikosti souboru. Ta je potřebná pro určení počtu byte přijatých dat na serveru. Jakmile zašle GPS klient zprávu na komunikační server protokolem SOAP, server se pokusí o start speciálního přenosového serveru. Přenosový server je samostatné odpojené vlákno, na které rodičovské vlákno nečeká. Komunikační server předá GPS klientu číslo portu, na kterém přenosový server naslouchá. Tento port je vybírán podle identifikačního čísla GPS klienta. Jakmile dorazí GPS klientu informace o portu, rozdělí klient JPEG data podle maximální nastavené velikosti zprávy (podle konstantní velikosti bufferu). Rozdělení dat JPEG souboru je potřeba, protože při odeslání všech dat obrázku najednou, a to i při rozlišení 640x480, dojde k přetečení maximální velikosti zprávy. Všechna tato rozdělená data jsou postupně odeslána na server. Jakmile jsou všechna data přijata, ukončí se spojení přenosového serveru a data se uloží na specifické místo jako JPEG soubor. A vlákno přenosového serveru se ukončí.

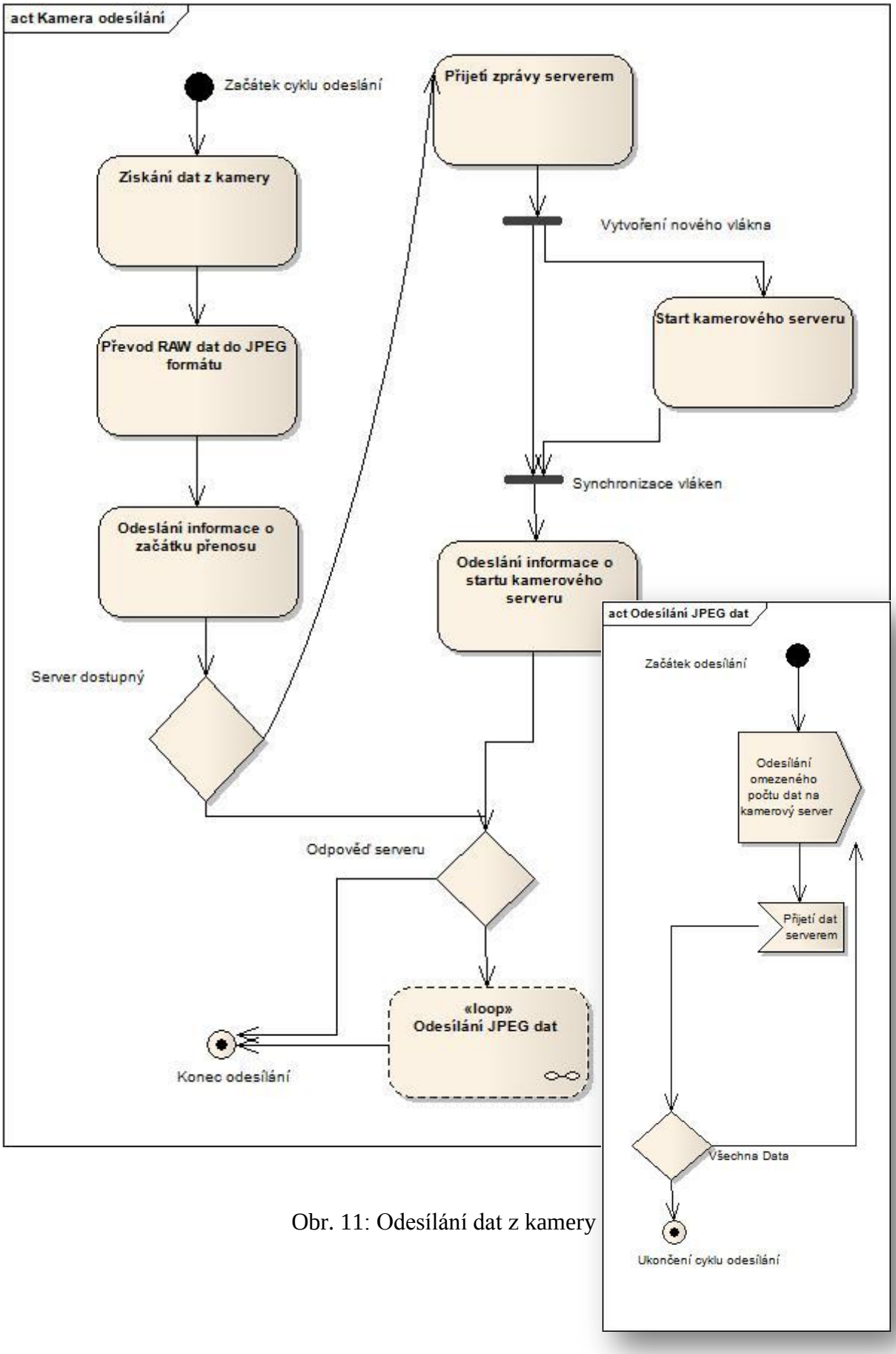
Další možností přenosu obrazových dat z GPS klienta by bylo využití přenosu přílohy zprávy protokolu SOAP. Je to velice efektivní přenos jakéhokoliv souboru bez nutnosti spouštět přenosový server. Háček je v tom, že knihovny použité pro SOAP přenos jak na straně GPS klienta, tak na straně komunikačního serveru, nemají přílohy protokolu SOAP implementované tak, aby spolu mezi odlišnými knihovnami komunikovali.

Součástí aplikace GPS klienta je i startovací skript a konfigurační soubor. Konfigurační soubor má jednoduchou syntaxi:

```
# interval odesílání soapu serveru
ISEND = 10
# četnost zasílání dodatečných gps informací soap serveru
OCCU = 30


# Informace potřebné pro XML přenos
# URL GPS Soap serveru
URL_GPS = http://82.202.96.10/axis2/services/position
URN_GPS = urn:position
# GPS Soap metoda
METHOD_GPS = SetPosition
```

Uvedený konfigurační soubor představuje pouze ukázku. V ukázce je uvedeno pouze nastavení nejdůležitější služby GPS klienta (služba zpracovávající informace o poloze). *Intervaly a četnosti* odesílání jsou globálním nastavením a vyjadřují: po kolikáté zprávě z GPS modulu dojde k odeslání SOAP zpráv na komunikační server. Tři řádky s postfixem GPS jsou konfigurací SOAP klienta pro konkrétní službu.

GPS klient obsahuje několik důležitých přepínačů a parametrů, které musí být pro spuštění aplikace specifikovány. Prvním přepínačem aplikace je cesta ke konfiguračnímu souboru. Dalším z parametrů aplikace, které vyžadují pozornost, je výběr deskriptoru GPS modulu připojeného k USB sběrnici. Tyto deskriptory se nacházejí uvnitř adresáře `/dev`. Skript startující GPS klienta tento parametr vybírá dynamicky, podle systémového protokolu, který obsahuje seznam připojených GPS modulů. Aplikaci GPS klienta je možné spustit jak samostatně, tak s pomocí aplikace GPSD. Stačí pouze specifikovat správný přepínač při startu aplikace. Mezi takové přepínače patří *-c* (aplikace startuje bez GPSD), *-g* (aplikace startuje ve spolupráci s GPSD) a *-gd* (aplikace startuje jak GPSD, tak i sama sebe). Při startu aplikace je potřeba si jeden z přepínačů *-c, -g, -gd* vybrat.



Obr. 11: Odesílání dat z kamery

3.2 Autorizační server

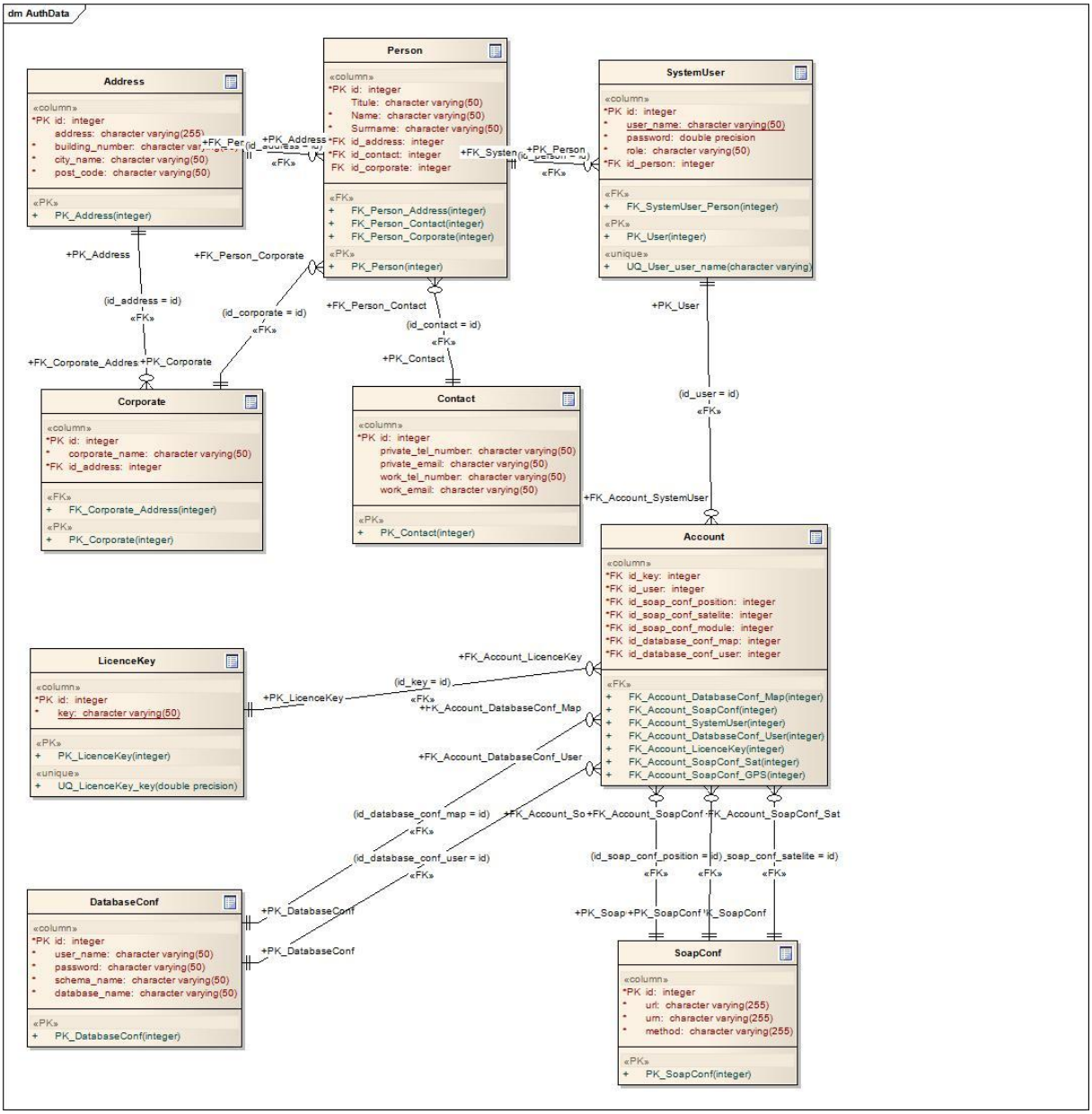
Autorizační server je důležitou součástí systému navrženého v diplomové práci. K tomu, aby bylo možné určovat u přihlašování v aplikaci operátora totožnost konkrétní osoby, slouží právě autorizační server. Systém autorizace v aplikaci operátora je dvouúrovňový.

Autorizace operátora je vyřešena komunikací s databázovým systémem PostgreSQL nainstalovaným na autorizačním serveru. Kvůli bezpečnosti je přenos realizován bezpečnou vrstvou připojení k databázovému systému PostgreSQL SSL. Tato vrstva zaručuje dostatečné zabezpečení. Pro autorizační server je z důvodu bezpečnosti také důležité, aby se jednalo buď o samostatný počítač, nebo autorizační databázový cluster. Autorizační server nesmí být sloučen s aplikačním serverem.

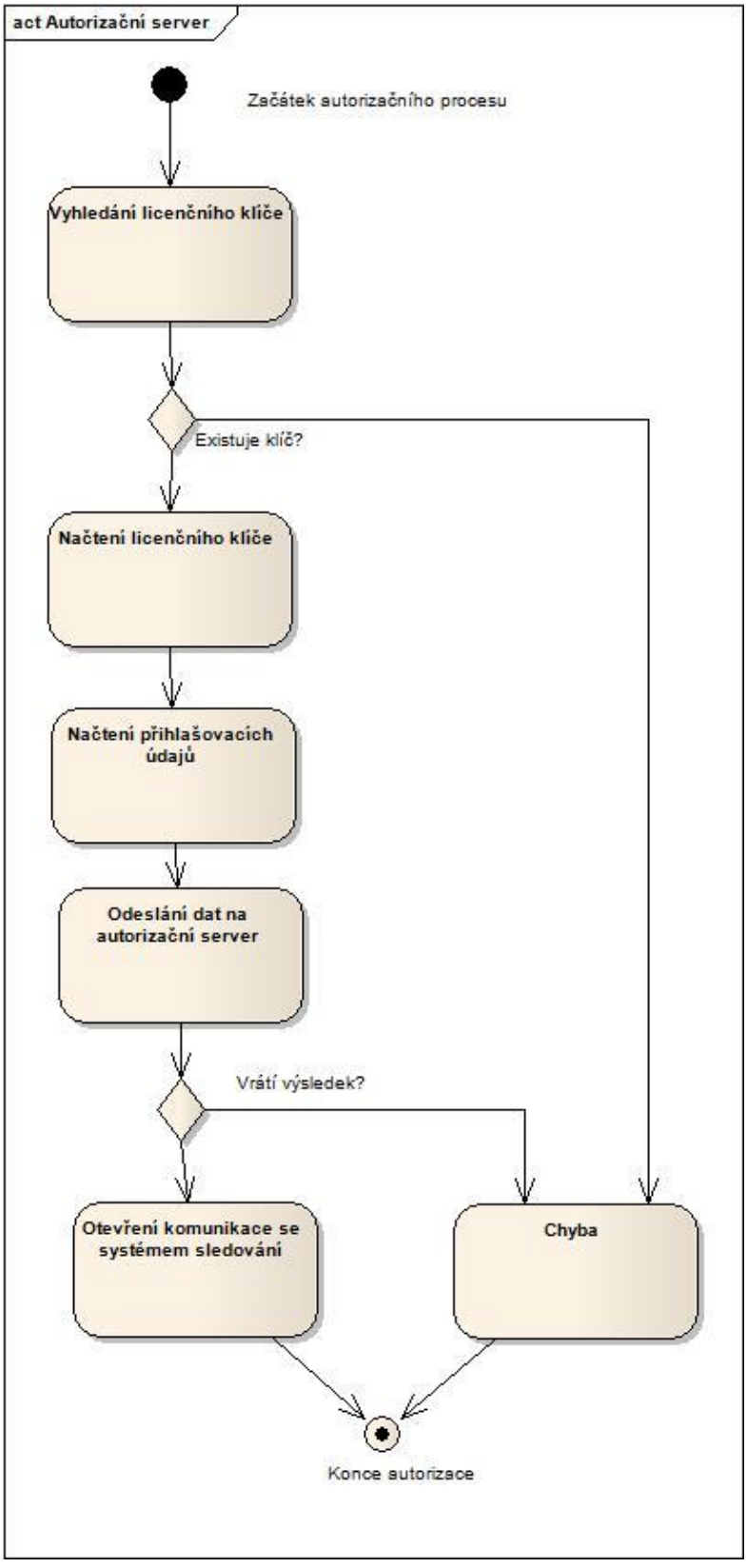
Princip autorizace operátora je následující. První úroveň je založena na licenčním klíči, který musí být v počítači operátora uložen společně s aplikací. Při startu aplikace operátora je tento klíč načten do paměti a v paměti uchován. Po načtení klíče se objeví uvítací okno pro zadání přihlašovacích údajů (ty je možné uložit a při dalším načtení se aplikace už neptá). Po zadání informací potřebných pro přihlášení do systému se odešle dotaz na autorizační server. Pokud autorizační server najde shodu, vrátí kladný výsledek, jinak skončí dotaz s chybou.

Za kladný výsledek dotazu se považují všechny informace, které je potřeba použít pro komunikaci se zbytkem systému sledování. Mezi tyto informace patří: přístupové údaje pro databázi vozidel, zásilek, osob a údaje potřebné pro komunikaci pomocí protokolu SOAP. Tyto informace nejsou obsaženy v aplikaci operátora, ale musí být dodatečně načteny z autorizačního serveru, jinak není možné komunikaci se systémem sledování zahájit.

V případě webových aplikací ale není možné tento autorizační server použít, protože o nasazení webových aplikací se stará přímo vlastník systému. Zákazník nemá možnost nahlížet přímo do kódu, a proto je zabezpečení řešeno jinak.



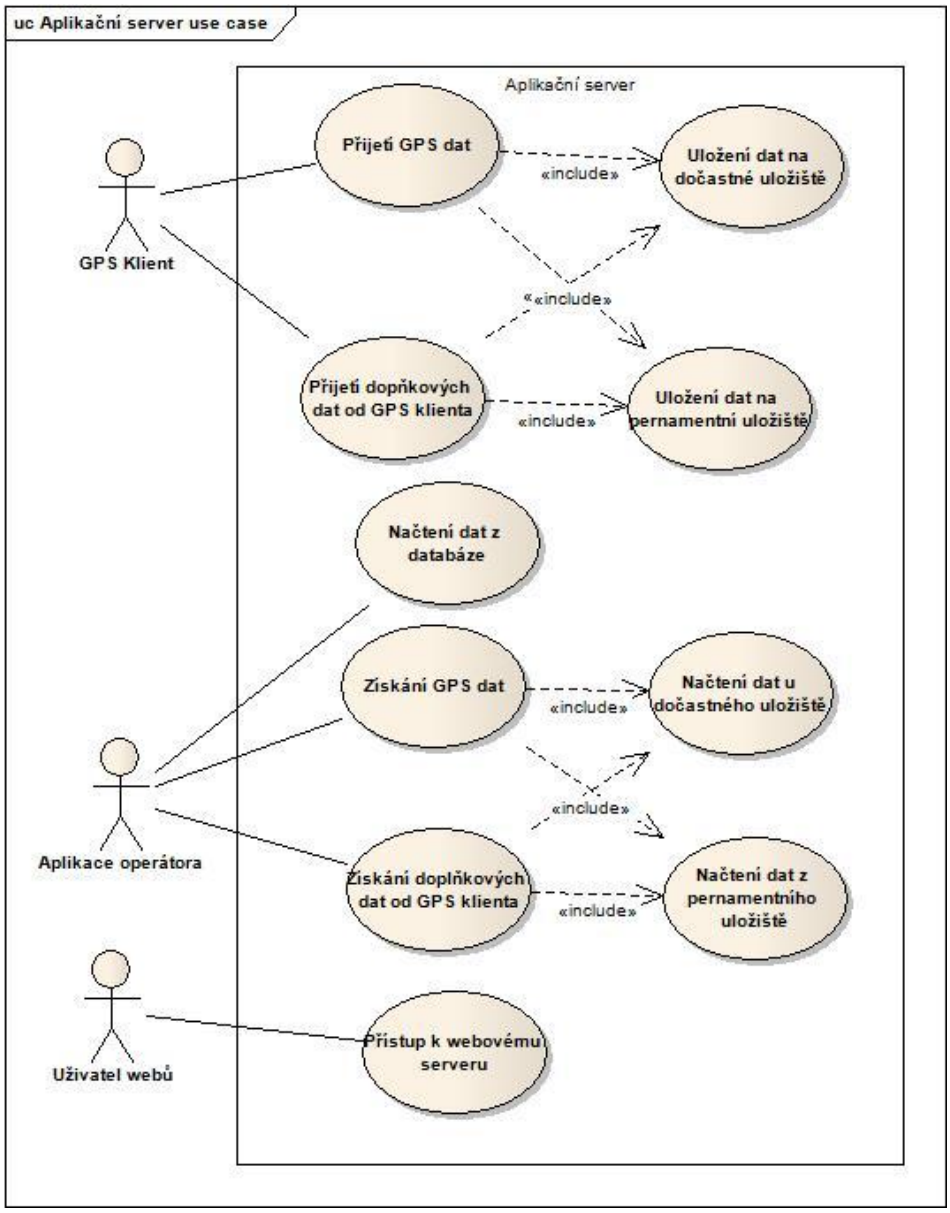
Obr. 12: Schéma databáze autorizačního serveru



Obr. 13: Postup autorizace aplikace operátora

3.3 Aplikační server

Aplikační server je v případě navrženého systému sledování centrálním mozkem systému. Obrázek 7 na začátku kapitoly informuje o tom, jaké komponenty musí aplikační server obsahovat. Tyto komponenty jsou: *Komunikační server* – starající se o přenosy pomocí protokolu SOAP, *Databázový server* – databázový systém PostgreSQL s geografickou nadstavbou PostGIS a konečně *webový server*.



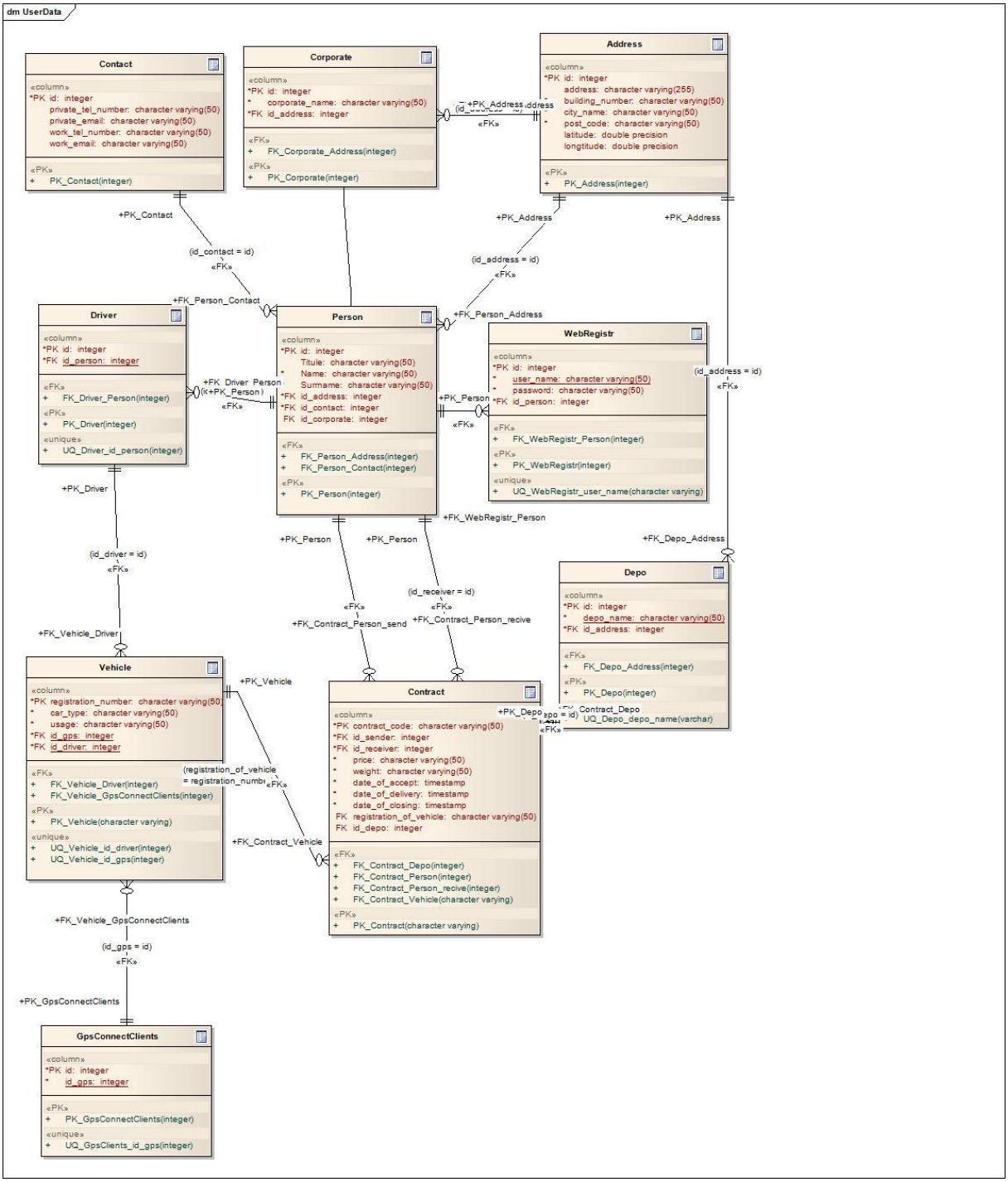
Obr. 14 Případy užití aplikačního serveru

Jako operační systém aplikačního serveru byl vybrán GNU/Linux, distribuce Ubuntu. GNU/Linux poskytuje potřebnou efektivitu pro potřeby systému sledování a odpovídá požadavku na systém. Tímto požadavkem je liberální licence. GNU/Linux je licencován licencí GPL (General Public Licence). Distribuce Ubuntu, ve verzi server, byla vybrána, protože obsahuje potřenou funkcionalitu spojenou se snadným nasazením a administrací.

Detailní informace o výběru komunikačního serveru, nebo databázového systému je uvedena ve druhé části diplomové práce. Principu funkce SOAP komunikace je věnována samotná kapitola, kde je popsána funkce služeb, synchronizace služeb a jiné.

V databázovém systému je vytvořena jedna databáze s několika specifickými databázovými schématy. Databáze v případě současného systému má název *osm*. V této databázi a hlavním schématu se nacházejí geografická data. Nad hlavním schématem databáze *osm* je aplikována nadstavba PostGIS. Proto se do hlavního schématu ukládají i informace o projeté trase.

Databáze *osm* se také obsahuje zákaznická data, tedy informace o lidech, zásilkách a podobně. Pro tato data je potřeba vytvářet speciální schémata. Každé schéma má prefix *UserData* a postfix číselného pořadí zákazníka. Pro testovací účely bylo vytvořeno schéma *UserData001*. Toto rozlišení je důležité pro bezpečnostní oddělení zákaznických dat. Nevýhodou řešení oddělených schémat je fakt, že bude docházet k duplicitě dat.



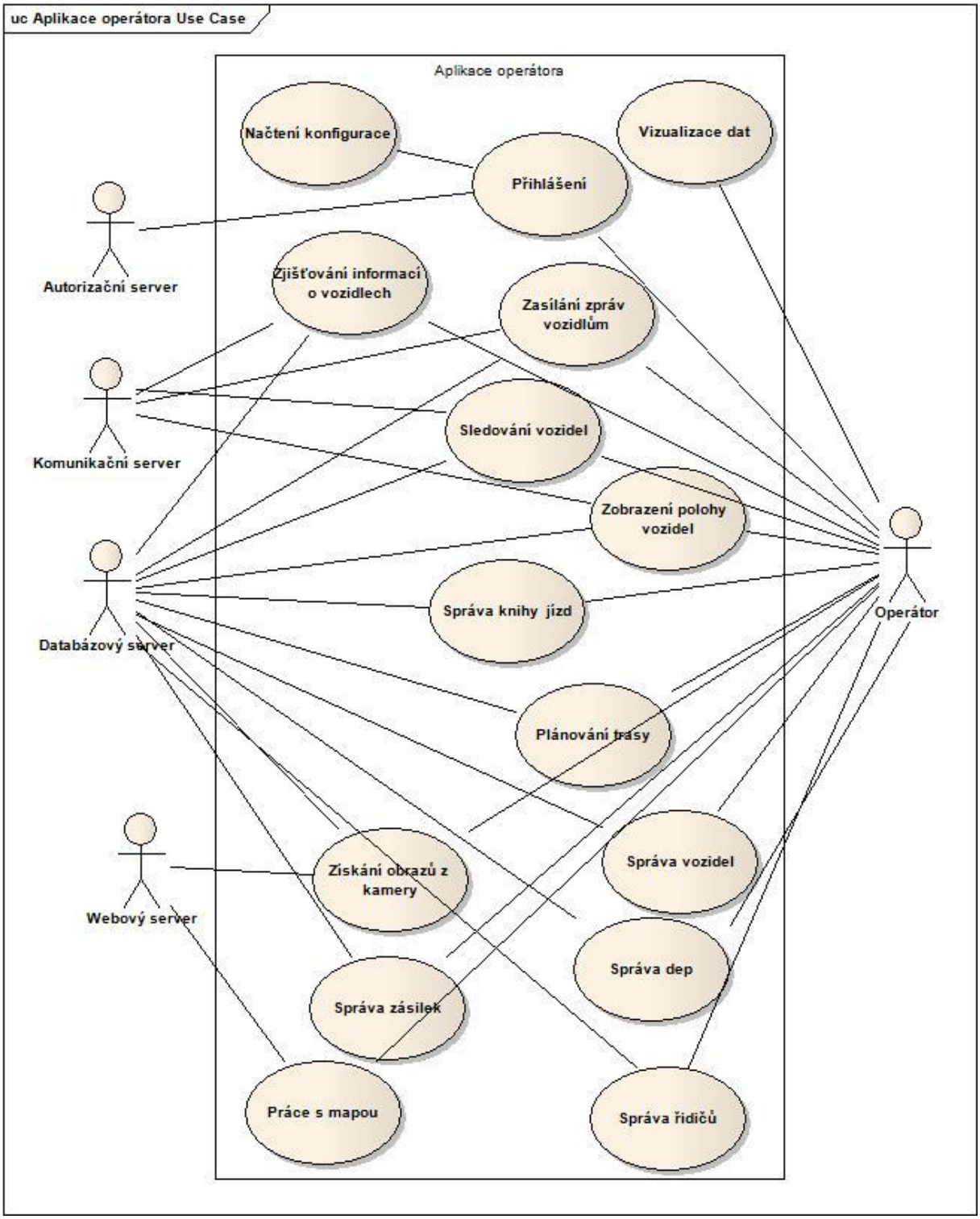
Obr. 15: Diagram znázorňující uživatelské schéma databáze – každý zákazník má vlastní schéma a odlišující se postfixem

3.4 Aplikace operátora

Aplikace operátora je aplikací sloužící ke správě celého systému sledování. Aplikaci doplňují vizualizační prvky. Jedná se o desktopovou aplikaci napsanou v programovacím jazyce Java. Aplikace je tedy proto částečně multiplatformní (částečně proto, jelikož je využita grafická knihovna SWT). Tento software je určen pro operátora dopravy. Operátor dopravy má na starost plánování tras, dohled nad rozvážkou a centrální správu údajů. Například, pokud je někde uvedena špatná adresa příjemce, je umožněno operátorovi tuto adresu opravit.

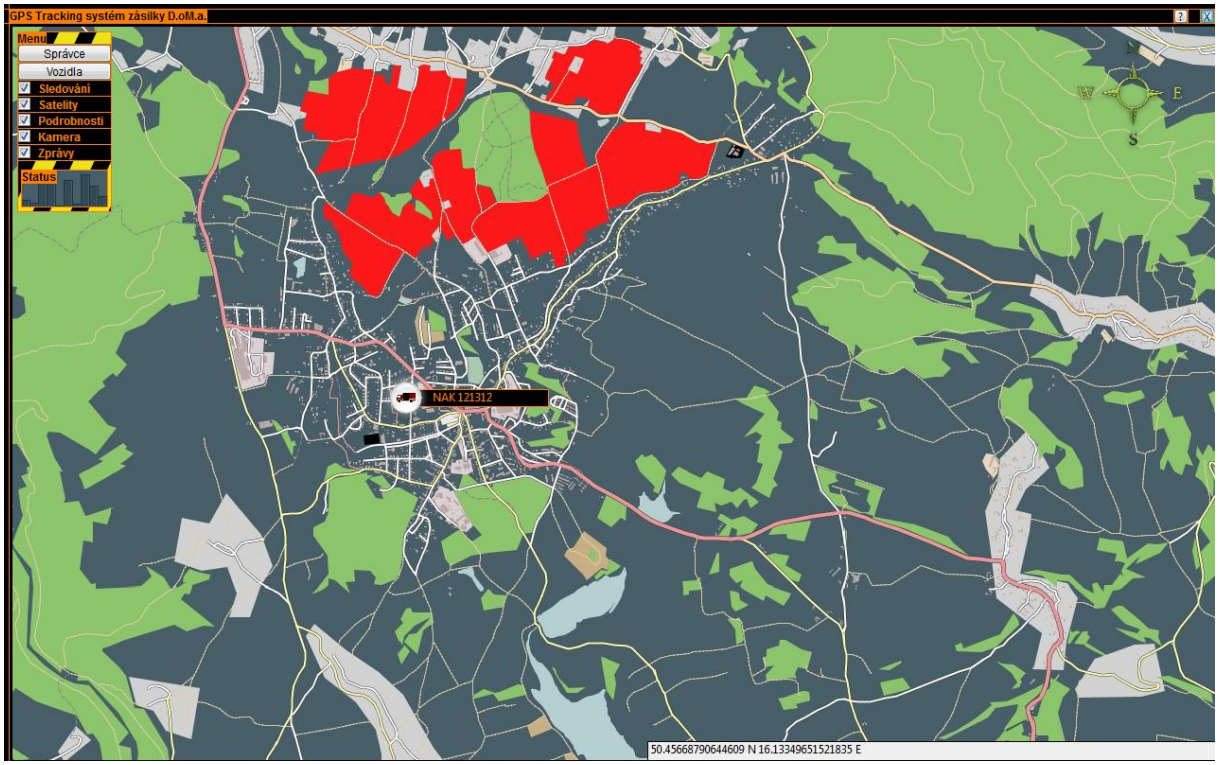
Aplikace je vícevláknová, kde jedno z vláken komunikuje například s komunikačním serverem pomocí protokolu SOAP, jiné vlákno zjišťuje podrobnosti o vozidle, jako třeba obraz z kamery. Další důležité vlákno je vláknem načítání mapy. Algoritmus načítání a generování mapy je popsán v kapitole 5.1. Jsou použity i další vlákna pro práci s databází na pozadí, aby nedocházelo k prodlevám a zásekům v grafickém prostředí.

Z diagramu případů užití (obrázek 16) je vidět, že aplikace disponuje pěticí aktérů. Hlavním aktérem je právě *operátor systému*. Jeho role byla popsána na začátku kapitoly. Dalšími aktéry jsou servery obstarávající data. Prvním ze serverů je *autorizační server*, který pomocí správných přihlašovacích údajů zašle aplikaci operátora konfigurační údaje. Po načtení konfiguračních údajů začnou být aktivní zbylí tři aktéři. Tito tři aktéři spolu tvoří aplikační server. *Databázový server* umožní aplikaci načíst údaje o zásilce, adresátech, vozidlech a depech. *Komunikační server* komunikující s aplikací operátora pomocí protokolu SOAP zasílá informace o současné poloze vozidel. Tímto komunikačním kanálem přichází informace i o satelitech. Poslední z trojice tvořící aplikační server je *webový server*. Pomocí tohoto aktéra získá aplikace operátora všechna možná obrazová data. Mezi nimi jsou například: mapa či záznamy z kamery.



Obr. 16: Nejdůležitější případy užití aplikace operátora

Mezi případy užití aplikace patří například: *přihlášení do systému, správa informací o vozidlech, řidičích, depech či zásilkách*. Dále se dá mezi případy užití zařadit zobrazení *polohy vozidel, sledování vozidel a zasílání zpráv vozidlům*. Důležitou vlastností je i získávání podrobných informací o vozidle (satelity, čidla). Aplikace je schopná získávat obraz z kamery vozidla. Pro sledovací část jsou důležité případy užití jako vizualizace, nebo práce s mapou. A pro zvýšení efektivity provozu jsou zde i případy užití jako je správa knihy jízd, nebo dokonce plánování trasy. Algoritmus na plánování trasy je popsán v kapitole 6.



Obr. 17: Aplikace operátora

3.5 Webové aplikace

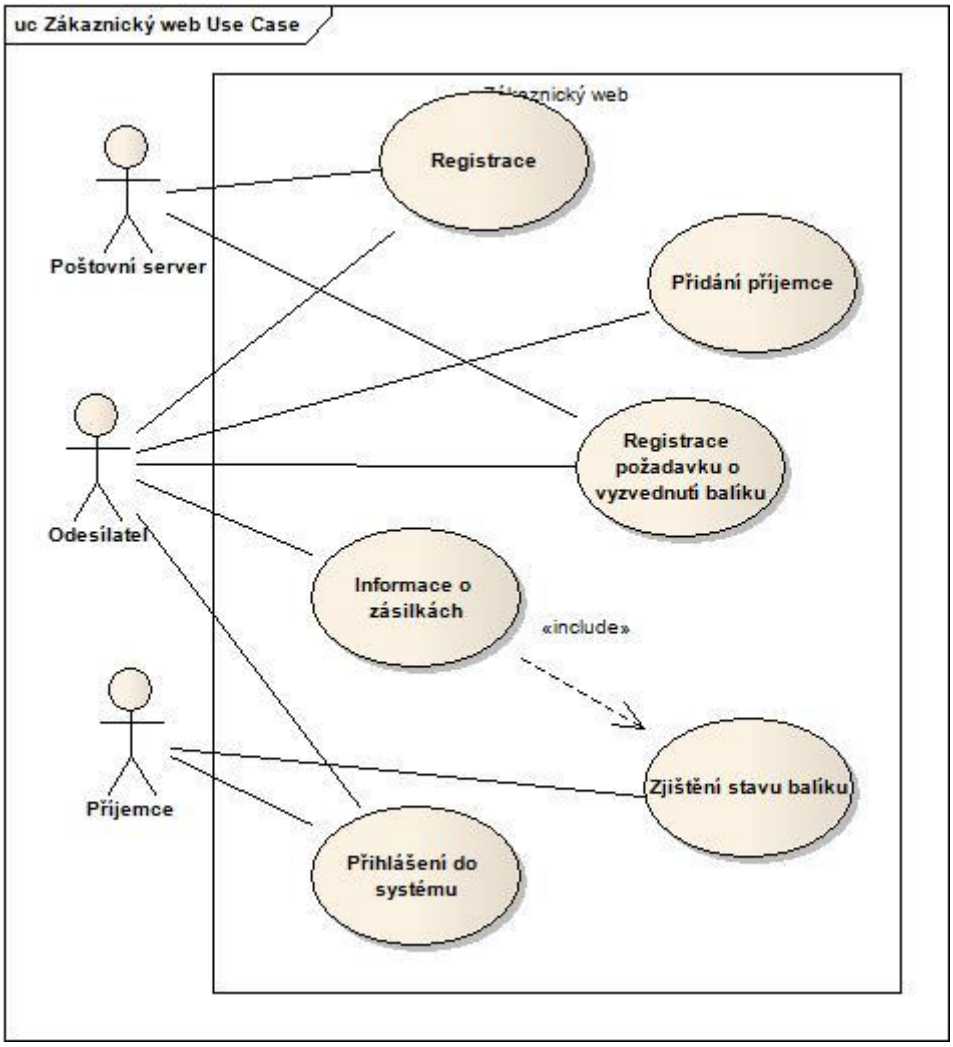
Webové aplikace jsou důležitou součástí systému. Tyto aplikace komunikují jak s řidiči, operátory depa, nebo třeba i s adresáty balíku. Jedná se o dvě naprosto odlišné aplikace. Jedna aplikace slouží k veřejné prezentaci systému pro adresáty a odesílatele balíku a druhá aplikace je určena pro řidiče, nebo operátory depa. První aplikace je normálně přístupná v síti Internet po ověření oprávnění. Druhá aplikace je kvůli zabezpečení přístupu dostupná pouze v interní síti po ověření identifikačního čísla osoby, který chce k systému přistupovat.

Obrázky webových aplikací jsou přiloženy v příloze 10.5.

3.5.1 Webová aplikace pro zákazníky

Aplikace určená pro komunikaci se zákazníky zajišťuje registraci zákazníků, nových zakázek do systému, sledování pohybu balíků, přidávání informací o příjemcích a jiné.

Registrace zákazníka slouží pro uživatele webu, kteří chtějí odesílat vlastní balíky. Registrace zákazníka je placená. Uživatel registrující se do systému dostane po ukončení registrace e-mail, který obsahuje fakturační údaje pro registraci. Zaplacením registrace dostane zákazník specifický počet balíkových kódů (kód balíku slouží pro usnadnění operace s balíkem). Jakmile bude chtít zákazník odeslat balík, vybere jeden z jemu přidělených kódů balíku. Po odeslání objednávky musí zákazník vytisknout dokument žádanky obsahující kód balíku. Společně s odesláním registrace balíku do systému se odešle e-mail příjemci (pokud ho odesílatel uvede) s informací o odeslání balíku na jeho jméno a webovou stránkou obsahující detailní informace o termínu doručení. Jakmile zákazník překročí počet jemu přidělených kódu, jsou mu další kódy fakturovány paušálně jednou za měsíc.

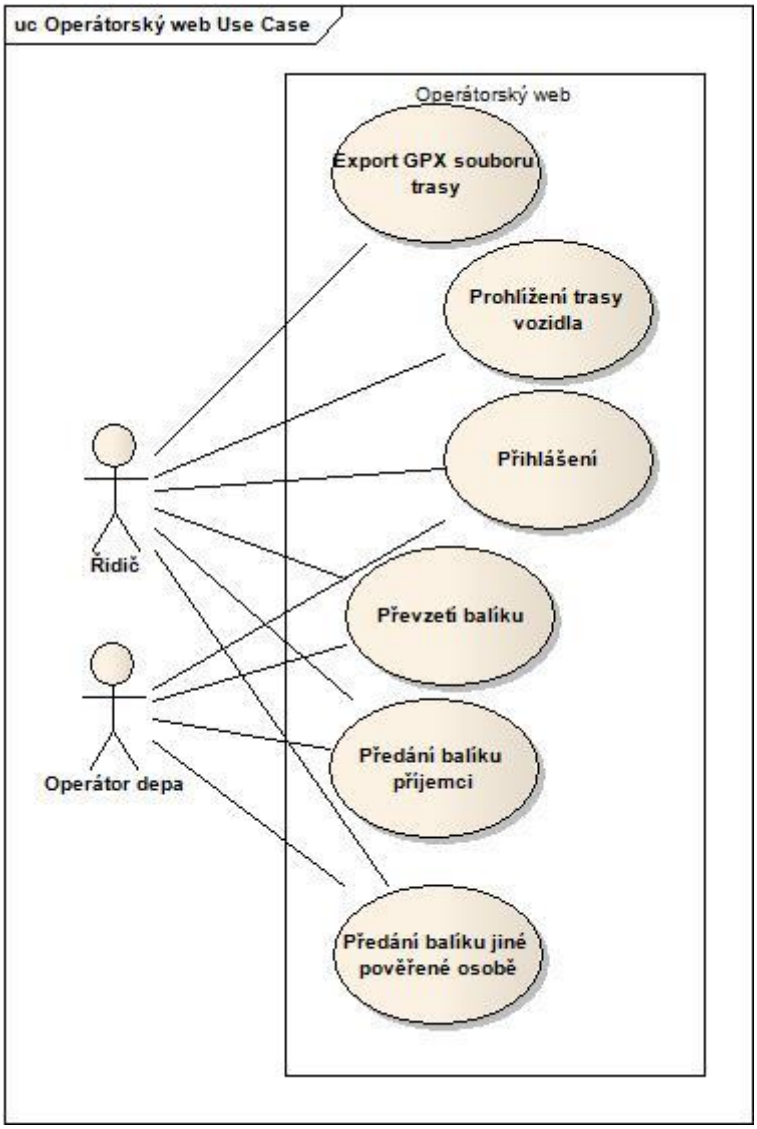


Obr. 18: Případy užití zákaznického webu

3.5.2 Webová aplikace pro řidiče a operátory

Web určený pro řidiče a operátory depa je zajímavější. Nejprve je nutné uvést, proč byla zvolena pro tento účel forma webové aplikace. Systém předpokládá, že řidič a operátor depa disponují speciální čtečkou čárových kódů. Tyto čtečky je možné připojit na Internet pomocí modulů WiFi nebo GSM. Čtečky ovšem mohou být postaveny na různých platformách. Proto je nejlepší použít řešení, které problém multiplatformovosti eliminuje. A tím je webová aplikace.

Webová aplikace pro přenosná zařízení musí být upravena tak, aby měla malé rozlišení, byla rychlá a obsahovala co nejmenší počet grafických vymožeností. Aplikace proto obsahuje přehledná ovládací rozhraní skládající, se pouze z formulářů pro zadávání a výpisů.



Obr. 19: Případy užití webu řidiče/operátora depa

Webová aplikace pro řidiče a operátory depa obsahuje podle obrázku 19 pouze dva aktéry, kteří mají podobné případy užití aplikace. Aplikace sama rozlišuje, o jakého aktéra se jedná, protože přihlašování do aplikace probíhá přes identifikační číslo zaměstnance. Každý zaměstnanec má na své kartě zaměstnance uvedeno právě toto identifikační číslo. Webová aplikace musí pro oba aktéry umět vyřešit převzetí balíku od odesílatele, řidiče, nebo opět operátora depa. Odesílatel připojí při odesílání balíku ještě podpis. Obrázky s podpisy se ukládají pod kódem balíku na specifické místo v aplikačním serveru.

Aplikace musí umět také v případě potřeby předat balík odpovědné osobě, co je v podstatě to samé, jako převzetí balíku od odpovědné osoby. Pod pojmem odpovědná osoba se rozumí například operátor depa. Důležitější předání, které musí aplikace zvládat, je předání příjemci. Příjemce, stejně jako odesílatel, musí doložit převzetí svým podpisem. Přebírat a předávat balíky mohou oba aktéři, jak řidič, tak operátor depa, protože pokud řidič nezastihne adresáta doma, zanechá balík v nejbližším depu. V tomto depu si pak následně může adresát zásilku vyzvednout. Proto musí i operátor depa mít možnost předat balík adresátovi. Podobná analogie platí i o odesílání balíku přímo z depa.

Řidič má ve webové aplikaci oproti operátorovi depa navíc další funkcionality. Tyto funkcionality se týkají trasy, kterou má projet. První funkcí je prohlídka seznamu bodů trasy. Pod seznamem bodů trasy se rozumí seznam zásilek, které má rozvést, včetně potřebné adresy a jména adresáta. Tento seznam je seřazen stejně jako body řidičovi trasy. Seznam se dynamicky upravuje podle předaných balíků. Druhou funkcionalitou, kterou má řidič k dispozici navíc, je export trasy, kterou má vykonat, do souboru GPX. Tento soubor může řidič uložit na paměťový modul a naimportovat do GPS navigace. I v tomto souboru jsou detailní informace o adresátech a balících. Tyto dodatečné funkcionality mají optimalizovat trasu řidiče a snížit náklady na provoz vozidla.

4. SOAP Komunikace

SOAP komunikace hraje v programu důležitou roli. Proto je vhodné vysvětlit její funkci.

Tuto komunikaci používají všechny části systému. GPS klient umístěn ve vozidle komunikuje primárně pomocí SOAP protokolu (pouze data, jako například obraz, se odesílají binárně). V případě GPS klienta jsou implementovány 3 SOAP metody. První a nejdůležitější metodou, je metoda na odesílání informací o poloze, zároveň s ní se zasílají také informace o rychlosti, nadmořské výšce, času a fixování satelitů. Nadmořská výška je aproximací vypočítaná z GPS signálu. Jde pouze o orientační informaci. Důležitými prvky zprávy jsou čas a rychlost. Pomocí těchto informací se následně sestavuje kniha jízd.

Služby jsou na serveru implementovány tři: *position*, *camera*, satelite. Služba *position* nastavuje a vrací informace o pozici GPS klienta. Služba *camera* slouží ke spouštění přenosového serveru pro přenos obrazových dat z kamery. A konečně služba *satelite* nastavuje a vrací informace o viditelných satelitech GPS klienta. Metody služeb jsou pak následně rozlišeny prefixem *Get/Set* podle toho, o jaký typ metody se jedná. Zda vrací, nebo nastavuje informace.

Výpis služeb jasně informuje o tom, jaké SOAP služby testovací aplikační server nabízí. Na testovací adrese komunikačního serveru: http://www.server.cz/axis2/services/ je možné najít tento výpis služeb:

Deployed Services

position

Service which sets and gets information about GPS position.

Available Operations

- SetPosition
- GetPosition

camera

Service which sets and gets information about GPS satelites.

Available Operations

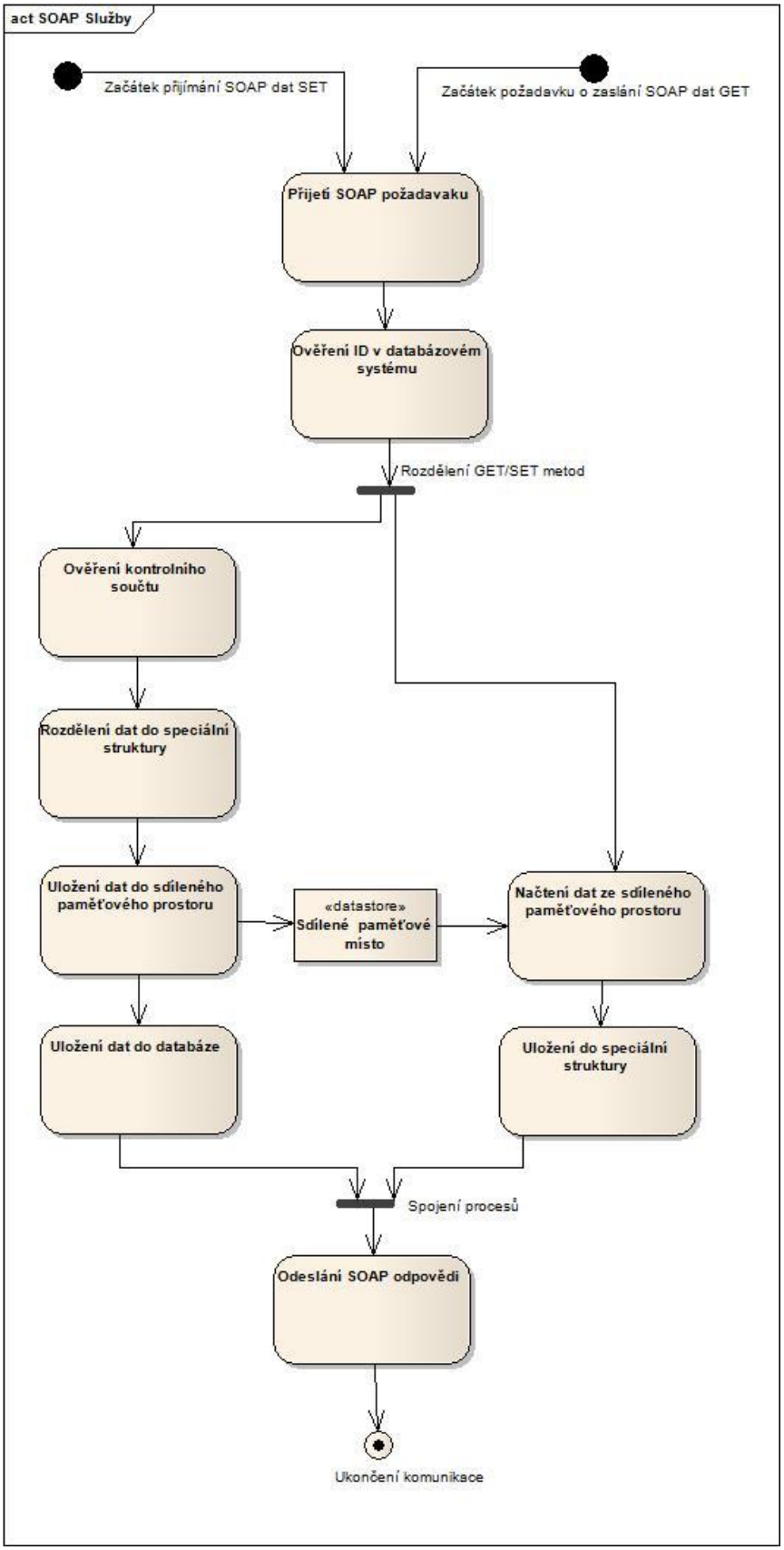
- SetCamera

satelite

Service which sets and gets information about GPS satelites.

Available Operations

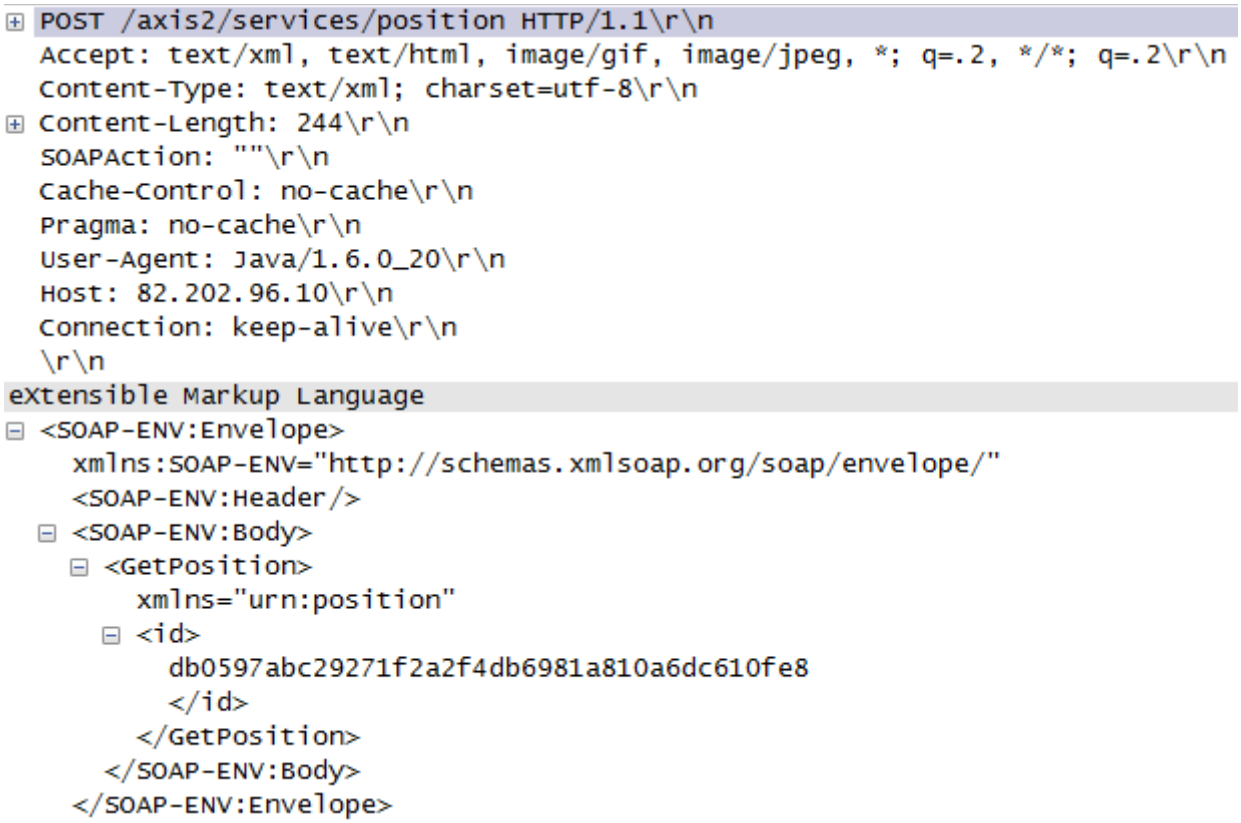
- SetSatelite
- GetSatelite



Obr. 20: Ukázka funkce a komunikace SOAP služeb systému sledování

4.1 Bezpečnost komunikace

SOAP komunikace probíhá pomocí XML souborů v čistém textu, což představuje velký bezpečnostní problém.



Obr. 21: Odchycení komunikace SOAP protokolu aplikací *Wireshark*

Proto W3C konsorcium navrhlo standart XML Security. Tento standart je určen k šifrování a podepisování těla XML zprávy. Je použit i pro SOAP komunikaci. Někdy ovšem bývá dost špatně implementován, ve většině současných SOAP knihovnách není XML Security ani zahrnut.

Otázkou je, jak je možné komunikaci zabezpečit před případnými útočníky. Jedna možnost je šifrování celého přenosu. K tomuto účelu se dají použít současné point-to-point tunelové/šifrovací protokoly, používané ve VPN sítích.

V dnešní době různé firmy zajišťující tracking systémy vozidel používají pro přenos informací o vozidlech smtp protokol. Dle mého názoru to je špatný přístup, protože se nejedná o realtime bezpečnou komunikaci. Když se použije SOAP přenos údajů navržený v této aplikaci a spojí se dohromady s VPN tunelem, vznikne dostatečný stupeň zabezpečení.

Mezi VPN algoritmy jsou k dispozici OpenVPN, PPTP, VPNC a jiné. Z těchto algoritmů je nejzajímavější PPTP (Point-to-Point Tunneling Protocol). Má sic nejmenší šifrovací klíč (128 bit), ale pro účel aplikace je dostačující. PPTP je zajímavý i pro to že instalace v GNU/Linux je snadná. Většina routerů tímto typem VPN disponuje už v základu a hlavně se jedná o implicitní VPN systém na operačním systému Windows. Není potřeba nic instalovat. Microsoft Windows je stále nejrozšířenější desktopový operační systém a proto je dobré řídit se trhem.

Mezi nejzajímavější možnosti zabezpečení SOAP komunikace však patří SSL vrstva webového serveru. V distribucích GNU/Linux jako je Ubuntu je většinou vše potřebné připraveno, modul webového serveru stačí pouze aktivovat. Rovněž knihovny SOAP protokolu mají často podporu SSL vrstvy už integrovanou. V praxi se tedy nechá tato technologie bez problémů použít.

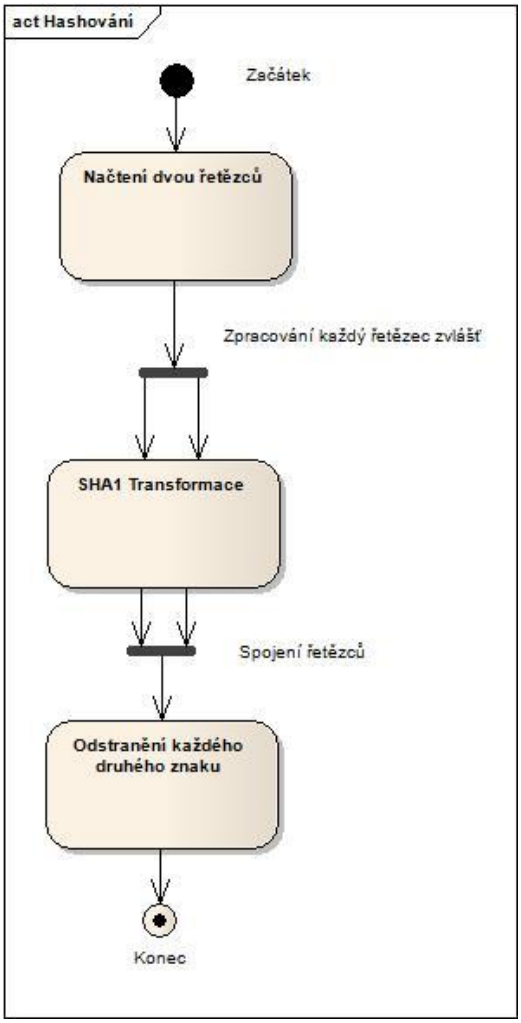
Stejná technologie šifrování se používá u spojení databázového systému PostrgeSQL. Při nasazení se změní pouze URL požadavku na adresu, na které komunikuje SSL vrstva.

Protože zabezpečení pomocí SSL vrstvy je obvykle dostačující a je zároveň jednoduché pro nasazení u zákazníka, byl tento způsob zabezpečení zvolen jako optimální pro datovou komunikaci systému.

4.2 Použití jmen a následné šifrování

Pokud by byla SSL vrstva prolomena, jsou SOAP zprávy zabezpečeny ještě mechanismem *hashování*. Zpráva obsahuje dva elementy, které jsou transformovány na *hash* pomocí speciálního algoritmu, který vychází z SHA1 jednosměrného *hashovacího algoritmu*. Je řeč o elementech *id* a *checksum*.

Id je důležitý element, který slouží jako vstupní brána ke všem serverovým službám. Id spočívá v sestavení řetězce skládajícího se z dvakrát po sobě jdoucí řetězců upravených pomocí SHA1 transformace *hostname* GPS klienta. Výsledný řetězec je následně ještě jednou transformován. Transformace spočívá v jednoduchém vynechání každého druhého znaku 80 byte dlouhého řetězce a jeho transformaci na 40 byte řetězec. Tyto operace jsou zvoleny proto, aby prolomení brutální sílou bylo složitější, než u běžného SHA1 algoritmu.



Obr. 22: Hashování

Element *checksum* je element, použitý jako kontrolní součet zprávy. Veškeré transformace zprávy jsou stejné jako u elementu id. Liší se jen ve druhém řetězci zprávy, kde se volí specifické variabilní číslo, například počet satelitů v danou chvíli měnící se s každou zprávou. Kdyby útočník prolomil SSL a dokázal dešifrovat *id*, element *checksum* bude sloužit jako poslední obranná linie.

4.3 Komunikace mezi serverovými službami

Procesy služby SOAP serveru jsou vytvářeny vlákny Apache web serveru. Tato vlákna jsou vytvořena, jakmile přijde na webové služby požadavek. Požadavek se vyřídí, odešle se odpověď a proces se ukončí. Toto chování má výhodu v tom, že vyřízení je přímočaré, rychlé a snadné. Problém vzniká v případě, kdy data tohoto procesu je potřeba synchronizovat s jiným procesem.

V případě potřebných běžících služeb pro tuto aplikaci vzniká komplikace. Jakmile přijde požadavek na SOAP server s daty od GPS klienta, jsou to často údaje, které je většinou naprosto zbytečné uchovávat pro další zpracování. Mezi taková data patří třeba informace o satelitech. Může jít i o situace, kdy není vhodné zbytečně zatěžovat databázový server. Třeba u dat obsahujících informaci o poloze by se muselo provádět při požadavku o zaslání dat o několik dotazů do databázového serveru navíc. Tím vzniká větší vytížení serveru a ke všemu naprosto zbytečně.

Komplikace tohoto programu, tedy komunikace mezi serverovými službami, je vyřešena pomocí mapování speciálních struktur do sdílené paměti. Při přijetí SOAP požadavků jsou údaje převedeny do speciální struktury a následně namapované na speciální místo v paměti. Klíč mapované paměti je volen v závislosti na ID GPS klienta. Ihned jak dorazí požadavek na vyzvednutí dat uložených v namapované paměti se stejným způsobem vypočte stejný klíč k mapované paměti a uložená struktura se načte z paměti, zpracuje a odešle v odpovědi.

Řešení komunikace přes mapování paměti snižuje režii na úplné minimum. Není potřeba zatěžovat databázový server. A není potřeba ani ukládat data na pernamentní médium, jako je pevný disk. Ukládání jak na pevný disk, tak do databázového serveru, je náročná operace na rozdíl od ukládání pouze do dočasné paměti.

5. Zobrazení kartografických dat

Mezi nejzajímavější část diplomové práce patří způsob zobrazování kartografických dat v aplikaci. Možných způsobů bylo otestováno v aplikaci několik.

Jako první bylo vyzkoušeno načítání geografických dat přímo v aplikaci operátora z databázového systému PostgreSQL, s nadstavbou PostGIS, stejným vláknem, kterým bylo ovládáno celé grafické rozhraní. Tento způsob se ukázal být velmi neefektivním. Následně bylo testováno načítání map na pozadí pomocí speciálního vlákna. Uvedená metoda je již o mnoho efektivnější, ale v případě podrobné mapy bylo potřeba stále načíst velké množství dat z databázového systému. Společnou nevýhodou obou zkoušených metod je fakt, že při překreslování polohy vozidla se rovněž musí znovu překreslovat veškerá vektorová geografická data. To představuje velkou režii pro počítač.

Pro vyšší efektivnost předchozích algoritmů byla do programu zahrnuta pomocná funkce. Ta funguje tak, že po načtení geografických dat z databázového systému aplikace operátora vykreslí do paměti bitmapu. Tu pak aplikace při každém pohybu vozidla vykreslí jako mapový podklad na zobrazovací zařízení. Ušetří se tak režie, protože nemusí být vykreslována opakovaně vektorová data. A také se omezí počet databázových dotazů na nutné minimum. Geografická data jsou načítána pouze tehdy když je posouván mapový podklad.

Uvedený způsob vytváření bitmapy snížil podstatně vytížení počítače. Problém ovšem přetrvává, jakmile je v aplikaci aktivováno sledování vozidla. Při něm je vozidlo umístěno do středu mapy a pohybuje se mapa. A pokud vozidlo změní vůči mapě pouze jen nepatrně svoji polohu, je nutné načíst nová geografická data. Časté načítání při sledování vozidla lze celkem snadno vyřešit tak, že se vytvoří bitmapa cca o velikosti 400 pixelů na každé ose větší než je zobrazovací zařízení. Vykreslením bitmapy s rezervou 400 pixelů na každé ose se vytvoří teoretický okraj o velikosti 200 pixelů na každé hraně. Teoretický okraj umožní posouvání mapového podkladu o 100 pixelů bez nutnosti generování nové bitmapy. Pokud bude mapový podklad posunut o víc jak 100 pixelů, aktivuje se na pozadí aplikace proces generování nové bitmapy.

Tak lze celkem snadno vyřešit neustálé načítání geografických dat v případě malého posunu mapy a zamezit výskyt černých míst na krajích. I přes předchozí doplňky se ale jedná o neefektivní vykreslovací metodu. Z tohoto důvodu byl pro aplikaci operátora vybrán naprosto odlišný algoritmus. Princip použitého algoritmu je popsán v následující podkapitole.

5.1 Efektivní vykreslování

Jak bylo uvedeno v předcházející části, cesta přímého načítání geografických dat z databázového systému do aplikace operátora k efektivitě nevede. Bylo nutné najít algoritmus, který je mnohem efektivnější a lépe použitelný.

Při testech se ukázalo, že nejlepší cestou je použití algoritmu, který je využíván poměrně často u mapových webových serverů. Jedná se o skládání mapového podkladu pomocí malých čtvercových obrázků mapy. Řešení tohoto typu je ozkoušené a oproti předešlému algoritmu sestavování mapového podkladu ze surových vektorových dat i mnohem rychlejší. V případě diplomové práce jde o skládání mapového podkladu z dílku velkých 200x200 pixelů. Takový obrázek je natolik malý, že přenos většího počtu lze zvládnout i při relativně pomalém Internetovém připojení.

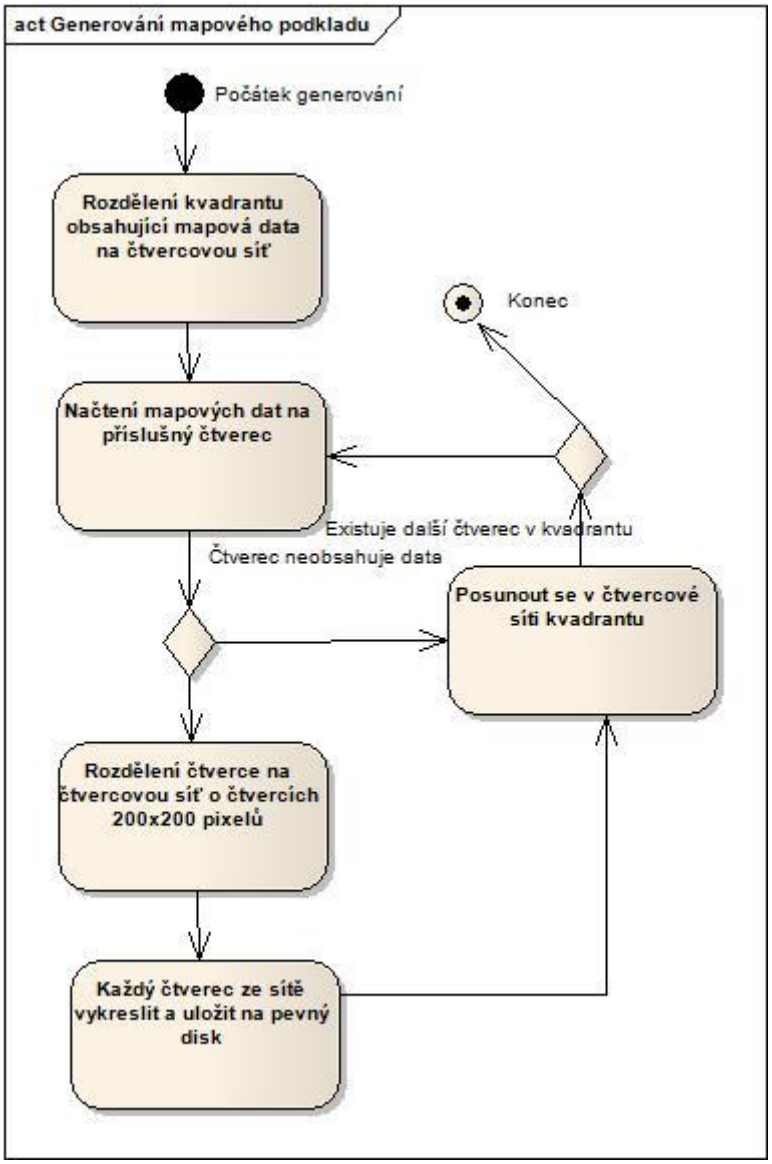
Bohužel i v tomto případě platí, že nic není úplně dokonalé a je potřeba počítat s malými kompromisy. Jedním z největších problémů je počet generovaných obrázků. Teoreticky, kdyby bylo možné používat v aplikaci operátora 10 stupňů přiblížení, pak by bylo potřeba vygenerovat 100 tis. obrázků. Je jasné, že vygenerování takového počtu obrázků zabere mnoho času. Pro názornost – nejvyšší stupeň přiblížení se generuje na počítači o taktu procesoru 3 GHz a plném vytížení zhruba 48 hodin a to i při použití šetrnějšího algoritmu. Je to zapříčiněno ukládáním malých souborů na pevný disk. Protože se ale obrázky generují pouze v případě změny dat v databázovém systému, je potřeba tento postup opakovat jen občas. Odměnou za čekání při generování obrázku je velké urychlení práce s mapovým podkladem u aplikace operátora.

K tomu, aby bylo možné generovat obrázky, je potřeba vytvořit aplikaci, která generování zajistí. Generátor obrázků by měl být schopen běžet na serveru, který absentuje grafické prostředí. A zde nastává problém. Aby bylo možné použít programovací jazyk Java a jeho knihovnu SWT, je potřeba mít právě grafické prostředí nainstalované, jinak není možné SWT použít. Proto aplikace na generování mapového podkladu je napsána pomocí nativních grafických knihoven programovacího jazyka Java.

Princip generátoru je následující. Podle stupně přiblížení rozdělí aplikace mapované území na AxA čtvercovou síť. Pokračuje tak, že načte geografická data odpovídající prvnímu kvadrantu mapy z databázového systému PostgreSQL (aplikace používá speciální úpravu připojení pomocí knihovny pro vytváření datových typů z nadstavby databázového systému PostGIS). Pokud zrovna určitý kvadrant neobsahuje žádná data, je vynechán a pokračuje se dalším kvadrantem. Když kvadrant obsahuje jakákoliv geografická data, pokračuje se druhým krokem algoritmu. Tento krok znamená rozdělení velkého kvadrantu na menší o rozměru 200x200 pixelů.

Menší kvadranty se následně uloží na pevný disk. Název souboru kvadrantu odpovídá jeho poloze – X_x-Y_y.jpg. Velká X a Y znamenají polohu velkého kvadrantu v čtvercové síti a malá x a y znamenají polohu obrázku ve velkém kvadrantu. Díky tomuto pojmenování je možné z jednotlivých obrázků celkem snadno složit celou mapu.

Celý algoritmus je právě díky dvojímu rozdělení velikosti kvadrantů urychlen. Generátor mapového podkladu načítá z databázového systému pouze informace pro tvorbu velkého kvadrantu. Zatížení databázového systému je tak nižší, než kdyby se zjišťovala geografická data pro každý malý obrázek. Způsob generování malých obrázků by však měl jednu výhodu. Tou výhodou je, že by bylo možné snížit počet obrázků tak, že veškeré malé obrázky, které by neobsahovaly jediný geografický údaj, by byly vynechány. V prvním případě jsou takto vynechány pouze velké prázdné kvadranty.



Obr. 23: Algoritmus generování mapového podkladu

Tabulka zobrazující rozdíly mezi mapovými podklady:

Zoom index	Počet kvad.	Z_order	Plochy	Budovy	Popisky
0,03	1	6 – 9	Ne	Ne	Ne
0.05	1	6 – 9	Ne	Ne	Ne
0,1	2x2	6 – 100	Ne	Ne	Ne
0,4	8x7	4 – 100	Ne	Ne	Ne
0,8	16x14	4 – 100	Ano	Ne	Ne
1	20x18	1 – 100	Ano	Ne	Ano
2	39x35	1 – 100	Ano	Ne	Ano
4	78x70	0 – 100	Ano	Ne	Ano
8	155x139	0 – 100	Ano	Ano	Ano
10	194x173	0 – 100	Ano	Ano	Ano

Poznámky k předchozí tabulce:

- Každý kvadrant, který je v tabulce obsažen, obsahuje 20x20 obrázků o rozměru 200x200 pixelů. Hodnoty počtu kvadrantů v tabulce jsou pouze teoretické. Při vynechávání prázdných kvadrantů se počet kvadrantů výrazně sníží. Přesto je počet malých vygenerovaných obrázků velký.
- Zoom index je konstanta určena pro zvětšování.

- Z_order je sloupec v databázi udávající informaci o pořadí zobrazovaných mapových dat. Vykreslování mapových podkladů musí probíhat od nejnižší hodnoty po nejvyšší.
- Zbylé sloupce tabulky označují vrstvy, které se budou vykreslovat pouze při určitém přiblížení.

Načítání mapových podkladů v aplikaci operátora je řešeno načítáním obrázků z webového serveru. Toto načítání provádí speciální vlákno aplikace, aby nedocházelo k prodloužení reakční doby celého grafického rozhraní aplikace. Počet potřebných obrázků pro zobrazení mapového podkladu se určí podle rozměru zobrazovacího zařízení. K tomuto rozměru je potřeba přičíst ještě 400 pixelů pro potřebný buffer. Teorie bufferu pro posouvání obrazu byla popsána v předchozí kapitole. Pro načítání obrázků je možné použít techniku mnoha mini vláken, které slouží pouze pro rychlé paralelní načtení obrázků.

V aplikaci byla použita i technika sestavování bitmapy z načtených obrázků, podobně jako u algoritmů vektorových dat. Výhody bitmapy nejsou ovšem v případě sestavování mapového podkladu z malých obrázků zas tak podstatné. Technika sestavování mapového obrazu z malých obrázků přímo na grafickém zařízení je výhodnější v tom, že při posunu mapy není potřeba opět znovu načítat všechny obrázky, ale pouze jen ty, co jsou potřeba načíst (okrajové obrázky, které načteny nebyly). Pro urychlení vykreslení je možné použít takovou metodu, kdy se nastaví časový interval, po který se musí načíst všechny potřebné obrázky. Pokud se obrázky v časovém intervalu nepodaří načíst, pak je místo obrázku zobrazena informace o chybě načtení.

Metoda přímého zobrazování malých obrázků nemá jen výhody, ale i jednu nevýhodu. Je jí o něco vyšší režie. Zajímavé by bylo prolnout obě jmenované metody a použít vždy výhodnější metodu zobrazení pro daný okamžik. Pro pohyby mapového podkladu je výhodnější metoda okamžitého skládání z důvodu načítání pouze okrajových oblastí, pro překreslování zobrazovacího zařízení, tedy pro statické zobrazení mapového podkladu, což je například pohyb vozidla, je zas mnohem výhodnější složení obrázků do bitmapy a vykreslování pouze jediné bitmapy, protože toto řešení vyžaduje menší režii při častém překreslování. Nejlepším řešením by bylo vykreslování měnících se mapových podkladů přímo pomocí malých obrázků a na pozadí aplikace sestavování statického obrazu. Jakmile je statický obraz vytvořen a nastane vhodná situace ho použít, použije se. Když nastane pohyb mapového podkladu, dojde k opětovnému použití algoritmu malých obrázků a jeho sestavení na pozadí statického obrazu.

5.2 Přichycení entity k cestě

Přichycení entity k cestě by se dalo formulovat jako problém mapování souřadnic vozidla na souřadnice silnice mapového podkladu. Určitě existuje mnoho různých složitých algoritmů. V diplomové práci je použit algoritmus, který řeší přichycení vozidla k cestě pomocí funkcí balíčku PostGIS databázového systému PostgreSQL.

Tento algoritmus je velmi vhodný a účinný. V podstatě funguje na principu vyhledávání nejvhodnějšího bodu přichycení v určitém kvadrantu. Zajisté by bylo výhodnější aproximovat kruh mnohoúhelníkovým polygonem než čtvercem, ale pro diplomovou práci je čtverec dostačující. Funkce balíčku PostGIS obsahují několik přímo vhodných funkcí.

Vstupem do algoritmu je poloha vozidla. Podle polohy vozidla se následně jednoduše připraví souřadnice kvadrantu tak, že se přičte nebo odečte 0.01 od výchozích souřadnic. Prvním krokem algoritmu je tedy vytvoření akčního prostoru, ve kterém se bude vyhledávaný připojovací bod nacházet. V databázovém systému se vyhledají všechny cesty, které mají alespoň jeden bod v tomto kvadrantu. Na cesty je ještě aplikováno pravidlo, že se nejedná o polní a lesní stezky, ale pouze ty, které jsou zpevněné. U těchto cest se zjistí vzdálenost a dotaz do databázového systému vrátí záznam tabulky který představuje cestu, která má vzdálenost nejnižší.

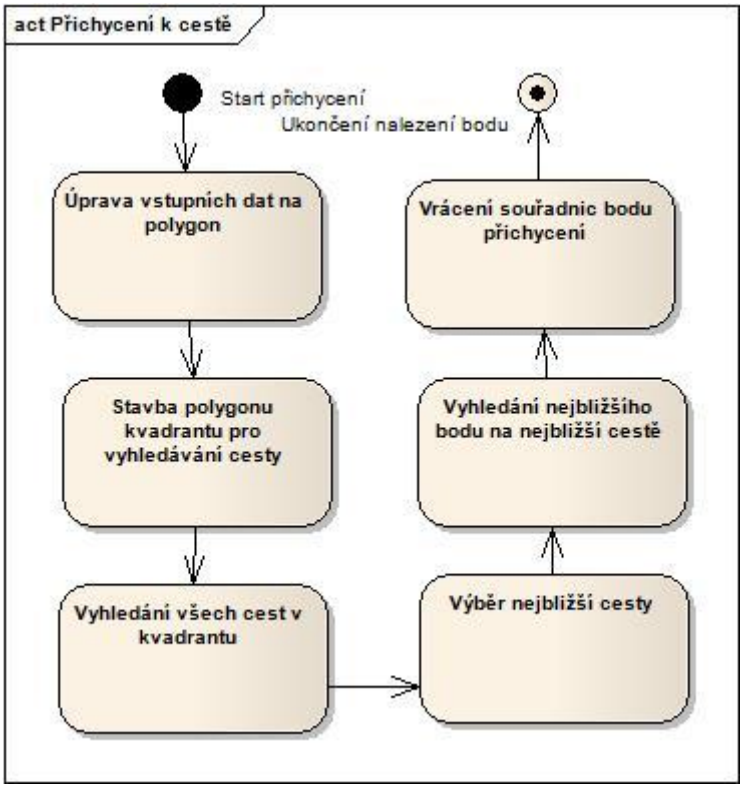
Posledním krokem algoritmu je zjištění souřadnice bodu cesty, který je nejblíže souřadnic polohy vozidla. I pro řešení tohoto problému obsahuje databázový systém PostgreSQL ve svém balíčku PostGIS vhodnou funkci. Celý algoritmus vrací kromě souřadnic i název cesty pro zobrazení detailních informací o vozidle. Algoritmus je možné vylepšit pomocí PL/SQL. Vytvořením funkce s proměnnými by se databázový dotaz zjednodušil a urychlil.

Popis funkcí algoritmu:

ST_PolygonFromText('POLYGON((16.08 50.47,16.08 50.48,16.09 50.48,16.09 50.47, 16.08 50.47))')	Funkce slouží k tvorbě databázového datového typu geometry typu polygon. Souřadnice musí tvořit uzavřený prostor.
ST_Intersects(geometry, geometry)	Funkce vrací TRUE, pokud mají oba parametry datového typu geometry společný jediný bod.
ST_GeomFromText('POINT(16.089694 50.48124)'),	Funkce vytváří prvek datového typu geometry. Geometry je typu point.
distance(geometry, geometry)	Funkce vrací vzdálenost mezi dvěma parametry typu geometry.
ST_line_locate_point(geometry, geometry)	Vrací 0 nebo 1 v závislosti nejbližšího bodu. Slouží k zjišťování nejbližšího bodu čáry k jinému bodu ležícímu mimo čáru.
st_line_interpolate_point(geometry, float)	Funkce vrací bod z čáry datového typu geometry, odpovídající závislosti funkce ST_line_locate_point

Funkce pro uchycení vozidla k cestě a příklad jejího použití:

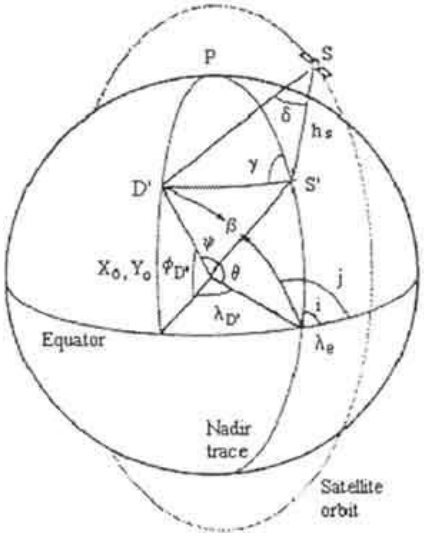
```
CREATE OR REPLACE FUNCTION glue_to_line(bod GEOMETRY) RETURNS SETOF RECORD
AS
$BODY$

DECLARE newPoint GEOMETRY;
DECLARE pomPoint GEOMETRY;
DECLARE newLine GEOMETRY;
DECLARE xPoint double precision;
DECLARE yPoint double precision;
DECLARE polygon GEOMETRY;
DECLARE polygonConstant FLOAT;
DECLARE nameField character varying(50);
DECLARE trash double precision;
BEGIN
      polygonConstant:=0.01;
      pomPoint :=setsrid(bod, 4326);
      xPoint := x(pomPoint);
      yPoint := y(pomPoint);


      polygon := setsrid(ST_PolygonFromText('POLYGON(( ' ||
      xPoint - polygonConstant ||' '|| yPoint - polygonConstant ||' ,' ||
      xPoint - polygonConstant ||' '|| yPoint + polygonConstant ||' ,' ||
      xPoint + polygonConstant ||' '|| yPoint + polygonConstant ||' ,' ||
      xPoint + polygonConstant ||' '|| yPoint - polygonConstant ||' ,' ||
      xPoint - polygonConstant ||' '|| yPoint - polygonConstant || '))'),
      4326);


      SELECT * INTO nameField, newLine, trash FROM (SELECT name, way,
distance(way,  pomPoint) FROM osm_line WHERE ST_Intersects(way, polygon) =
TRUE ORDER BY distance FETCH FIRST 1 ROW ONLY) AS vzdalenost;


      SELECT (st_line_interpolate_point(newLine,
ST_line_locate_point(newLine,  pomPoint))) INTO newPoint;


      RETURN QUERY SELECT newPoint, newLine, nameField;
END;
$BODY$
  LANGUAGE 'plpgsql' VOLATILE
  COST 100;


SELECT * INTO newPoint, pomLine from glue_to_line(pomPoint)  f(point
geometry, way geometry, name character varying(50));
```



Obr. 24: Princip přichycení entity k cestě

5.3 Zobrazení satelitů

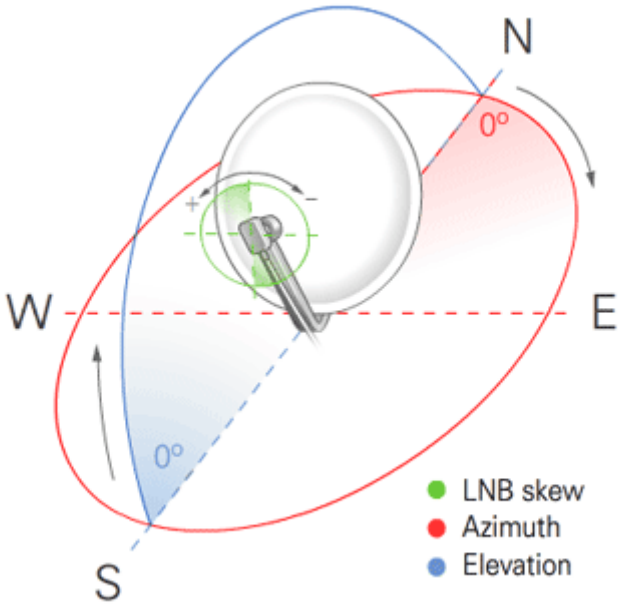
Jedním ze zajímavějších zobrazovacích algoritmů je vizuální zobrazení satelitů. Poloha satelitů vůči pozorovateli je udávána pomocí dvou polohových informací: azimut a elevace. Obě dvě hodnoty udávají úhly, pod kterými je možné satelit od pozorovatele zahlédnout.



Obr. 25: Zobrazení oběhu družice [19]

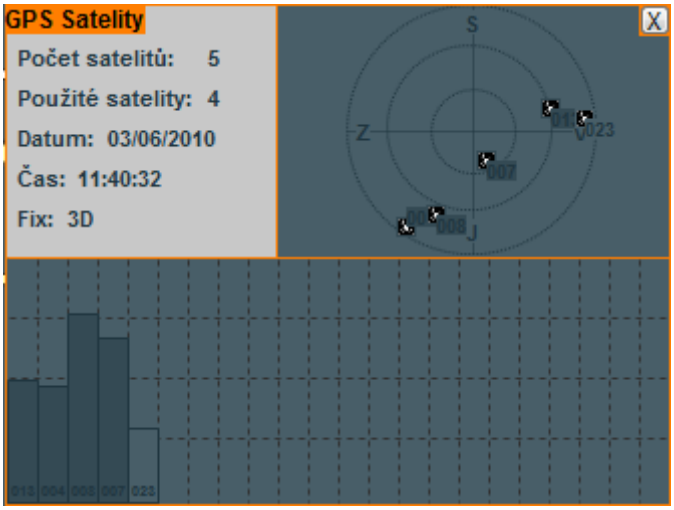
Azimut je geografický orientovaný úhel. Počítá s 0-360° a znázorňuje směr. Nultý stupeň znázorňuje sever, 90° východ, 180° jih a 270° západ. Elevace je opět orientovaný úhel, který znázorňuje náměr. Úhel elevace svírá GPS modul spolu se satelitem. Rozsah úhlu elevace je 0-90°, kde 90° znamená, že satelit je přímo nad GPS modulem.

Jak je z obrázku 26 zřejmé, údaje o poloze satelitů jsou udávány ve třech dimenzích. K tomu, aby bylo možné zobrazit polohu satelitu v aplikaci o dvou dimenzích, je potřeba provést sjednocení hodnot a převést výpočet pomocí goniometrických funkcí na kruh.



Obr. 26: Zobrazení pojmů azimut a elevace [18]

Na obrázku 27 je vidět, že satelity lze zobrazit v kruhu. Pozice satelitu v kruhu se určuje právě podle azimutu a elevace. Hodnota azimutu určuje, jaký úhel bude satelit svírat s počátkem. Nulový úhel udává sever. Elevace je zastoupena vzdáleností satelitu od středu kružnice. Pokud má elevace úhel nula stupňů, pak satelit leží na okrajové kružnici. Pokud ovšem je elevace 90°, pak leží satelit přímo ve středu kruhu.



Obr. 27: Reprezentace GPS satelitů v diplomové práci

Třetí zajímavou hodnotou, která se týká satelitů a kterou ale už vypočítává GPS modul, je hodnota SNR (Signal-to-Noise Ratio). Hodnota udává sílu signálu satelitu. Je vypočtena podílem hodnoty signálu satelitu a hodnoty zarušení signálu.

Jak vykreslit polohu satelitu v kruhu? Pomocí složení dvou goniometrických výpočtů:

```
Y = stredY - (r * cos(elevace) * cos(azimuth));
X = stredX + (r * cos(elevace) * sin(azimuth));
```

Malé upozornění ale patří tomu, že výpočty se provádějí v radiánech, nikoliv stupních. Proto je potřeba elevace a azimuth převést na radiány.

6. Výpočet optimální trasy

Popis výpočtu optimální cesty je zahrnut v analýze a návrhu aplikace operátora. Vzhledem k časové tísni nebyla tato funkce prozatím implementována. Tato funkcionalita slouží pro naplánování optimální trasy okruhu, který musí řidič projet. Do aplikace operátora byla přidána, aby umožnila co nejefektivnější projetí trasy a zajistila tím co nejrychlejší dodání zásilky a úsporu finančních nákladů na provoz vozidla. Algoritmus pro výpočet optimální trasy je zaměřen na optimalizaci délky trasy.

Řešení předpokládá využití Dijkstrova algoritmu hledání nejkratší cesty v grafu. Algoritmus vyhledává nejkratší okruhovou cestu podle zadaného kvadrantu. Pro účel diplomové práce je dostatečně efektivní a rychlý. Cesta je vyhledávaná podle umístění odesílatelů/příjemců/dep v daném kvadrantu v určitý den. Každý odesílatel/příjemce/depo představuje bod začátku/konce úseku plánované trasy. Dijkstrův algoritmus se použije dvakrát. Poprvé se určí nejvhodnější pořadí bodů, které musí řidič navštívit, v druhém kroku se algoritmus aplikuje na každý dílčí úsek. Z těchto úseků se pak sestaví celá trasa. Druhý krok algoritmu je složitý a pomalý, protože se musí využívat funkcí balíčku PostGIS databázového systému PostgreSQL .

Popis Dijkstrova algoritmu [8]:

Mějme graf G, v němž hledáme nejkratší cestu. Řekněme, že V je množina všech vrcholů grafu G a množina E obsahuje všechny hrany grafu G. Algoritmus pracuje tak, že si pro každý vrchol v z V pamatuje délku nejkratší cesty, kterou se k němu dá dostat. Označme tuto hodnotu jako $d[v]$. Na začátku mají všechny vrcholy v hodnotu $d[v]=\infty$, kromě počátečního vrcholu s, který má $d[s]=0$. Nekonečno symbolizuje, že neznáme cestu k vrcholu.

Dále si algoritmus udržuje množiny Z a N, kde Z obsahuje už navštívené vrcholy a N dosud nenavštívené. Algoritmus pracuje v cyklu tak dlouho, dokud N není prázdná. V každém průchodu cyklu se přidá jeden vrchol $vmin$ z N do Z, a to takový, který má nejmenší hodnotu $d[v]$ ze všech vrcholů v z N.

Pro každý vrchol *u*, do kterého vede hrana (označme její délku jako *l(vmin,u)*), z *vmin*, se provede následující operace: pokud (*d[vmin] + l(vmin,u)*) < *d[u]*, pak do *d[u]* přiřaď hodnotu *d[vmin] + l(vmin,u)*, jinak neprováděj nic. Až algoritmus skončí, potom pro každý vrchol *v* z *V* je délka jeho nejkratší cesty od počátečního vrcholu *s* uložena v *d[v]*.

Algoritmus výpočtu optimální trasy vychází ze zobrazovacího algoritmu přichycení entity k cestě. První a poslední bod optimalizované cesty se přesně tak i volí. Mezi první a poslední body patří i body trasy, jako je například depo, nebo příjemce. Dalším složitým a výpočetně náročným krokem hledání cesty, je hledání dalšího bodu cesty. Opět se na hledání dalšího bodu používá modifikovaný algoritmus přichycení entity k cestě s tím rozdílem, že může vracet více jak jeden bod. Může vracet sice více jak jeden bod, ale na současné cestě, která obsahuje současný bod, může vrátit maximálně 2 body. Každým směrem pouze jeden bod. Pokud ovšem vrátí dva a více bodů, znamená to, že se na současnou cestu napojuje jiná cesta. Z toho lze usoudit, že se jedná o křižovatku a možné větvení silnice. V tomto okamžiku může nastat malý problém, protože v databázovém systému není nikde přímo definováno spojování dvou a více silnic, takže křižovatky algoritmus pouze odhaduje. Je vidět, že je potřeba se neustále dotazovat databázového systému na další a další bod cesty. Rozhodně nemá cenu se pokoušet vyhledávat nejkratší cestu například na polních a lesních cestách, tím se sníží počet možných výsledků.

Dvě možnosti efektivního řešení:

- Načítání všech potřebných bodů přímo do aplikace operátora.
- Využívat Dijkstrův algoritmus přímo na straně databázového systému.

Řešení na straně databázového systému představuje optimální možnost. V závislosti na době výpočtu a zatížení databázového systému při výpočtu optimální cesty by bylo vhodné, pokud bude výpočet časově nebo výpočtově náročný, ukládat plánovanou trasu hned do databázového systému. Pokud ovšem bude výpočet rychlý a jednoduchý, je zbytečné tuto trasu ukládat. Už jen proto, že se trasa může dynamicky měnit podle event. nových požadavků na odeslání zásilky.

Databázový systém by pak vracel po úspěšném nalezení trasy přímo celou trasu, nebo její úsek v datovém typu geometry. Výpočty by zůstaly na straně výkonného serveru a aplikace operátora by zůstala bez výpočtové zátěže. Výhoda výpočtu trasy na straně databázového serveru by se projevila i u řidičů. Mohli by si tak při změně trasy znovu přepočítat optimální trasu a exportovat ji do souboru GPX pro použití v jejich GPS navigacích. Zvýšila by se tím efektivita reakce na případnou změny trasy.

Po vypočtení nejkratší trasy je trasa v aplikaci operátora vizualizovaná a proložena do mapového podkladu společně s vrstvou bodů, podle kterých se trasa sestavuje. Další možnou vrstvou je zobrazení hranice působiště řidiče.

7. Kniha jízd

Velmi důležitým prvkem aplikace je kniha jízd. Kniha jízd je služba primárně zpracovávaná databázovým systémem PostgreSQL za pomocí balíčku PostGIS. Díky tomuto systému je možné vytvořit knihu jízd bez nutnosti zasílání velkého množství geografických informací z GPS přístroje.

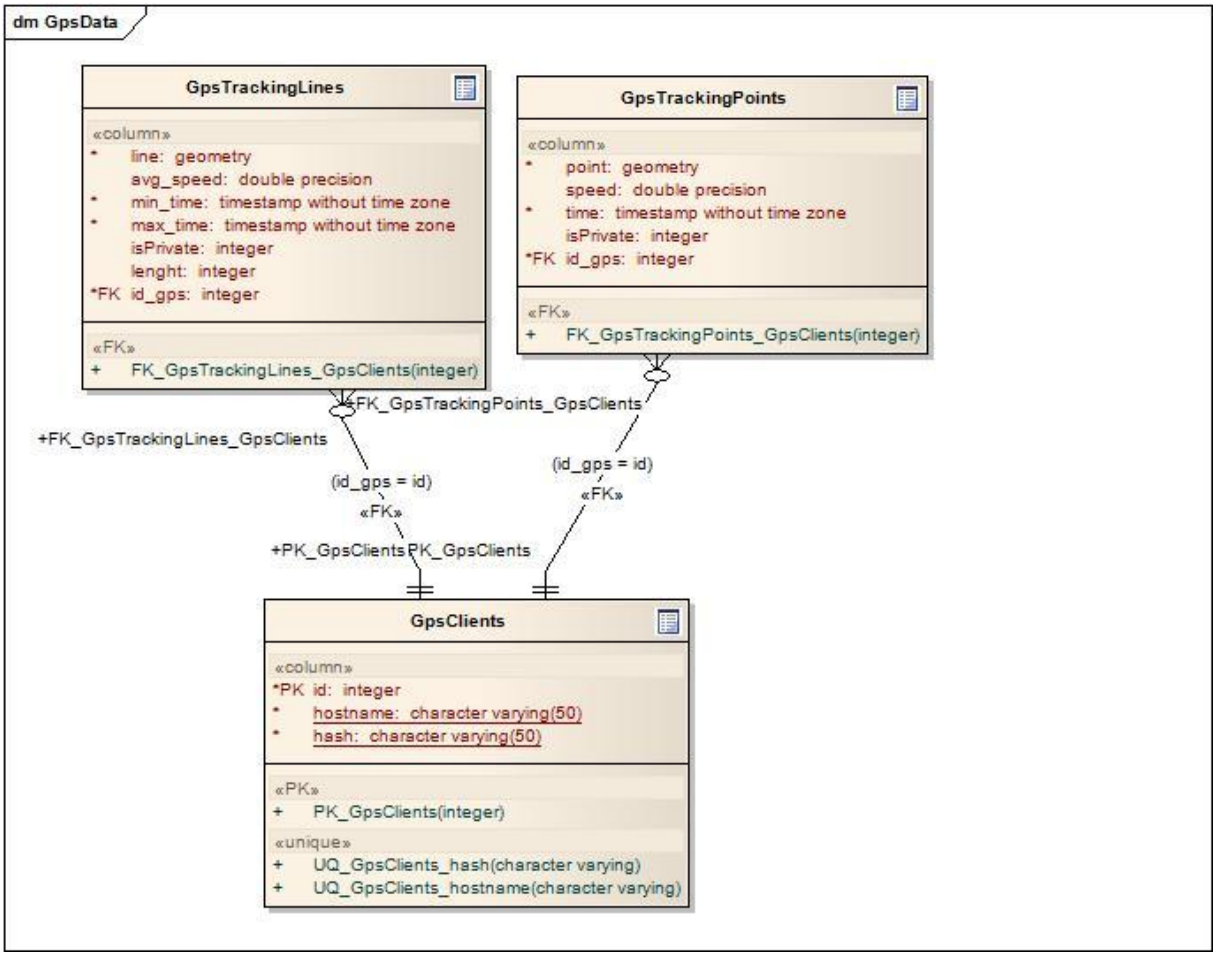
GPS přístroj generuje velké množství geografických dat, často mnoho i za jednu sekundu. K tomu, aby se taková kvanta dat dala zasílat na server prostřednictvím spojení realizované službou GPRS, došlo by během několika prvních sekund k zahlcení přenosového pásma, které poskytuje modem.

Proto je nutné nastavit v GPS klientské aplikaci časové intervaly pro odesílání zpráv. Při testování byly použity intervaly v trvání 10-15 sekund (v podstatě jde o 10 zpráv zachycených na deskriptoru GPS modulu – tyto hodnoty při zhruba třech satelitech odpovídají času 10 sekund). Pomocí konfiguračního souboru je možné tyto údaje libovolně měnit. Časové intervaly těchto zpráv byly pro vytvoření knihy jízd shledány jako dostačující. Důvod je prostý. Tyto údaje ve finále slouží pouze jako data aproximace, kterou právě PostGIS umožňuje.

Po odeslání SOAP zprávy obsahující informace o poloze tvorba knihy jízd pro GPS klienta končí. Vše ostatní se odehrává na straně serveru. Jakmile SOAP server příjme zprávu, zjistí ID GPS klienta a uloží zprávu do databázového systému, přesněji do tabulky GpsTrackingPoints:

point	Do tohoto sloupce se ukládá datový typ geometry spatial nadstavby PostGIS. Geometrie je typu point.
speed	Položka je informací o rychlosti GPS klienta, datový typ byl zvolen double.
time	Čas vysílaný z GPS satelitů, důležitá informace pro následné zpracování, opět datový typ double.
isPrivate	Tento záznam informuje o tom, zda je cesta soukromá, nebo služební. Pomocí tohoto ukazatele se hodnoty zpracovávají do různých tabulek.
id_gps	V tomto případě se nejdená o nic jiného, než ID GPS klienta z tabulky GpsClients.

Samozřejmě se může objevit otázka: Pokud server zachytává řekněme každých 15 sekund zprávu o poloze, bude tabulka brzo přetékat záznamy. A odpověď by zněla, ano. Ve skutečnosti tomu tak ale není. Stejně jako je možné generovat datový typ geometry z jediného bodu, je možné spojovat již vytvořené geometry do struktury line (čáry). V podstatě je potřeba vytvořit vhodný databázový trigger, který bude vykonán po vložení záznamů. Trigger musí umět, podle vhodných požadavků, transformovat data do typu line a přesunout na jiné, správné místo. Původní data jsou z tabulky GpsTrackingPoints průběžně mazána.



Obr. 28: Struktura tabulek v databázovém systému potřebných pro tvorbu knihy jízd

Z obrázku 28 je jasné, že je potřeba přesouvat data do tabulky GpsTrackingLines. Tabulka je upravena tak, aby mohla uchovávat jak soukromé, tak služební cesty. Pokud se jedná o služební cestu, GPS klient má přepnutý spínač na hodnotu 0. Tuto hodnotu zašle pomocí SOAP zprávy a vloží do tabulek jako příznak isPrivate = 0. Pokud se jedná o soukromou cestu, je spínač i příznak přepnut na hodnotu 1. V zobrazení knihy jízd je pak rozdíl znatelný v tom, že obsluha knihy jízd nebude schopna vizuálně graficky zobrazit projetou soukromou trasu, pouze trasu služební.

GpsTrackingLines má strukturu:

line	Jedná se o datový typ geometry, který má podobný význam jako u GpsTrackingPoints. Jen uchovává místo jediného bodu celou generovanou cestu.
avg_speed	Už název naznačuje, že se jedná o průměrnou rychlost za celou cestu.
min_time	Celá cesta je generována na časovém intervalu, min_time je dolní hranice.
max_time	Horní hranice intervalu.
isPrivate	Tento záznam informuje o tom, zda je cesta soukromá, nebo služební. Pomocí tohoto ukazatele se hodnoty zpracovávají do různých tabulek.
length	Délka ujeté trasy. Pole se vyplňuje pouze tehdy, je-li trasa namapována.
id_gps	Opět se jedná o ID GPS modulu.

Jak již bylo v předchozích řádcích uvedeno, k přesunu mezi tabulkami dochází pomocí databázového triggeru. Pro přesun musí být splněny následující podmínky:

- Trigger najde v GpsTrackingPoints záznam s rychlostí nižší, než povolená.
- Vozidlo není v pohybu déle než dvě minuty.

Pokud je splněna jedena z uvedených podmínek, trigger data přesune do jiné, správné tabulky.

Následující popsané dělení dat nepředstavuje ten nejjemnější způsob, ale je to dobrý základ pro nastávající aproximaci. Aproximace vychází z algoritmu pro uchycení vozidla k silnici.

Při spojení všech bodů struktury line v GpsTrackingLines je geometrickým výsledkem lomená čára. Při každém spojení dvou bodů bohužel vzniká dost velká chyba, která je závislá na délce kroku. Sečtením všech chyb by výsledkem byla nemalá ztráta.

Proto je výhodné pro tvorbu knihy jízd použít aproximaci. Pokud se uchytí každý bod projeté cesty, uložené v databázi, na mapový podklad, který je zatížen mnohem menší chybou než zdrojová data, není nutné spojovat zdrojová data do podoby velmi nepřesné lomené čáry, ale spojit data mapového podkladu. Využije se stejný algoritmus přichycení bodu na trasu, jako při uchycení vozidla na cestu. Takto budou zdrojová data projeté cesty sloužit při výpočtu pouze jako aproximované krajní body intervalu úseku cesty. Tyto úseky se následně spojí do jednoho celku. Chyba se díky aproximaci sníží pouze na chybu nepřesnosti mapového podkladu. Jedná se o celkem náročnou operaci, proto je tento problém v diplomové práci vyřešen změnou dat zdrojových za výslednou aproximaci. Mapování databázového datového typu geometry ze zdrojových dat za výslednou aproximaci se navenek projeví nastavením pole *length*, kde se ukládá délka projeté trasy. Toto pole je vyplněno pouze po provedení mapování projeté trasy na geografická data. Mapování projeté trasy na mapový podklad se provádí pomocí triggeru po vkládání do tabulky GpsTrackingLines.

Kniha jízd je k dispozici pouze v aplikaci operátora. Umožňuje prohlížet seznam projetých tras seřazených podle datumu. Všechny trasy ze stejného dne jsou spojeny do jedné trasy. Trasy je možné v aplikaci operátora i vizualizovat. V případě vizualizace je potřeba zvolit všechny trasy, které mají být vizualizovaný (vizualizovat lze pouze trasy označené jako služební).

Vizualizace projeté služební trasy je složena ze tří vrstev. Na nejnižší vrstvě je mapový podklad, stejný jako se používá v případě sledování vozidla. Na druhé vrstvě je zobrazena vektorově projetá trasa prolnutá do mapového podkladu. Na poslední volitelné vrstvě jsou zobrazeny body rozvozu zásilky (odesílatelé/příjemce/depa). Na stranách úseků projetých tras jsou zobrazeny informace o délce, průměrné rychlosti a časovém intervalu projetí úseku.

8. Závěr

Závěrem je dobré zamyslet se nad možnými dalšími rozšířeními, modifikacemi a změnami. Hned první z možných změn, které by bylo vhodné v systému provést, je změna aplikace operátora na aplikaci webovou. Důvodů je k tomuto rozhodnutí hned několik. Nejdůležitějším důvodem je především bezpečnost. Z hlediska bezpečnosti je vhodnější přenášet přes Internet co nejmenší objem dat. Pokud by byla aplikace operátora integrována někde v clusteru majitele systému a zákazníkovi by se poskytovala pouze služba, pak by veškeré datové přenosy probíhaly pouze v rámci datového clusteru, nebo serverové farmy. Vedle bezpečnosti datových přenosů se tím řeší i jejich rychlost. V rámci serverové farmy je zpravidla k dispozici optické vedení. U současného řešení potřebuje počítač operující s aplikací operátora rychlejší datové připojení z důvodu přenosů obrázků. Integrovaná webová aplikace by tento problém eliminovala. V neposlední řadě by odpadla i nutnost vlastnit autorizační server v rámci systému.

Dalším krokem by mohlo být i porovnání různých mapových podkladů v aplikaci operátora. Použité mapové podklady jsou místy málo detailní a použití mapových podkladů se 100%-ním pokrytím popisných čísel by usnadnilo vyhledávání optimální trasy. Kromě toho by aplikace mohla obsahovat aktivní body zájmu, jako jsou městské webové kamery. Rovněž benzínové stanice by se mohly používat jako body zájmu při plánování trasy pro automatické zastávky vozidla v závislosti na stavu paliva v nádrži a vypočtené spotřebě vozidla. A v neposlední řadě i pro zákonem předepsané pravidelné přestávky řidičů-

Návrh systému pro sledování zásilek by bylo vhodné rozšířit o vytvoření vlastní GNU/Linux embedded distribuce určené pro jednoúčelové sledovací zařízení ve vozidle. Takovýto GNU/Linux je možné maximálně minimalizovat. V rámci testů byla zkoušeno speciálně upravená distribuce Debian pro běh z flash paměti. Podařilo se Debian Linux dostat na flash paměť s kapacitou 512 MB. Z důvodu omezeného časového prostoru bylo od dalších testů upuštěno.

Zajímavým počinem by bylo i napsat *GPS klienty* pracující na mobilních operačních systémech, jako jsou Windows Mobile, Android či Symbian. Důvodů proto je hned několik. V dnešní době mají mobilní telefony, postavené na jednom z těchto operačních systémů, běžně integrovaný GPS modul, který má sériové rozhraní a podporuje protokol NMEA. Pořizovací cena těchto zařízení je řádově nižší, než cena speciálních zařízení pro sledování. Další výhodou je i možnost snadné práce se zpětnými zprávami GPS klientovi. Dokonce by toto řešení bylo výhodné i v případě integrování GPS navigaci do aplikace *GPS klienta*.

V neposlední řadě se nabízí rozšíření *GPS klienta* o napojení systému na palubní počítač vozidla. Umožnilo by to získávat přesné informace o rychlosti, spotřebě a množství paliva v nádrži přímo od vozidla. Systém by tak získával naprosto přesné údaje. Všechny tyto informace lze přeměnit na indicie při plánování optimální trasy. Další výhodou napojení na palubní počítač by byla i základní diagnostika vozidla.

Téma diplomové práce bylo velmi zajímavé a poučné. Nabízí se pokračovat v tomto projektu dál a dovést ho do produkční fáze, kdy je možné aplikaci volně používat ke sledování zásilek. Většinu informací a poznatků z diplomové práce je možné použít i na budování jiných sledovacích softwarů, než je sledování zásilek. Jsem přesvědčen, že aplikace pracující s geografickými daty budou v krátké době ještě mnohem více žádanější než nyní. Podle průzkumu jednoho konkurenčního systému věřím, že nebude dlouho trvat a zákazníci začnou hledat lepší Internetové řešení. To by mohla být vhodná příležitost pro aplikace postavené na této diplomové práci. Všechno naznačuje, jako například zvyšování oblíbenosti hry Geocaching, že se zákazníci už naučili věřit a používat GPS navigace. Především v automobilové dopravě neřekly GPS navigace ještě své poslední slovo. Přímo se nabízí použít ji pro sledování odcizených vozidel policií.

9. Seznam použité literatury

[1] KISZKA, Bogdan. *1001 tipů a triků pro jazyk Java*. Brno : Computer Press, a. s., 2009. 542 s. ISBN 987-80-251-2467-3, K1687.

[2] JELÍNEK, Lukáš. *Jádro systému Linux : Kompletní průvodce programátora*. Brno : Computer Press, a. s., 2008. 686 s. ISBN 987-80-251-2084-2, K1473.

[3] URMAN, Scott; HARDMAN, Ron; MCLAUGHLIN, Michael. *Oracle : Programování v PL/SQL*. Brno : Computer Press, a. s., 2007. 720 s. ISBN 978-80-251-1870-2, K1468.

[4] GPS eXchange Format. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2010-08-16]. Dostupné z WWW: <http://en.wikipedia.org/wiki/GPS_eXchange_Format>.

[5] Standard Widget Toolkit. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2010-08-16]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Standard_Widget_Toolkit>.

[6] *Programming tutorials and source code examples* [online]. 2009 [cit. 2010-08-16]. Dostupné z WWW: <http://www.java2s.com/>.

[7] *PostGIS 2.0.0SVN Manual* [online]. 2009 [cit. 2010-08-16]. Dostupné z WWW: <http://postgis.refractions.net/documentation/manual-svn/>.

[8] Dijkstrův algoritmus. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2010-08-16]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Dijkstrův_algoritmus>.

[9] SOAP. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2010-08-16]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/SOAP>.

[10] Web Services Description Language. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2010-08-16]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Web_Services_Description_Language>.

[11] GUNATHILAKE, Lahiru. *WSO2 Oxygen Tank* [online]. 2008 [cit. 2010-08-16]. Web Services are Faster – Apache Axis2/C Performance: Round 1. Dostupné z WWW: <http://wso2.org/library/3532>.

[12] *Apache Axis2/C* [online]. 2005 [cit. 2010-08-16]. Dostupné z WWW: <http://ws.apache.org/axis2/c/>.

[13] *CzechDUINO* [online]. 2010 [cit. 2010-08-16]. Dostupné z WWW: <http://www.czechduino.cz>.

[14] *CSOAP* [online]. 2006 [cit. 2010-08-16]. Dostupné z WWW: <http://csoap.sourceforge.net/>.

[15] *PostgreSQL 8.4.4 Documentation* [online]. 2010 [cit. 2010-08-16]. Dostupné z WWW: <http://www.postgresql.org/docs/8.4/interactive/index.html>.

[16] Global Positioning System. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, [cit. 2010-08-16]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Global_Positioning_System>.

[17] DLOUHÝ, Martin. *Robotika.cz* [online]. 2006-06-27 [cit. 2010-08-16]. GPS. Dostupné z WWW: <http://robotika.cz/guide/gps/cs>.

[18] *SatCentrum.eu* [online]. 2006 [cit. 2010-08-16]. Parametry pro nastavení satelitní paraboly. Dostupné z WWW: <http://www.satcentrum.eu/showpage.php?name=nastav>.

[19] *GISdevelopment* [online]. 1997 [cit. 2010-08-16]. Surface Temperatrue Retrieval from Satellite Data Over South China sea . Dostupné z WWW: <http://www.gisdevelopment.net/aars/acrs/1997/ps2/ps4019.asp>.

[20] *GPS* [online]. 2000 [cit. 2010-08-16]. Dostupné z WWW: <http://gps.slansko.cz/>.

10. Přílohy

10.1 Databázový trigger pro ukládání GPS bodů

```
CREATE OR REPLACE FUNCTION gps_points_to_lines()
  RETURNS trigger AS
$BODY$
DECLARE max_time_interval INTERVAL;
DECLARE min_speed INTEGER;
BEGIN
      min_speed := 0;
      max_time_interval := INTERVAL '1 minute';
      IF NEW.speed <= min_speed THEN
            INSERT INTO gpstrackinglines (SELECT
makeline_garray(gps_array_accum(point)), avg(speed), min(time), max(time),
isPrivate, 0 lenght, id_gps FROM gpstrackingpoints WHERE id_gps =
NEW.id_gps GROUP BY id_gps, isPrivate);
            DELETE FROM gpstrackingpoints WHERE id_gps = NEW.id_gps;
            RETURN NULL;
      ELSE
            IF (SELECT age(NEW.time, max(time)) >= max_time_interval FROM
gpstrackingpoints WHERE id_gps = NEW.id_gps GROUP BY id_gps) THEN
                  INSERT INTO gpstrackinglines (SELECT
makeline_garray(gps_array_accum(point)), avg(speed), min(time), max(time),
isPrivate, 0 lenght, id_gps FROM gpstrackingpoints WHERE id_gps =
NEW.id_gps GROUP BY id_gps, isPrivate);
                  DELETE FROM gpstrackingpoints WHERE id_gps = NEW.id_gps;
                  RETURN NEW;
            END IF;


            INSERT INTO gpstrackinglines (SELECT
makeline_garray(gps_array_accum(point)), avg(speed), min(time), max(time),
isPrivate, 0 lenght, id_gps FROM gpstrackingpoints WHERE id_gps =
NEW.id_gps AND date_trunc('hour', time) < date_trunc('hour',
current_timestamp) GROUP BY id_gps, isPrivate, date_trunc('hour', time));


            DELETE FROM gpstrackingpoints WHERE id_gps = NEW.id_gps AND
date_trunc('hour', time) < date_trunc('hour', current_timestamp);
            RETURN NEW;
      END IF;
      RETURN NULL;
END;
$BODY$
  LANGUAGE 'plpgsql' VOLATILE
  COST 100;



CREATE TRIGGER gps_make_lines BEFORE INSERT ON gpstrackingpoints
  FOR EACH ROW EXECUTE PROCEDURE gps_points_to_lines();
```

10.2 Databázový trigger a funkce pro tvorbu projeté trasy

```
CREATE OR REPLACE FUNCTION gps_track_to_maplines() RETURNS trigger AS
$BODY$
BEGIN
      NEW.line := build_route(NEW.line);
      NEW.length := length(NEW.line);
      RETURN NEW;
END;
$BODY$
  LANGUAGE 'plpgsql' VOLATILE
  COST 100;
```

```
CREATE OR REPLACE FUNCTION build_route(lines GEOMETRY) RETURNS GEOMETRY AS
$BODY$
DECLARE points_number INTEGER;
DECLARE points_count INTEGER;
DECLARE points_numberSubLine INTEGER;
DECLARE editLine GEOMETRY[];
DECLARE pomPointsArray GEOMETRY;
DECLARE pointsArray GEOMETRY;
DECLARE pomPoint GEOMETRY;
DECLARE newPoint GEOMETRY;
DECLARE newLine GEOMETRY;
DECLARE newLinePart GEOMETRY;
DECLARE oldLength float;
DECLARE newLength float;
DECLARE pomLine GEOMETRY;
DECLARE oldLine GEOMETRY;
DECLARE isSet BOOLEAN;
BEGIN
      isSet := false;
      points_count:= 1;
      FOR points_number IN 1..numpoints(lines) LOOP
            BEGIN
                  pomPoint :=setsrid(pointn(lines, points_number), 4326);

                  SELECT * INTO newPoint, pomLine from
glue_to_line(pomPoint)  f(point geometry, way geometry, name character
varying(50));

                  IF (isSet = false) THEN
                        BEGIN
                              isSet := true;
                              editLine[points_count]:= newPoint;
                              points_count:= 1 + points_count;
                        END;
                  ELSE
                        BEGIN
                              IF (geometry_same(pomLine, oldLine)) THEN
                                    BEGIN
                                          points_numberSubLine := 0;
```

```
                                        pomPointsArray := null;
                                        pomPointsArray :=
line_part(pomLine, editLine[points_count - 1], newPoint);
                                        IF (numpoints(pomPointsArray) >
0) THEN


                                              BEGIN
                                                    FOR
points_numberSubLine IN 1..numpoints(pomPointsArray) LOOP
                                                          BEGIN

      editLine[points_count]:= pointn(pomPointsArray,
points_numberSubLine);
                                                          points_count:=
1 + points_count;


                                                          END;
                                                    END LOOP;
                                              END;
                                        END IF;
                                  END;
                            ELSE
                                  BEGIN
                                        points_numberSubLine := 0;
                                        pomPointsArray := null;
                                        pomPointsArray :=
subroute(editLine[points_count - 1] , newPoint, oldLine, pomLine);
                                        FOR points_numberSubLine IN
1..numpoints(pomPointsArray) LOOP
                                              BEGIN
                                              editLine[points_count]:=
pointn(pomPointsArray, points_numberSubLine);
                                                    points_count:= 1 +
points_count;
                                              END;
                                        END LOOP;
                                  END;
                            END IF;
                        END;
                  END IF;
                  oldLine:=pomLine;
            END;
      END LOOP;
      newLine := makeline_garray(editLine);
      RETURN newLine;
END;
$BODY$
  LANGUAGE 'plpgsql' VOLATILE
  COST 100;
```

```
CREATE OR REPLACE FUNCTION line_part(cesta GEOMETRY, prvniBod GEOMETRY,
druhyBod GEOMETRY)
  RETURNS GEOMETRY AS
$BODY$
DECLARE castCesty GEOMETRY;
DECLARE oldLength float;
DECLARE newLength float;
```

```
DECLARE points_numberSubLine INTEGER;
DECLARE editLine GEOMETRY[];
BEGIN
      oldLength:=ST_line_locate_point(cesta,  prvniBod);
      newLength:=ST_line_locate_point(cesta,  druhyBod);
      IF (oldLength > newLength) THEN
            BEGIN
                  SELECT ST_Line_Substring(cesta,newLength, oldLength) INTO
castCesty;
                  FOR points_numberSubLine IN REVERSE numpoints(castCesty)-
1..1 LOOP
                        BEGIN
                              editLine[numpoints(castCesty) -
points_numberSubLine]:= pointn(castCesty, points_numberSubLine);
                        END;
                  END LOOP;
            END;
      ELSE
            BEGIN
                  IF (oldLength = newLength) THEN
                        BEGIN
                              RETURN null;
                        END;
                  ELSE
                        BEGIN
                              SELECT ST_Line_Substring(cesta, oldLength,
newLength) INTO castCesty;
                              FOR points_numberSubLine IN
2..numpoints(castCesty) LOOP
                                    BEGIN
                                          --raise notice 'PRIDAVEK';
                                          editLine[points_numberSubLine]:=
pointn(castCesty, points_numberSubLine);
                                    END;
                              END LOOP;
                        END;
                  END IF;
            END;
      END IF;
      RETURN makeline_garray(editLine);
END;
$BODY$
  LANGUAGE 'plpgsql' VOLATILE
  COST 100;
```

```
CREATE OR REPLACE FUNCTION subroute(prvniBod GEOMETRY, druhyBod GEOMETRY,
prvniCesta GEOMETRY, druhaCesta GEOMETRY)
  RETURNS GEOMETRY[] AS
$BODY$
DECLARE cestaPrvniReturn GEOMETRY;
DECLARE cestaDruhaReturn GEOMETRY;
DECLARE cestaPrvniReturnNum INTEGER;
DECLARE cestaDruhaReturnNum INTEGER;
DECLARE lineArrayPrvni GEOMETRY;
DECLARE lineArrayDruhy GEOMETRY;
DECLARE returnArray GEOMETRY;
DECLARE returnNum INTEGER;
BEGIN
```

```
      SELECT * INTO cestaPrvniReturn, cestaDruhaReturn,
cestaPrvniReturnNum, cestaDruhaReturnNum FROM point_in_route(prvniCesta,
druhaCesta) f(a GEOMETRY, b GEOMETRY,c INTEGER, d INTEGER);



      lineArrayPrvni := line_part(prvniCesta, prvniBod ,cestaPrvniReturn);
      lineArrayDruhy := line_part(druhaCesta, druhyBod ,cestaDruhaReturn);


      FOR returnNum IN 1..numpoints(lineArrayPrvni) LOOP
            BEGIN
                  returnArray[returnNum] := pointn(lineArrayPrvni,
returnNum);
            END;
      END LOOP;


      FOR returnNum IN 1..numpoints(lineArrayDruhy) LOOP
            BEGIN
                  returnArray[returnNum + returnOld] :=
pointn(lineArrayDruhy, returnNum);
            END;
      END LOOP;
      RETURN returnArray;
END;
$BODY$
  LANGUAGE 'plpgsql' VOLATILE
  COST 100;
```

```
CREATE OR REPLACE FUNCTION point_in_route(prvniCesta GEOMETRY, druhaCesta
GEOMETRY)
  RETURNS SETOF RECORD  AS
$BODY$
DECLARE cestaPrvniInk INTEGER;
DECLARE cestaDruhaInk INTEGER;
DECLARE bodPrvniReturn GEOMETRY;
DECLARE bodDruhaReturn GEOMETRY;
DECLARE cestaPrvniReturn GEOMETRY;
DECLARE cestaDruhaReturn GEOMETRY;
DECLARE cestaPrvniReturnNum INTEGER;
DECLARE cestaDruhaReturnNum INTEGER;
DECLARE minDistance double precision;
DECLARE pomDistance double precision;
BEGIN
      minDistance := 20000000;
      FOR cestaPrvniInk IN 1..numpoints(prvniCesta) LOOP
            BEGIN
                  FOR cestaDruhaInk IN 1..numpoints(druhaCesta) LOOP
                        BEGIN


                              bodPrvniReturn := pointn(prvniCesta,
cestaPrvniInk);
                              bodDruhaReturn := pointn(druhaCesta,
cestaDruhaInk);


                              SELECT distance(bodPrvniReturn,
```

```
bodDruhaReturn)INTO pomDistance;


                              IF (minDistance > pomDistance) THEN
                                    BEGIN
                                          cestaPrvniReturn :=
bodPrvniReturn;
                                          cestaDruhaReturn :=
bodDruhaReturn;
                                          cestaPrvniReturnNum :=
cestaPrvniInk;
                                          cestaDruhaReturnNum :=
cestaDruhaInk;
                                          minDistance := pomDistance;
                                    END;
                              END IF;
                        END;
                  END LOOP;
            END;
      END LOOP;
      RETURN QUERY SELECT cestaPrvniReturn, cestaDruhaReturn,
cestaPrvniReturnNum, cestaDruhaReturnNum;
END;
$BODY$
  LANGUAGE 'plpgsql' VOLATILE
  COST 100;
```

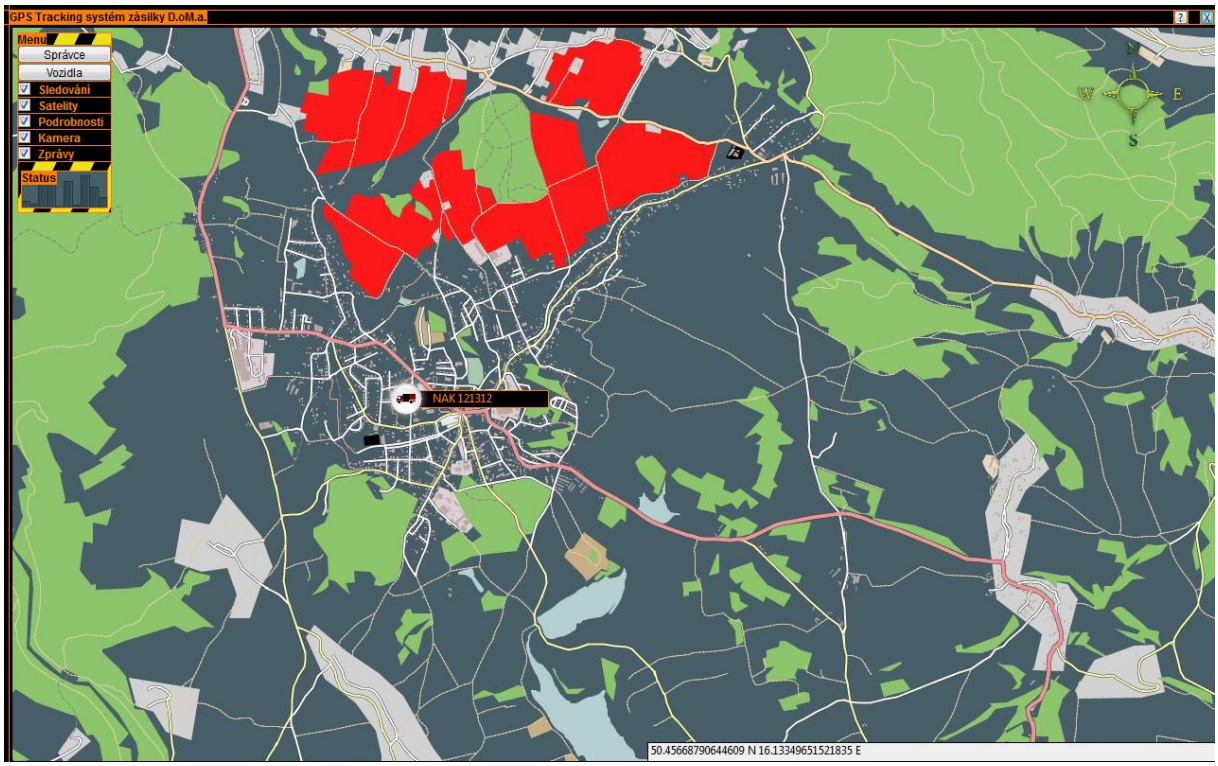
10.3 Nápověda aplikace osm2pgsql

```
Usage:
        osm2pgsql [options] planet.osm
        osm2pgsql [options] planet.osm.{gz,bz2}
        osm2pgsql [options] file1.osm file2.osm file3.osm
This will import the data from the OSM file(s) into a PostgreSQL diabase
suitable for use by the Mapnik renderer
Options:
   -a|--append          Add the OSM file into the database without removing
                        existing data.
   -b|--bbox            Apply a bounding box filter on the imported data
                        Must be specified as: minlon,minlat,maxlon,maxlat
                        e.g. --bbox -0.5,51.25,0.5,51.75
   -c|--create          Remove existing data from the database. This is the
                        default if --append is not specified.
   -d|--database        The name of the PostgreSQL database to Conner
                        to (default: gis).
   -l|--latlong         Store data in degrees of latitude & longitude.
   -m|--merc            Store data in proper spherical mercator (default)
   -M|--oldmerc         Store data in the legacy OSM mercator format
   -E|--proj num        Use projection EPSG:num
   -u|--utf8-sanitize   Repair bad UTF8 input data (present in planet
                        dumps prior to August 2007). Adds about 10%
overhead.
   -p|--prefix          Prefix for table names (default planet_osm)
   -s|--slim            Store temporary data in the database. This greatly
                        reduces the RAM usage but is much slower.
   -S|--style           Location of the style file. Defaults to
/usr/local/share/osm2pgsql/default.style
   -C|--cache           Only for slim mode: Use upto this many MB for
caching nodes
                        Default is 800
   -U|--username        Postgresql user name.
   -W|--password        Force password prompt.
   -H|--host            Database server hostname or socket location.
   -P|--port            Database server port.
   -e|--expire-tiles [min_zoom-]max_zoom        Create a tile expiry list.
   -o|--expire-output filename  Output filename for expired tiles list.
   -O|--output          Output backend.
                        pgsql - Output to a PostGIS database. (default)
                        gazetteer - Output to a PostGIS database suitable
for gazetteer
                        null  - No output. Useful for testing.
   -x|--extra-attributes
                        Include attributes for each object in the database.
                        This includes the username, userid, timestamp and
version.
                        Note: this option also requires additional entries
in your style file.
   -k|--hstore          Generate an additional hstore (key/value) column to
postgresql tables
   -G|--multi-geometry          Generate multi-geometry features in
postgresql tables.
   -h|--help            Help information.
   -v|--verbose         Verbose output.
Add -v to display supported projections.
Use -E to access any espg projections (usually in /usr/share/proj/epsg)
```

10.4 Uživatelský manuál k aplikaci operátora

V této příloze je popsán uživatelský manuál k aplikaci operátora. Manuál je pro větší názornost doplněn screen shoty.

Na obrázku 29 je zachycen vzhled aplikace, těsně po zapnutí a přihlášení uživatele. Jak je z obrázku patrné, je aplikace nastavená tak, aby největší plochu zabíraly mapové podklady. Operátor tak má přehled v rámci největšího možného kvadrantu.



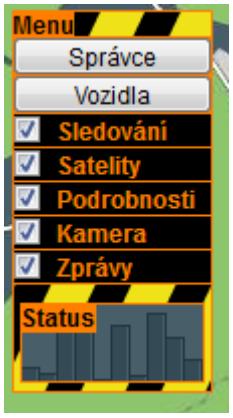
Obr. 29: Snímek aplikace operátora

10.4.1 Základní ovládací prvky

Aplikace obsahuje několik základních ovládacích prvků, které jsou zobrazeny přímo na ploše aplikace ihned po startu. Mezi tyto ovládací prvky patří: hlavní menu aplikace obr. 30, tlačítka pro nápovědu a ukončení obr. 31, a informační panel aplikace obr. 32.

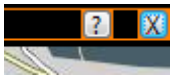
Hlavní menu aplikace obsahuje dvě tlačítka a pět zaškrtávacích polí. Tlačítky jsou: *Správce* a *Vozidla*.

Tlačítko *Správce* aktivuje zobrazení okno správce (viz. kapitola 10.4.2), která je určena pro administraci databázových údajů jako jsou adresáti, řidiči, vozidla a jiné. Tlačítko *Vozidla* aktivuje zobrazení tabulky se seznamem všech dostupných vozidel (viz. obr. 37). Zaškrtávací pole pod tlačítky slouží pro aktivaci účelových oken při výběru sledovaného vozidla (obr. 40).

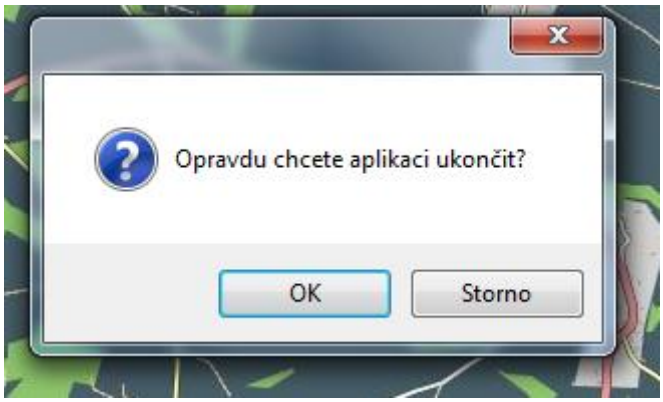


Obr. 30: Obrázek zobrazuje hlavní menu aplikace

Oblast tlačítek pro ukončení a nápovědu (obr. 31) se nachází v pravém horním rohu aplikace. Tlačítko *nápověda* aktivuje okno s nápovědou aplikace a tlačítko *ukončení* otevře dialog (obr. 32) s potvrzením souhlasu o uzavření aplikace. Po volbě *Ok* se aplikace operátora automaticky uzavře.

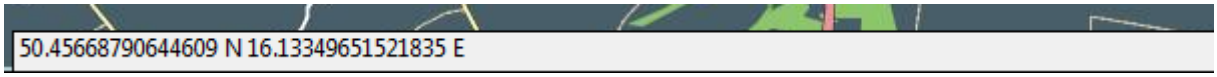


Obr. 31: Help a ukončovací tlačítko aplikace operátora



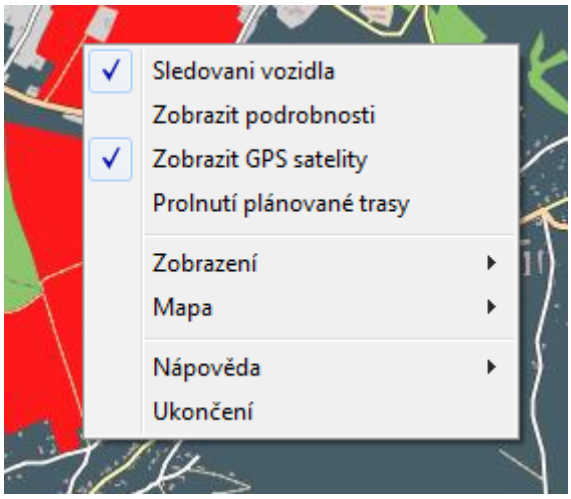
Obr. 32: Uzavírací dialog aplikace

Informační panel (obr. 33) aplikace informuje o různých událostech, které v aplikaci nastanou. Například uzavření či otevření nějakého okna. Jinak plní informační panel funkci zobrazování GPS polohy, na které se nachází kurzor myši.



Obr. 33: Informační panel aplikace

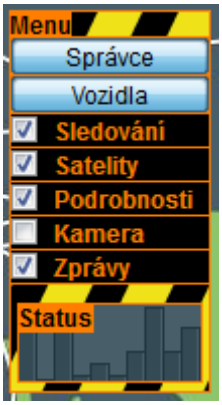
Mezi hlavní ovládací prvky je možné zařadit i gesta myši. Pomocí myši je možné posouvat mapový podklad, přibližovat jej, ale také i vybírat vozidla. Pokud je na mapovém podkladu zmáčknuto pravé tlačítko myši, aktivuje se ovládací nabídka, zobrazená na obrázku 34. Nabídka obsahuje podobné možnosti jako předchozí ovládací prvky. Nachází se zde ale i další možnosti jako např.: *Sledování vozidla*, nebo volby vztažené k mapovému podkladu.



Obr. 34: Vyskakovací menu s aktivovanou volbou *Sledování vozidla*

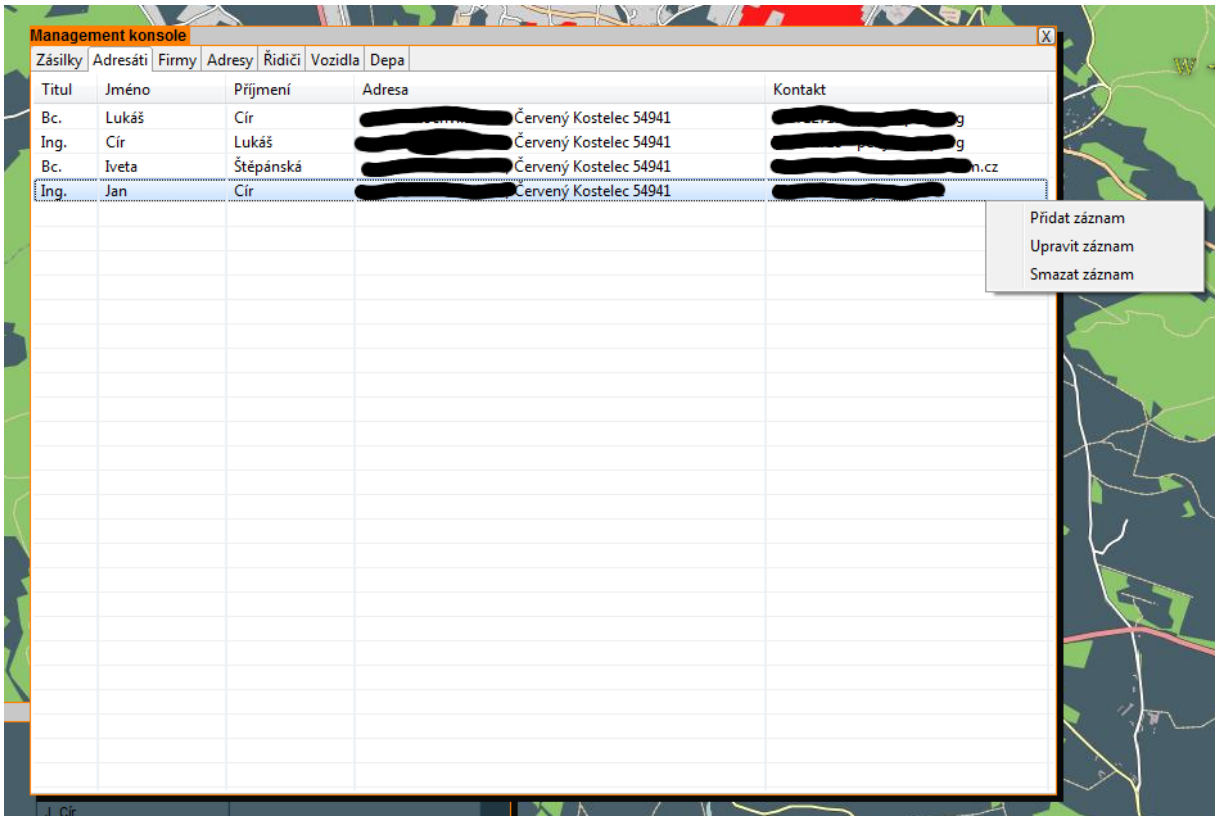
10.4.2 Správa uživatelských dat

Okno pro správu uživatelských dat, označené jako *Management konsole*, se zobrazí uprostřed aplikace. Po aktivaci tlačítka *Správce*, která se nachází v hlavním menu (obr. 35).



Obr. 35: Menu s otevřeným oknem pro správu uživatelských dat

Management konsole (obr. 36) obsahuje několik záložek. Každá záložka obsahuje specifickou tabulku. Tabulky. V řádcích tabulek se zobrazují uživatelská data jako jsou adresy, adresáti, řidiči a další.



Obr. 36: Okno pro správu uživatelských dat

Práce s tabulkou je jednoduchá: levé tlačítko myši slouží pro výběr řádku tabulky a pravé tlačítko k aktivaci vyskakovacího menu akcí:

Přidat záznam	Tato akce je aktivní v celé tabulce bez závislosti na vybraném řádku. Při aktivaci akce se objeví dialog pro přidávání nového záznamu. Po ukončení dialogu dojde k přidání záznamu do databázového systému i tabulky.
Upravit záznam	Tato akce je už závislá na vybraném řádku tabulky. Při aktivaci se otevře dialog stejný jako pro přidávání, ale už s před-vyplněnými údaji. Po ukončení dialogu dojde k změně stávajících údajů v databázovém systému i tabulce.
Smazat záznam	Poslední akce smaže záznam jak v databázovém systému, tak tabulce.

Práce s tabulkami ve všech záložkách jsou naprosto identické, jen struktury tabulek a dialogy přidávání/editování se od sebe navzájem liší.

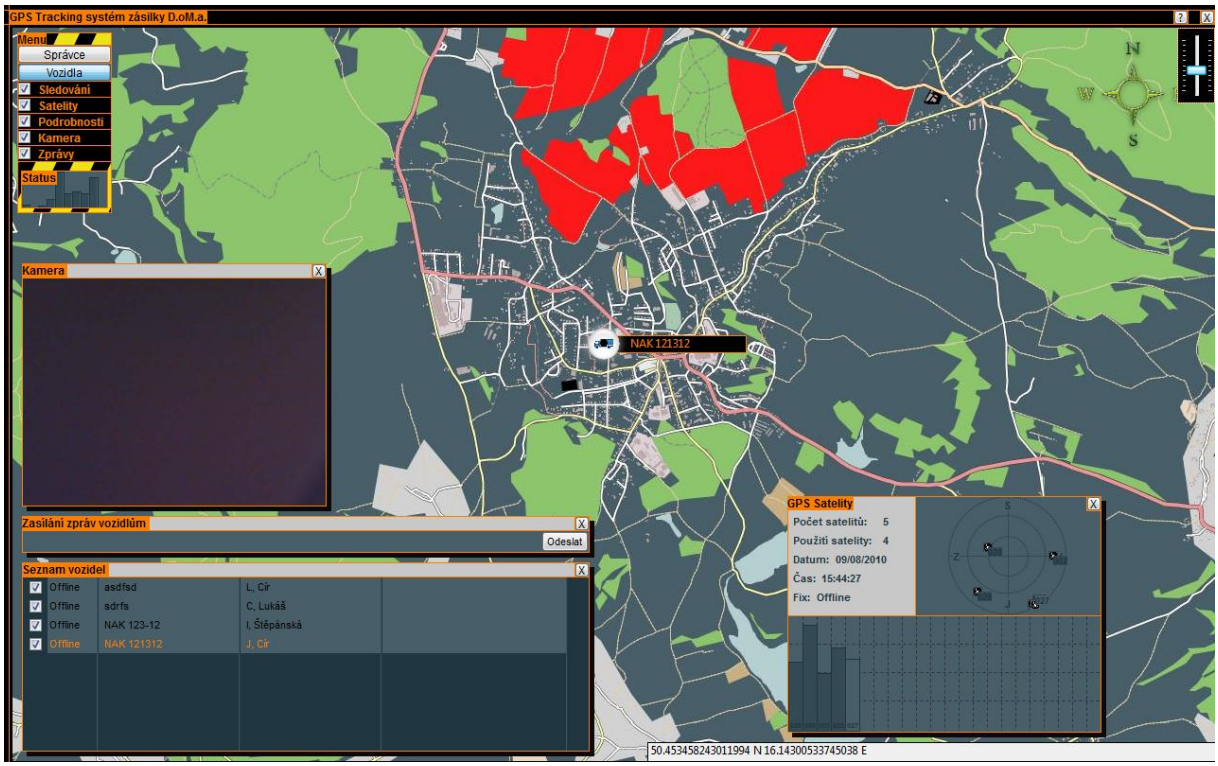
10.4.3 Práce s vozovým parkem

K tomu, aby bylo možné pracovat s vozovým parkem, je důležité vědět, kde se nachází seznam vozidel. *Seznam vozidel* je možné otevřít pomocí hlavního menu aplikace viz. obrázek 35. V tomto menu se musí aktivovat tlačítko *Vozidla*. Následně se otevře vlevo dole dialogové okno se seznamem vozidel (obr. 37).

Obr. 37: *Seznam vozidel* s popiskami tabulky

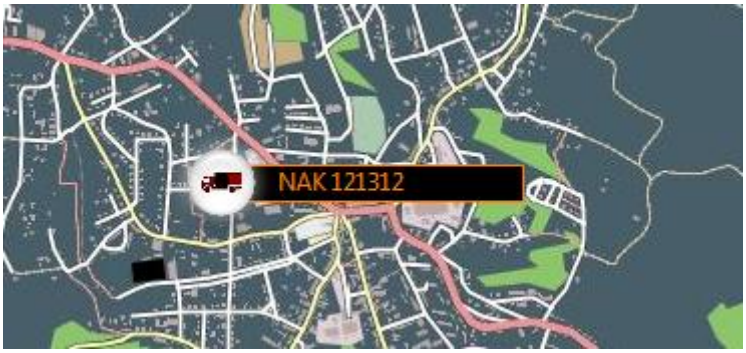
Seznamu na obrázku 37 informuje o vozidlech, která jsou k dispozici, o jejich řidičích a poblíž jakého města či vesnice se vozidlo v daném okamžiku nachází. Implicitně jsou popisky tabulky schované a aktivují se poklepem na horní rám okna.

Po kliknutí levým tlačítkem myši na jeden z řádků tabulky se vybere konkrétní vozidlo. Po vybrání vozidla se automaticky otevřou všechna okna, která jsou zaškrtnuta v hlavním menu viz. obrázek 38.



Obr. 38: Snímek aplikace s vybraným jedním z vozidel

Vybrat vozidlo je také možné přímo na mapovém podkladu kliknutím levého tlačítka myši nad ikonou vozidla. Jakmile je vozidlo vybráno, změní jeho ikona barvu. Příklad je uveden na obrázcích 39 a 40.

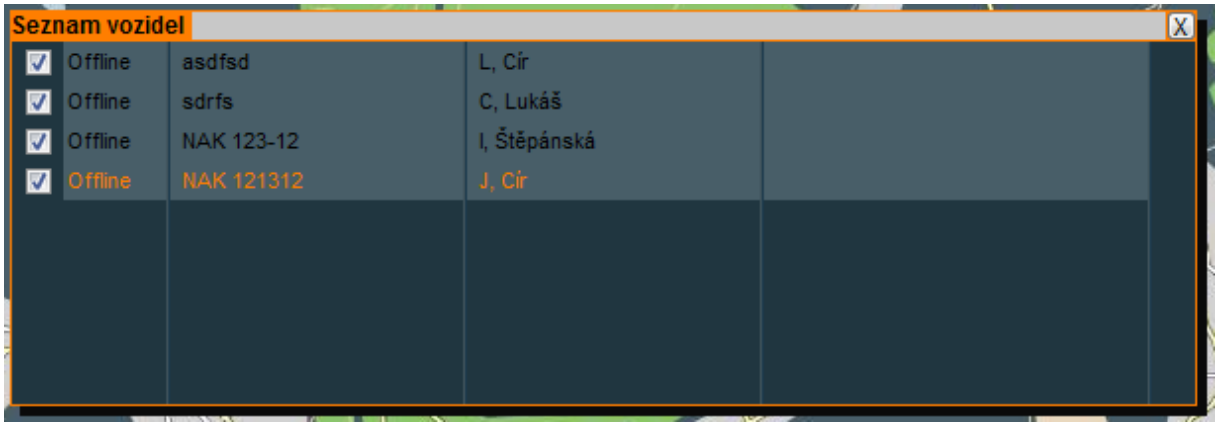


Obr. 39: Detail zabarvení nevybraného offline vozidla

Obr. 40: Detail zabarvení vybraného offline vozidla

Na obrázcích 39 a 40 je zobrazen příklad změny barvy *offline* vozidla. Pokud je vozidlo *online*, potom má nevybrané vozidlo barvu zelenou. Po jeho zvolení se barva změní na oranžovou.

Na obrázku 41 je vyobrazen *Seznam vozidel* s jedním aktivním (oranžovým) vozidlem. Pokud by se v seznamu nacházelo nějaké online vozidlo, je celý řádek tabulky zvýrazněn tmavší zelenou barvou. Seznam vozidel obsahuje zaškrtávací pole vlevo. Po zaškrtnutí, je vozidlo zobrazeno na mapovém podkladu. V opačném případě se vozidlo nezobrazí.



Obr. 41: Seznam vozidel s vybraným vozidlem

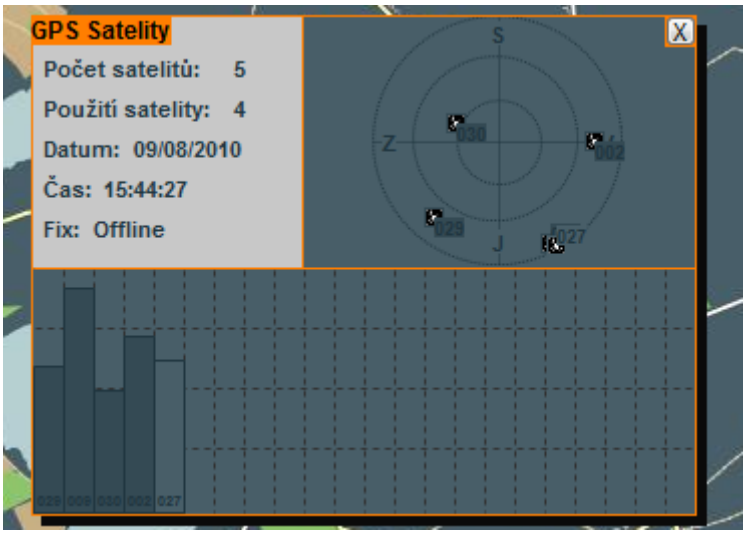
Po vybrání jednoho z vozidel se otevřou zvolená dialogová okna. Mezi tato okna patří dialog kamery, který v určitém časovém intervalu zobrazuje obrázky z kamery vozidla. Okno kamery je znázorněno na obrázku 42.

Obr. 42: Okno pro zobrazení obrazu z kamery vozidla

Pokud bude umět GPS klient přijímat zprávy, pak je zajímavou vlastností aplikace i okno na *Zasílání zpráv vozidlům* (obr. 43). Jedná se textovou řádku, do které se vyplní zpráva a stiskem tlačítka *Odeslat* se odešle GPS klientovi.

Obr. 43: Vstupní formulář pro odesílání zpráv GPS klientu

Posledním oknem, které se automaticky otevře, je dialogové okno *GPS satelitů* (obr. 44). V okně jsou zobrazeny podrobné informace o satelitech, jejich poloha i sloupcově síla jednotlivých signálů. Pokud operátor přejede myší nad sloupec signálu konkrétního satelitu, zvýrazní se poloha toho satelitu na mapce oblohy. Zároveň se vypíší detailní informace o tomto satelitu. Jsou zde i informace o čase sběru detailů o satelitech, nebo druh fixování satelitů. Pokud je fixování *offline*, znamená to, že poslední přijaté údaje z GPS klienta jsou starší než několik minut.

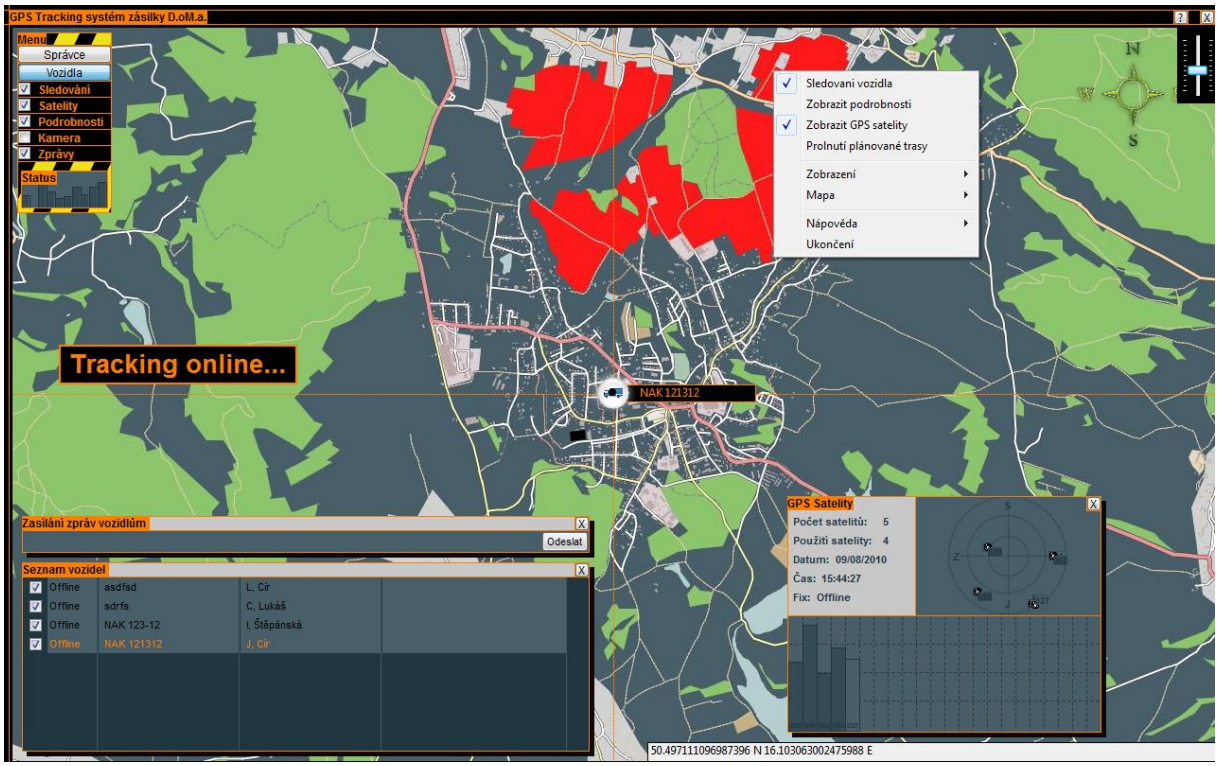


Obr. 44: Dialogové okno GPS satelitů

Na hlavním menu je ještě zašrtávací volba *Podrobnosti*. *Podrobnosti* jsou informace na mapovém podkladu objevující se vedle kurzoru myši v okolí ikony vozidla.

10.4.4 Sledování vozidla

Jakmile nastane potřeba sledovat jedno vozidlo v reálném čase, obsahuje aplikace možnost *Sledování vozidla*. Pro aktivaci sledování vozidla, jak je zobrazeno na obrázku 45, je potřeba označit příslušné vozidlo (viz. předchozí kapitola) a pomocí pravého tlačítka myši na mapovém podkladu zobrazit vyskakovací menu z obrázku 34. Po zaškrtnutí volby *Sledování vozidla* v tomto menu je *Sledování vozidla* aktivní.



Obr. 45: Aplikace s aktivovanou volbou sledování vozidel

Jakmile je volba *Sledování vozidla* aktivována, přizpůsobí se mapový podklad aplikace operátora tak, že se vycentruje podle polohy sledovaného vozidla.

Informaci o aktivním sledování dává velký nápis *Tracking online...*, viz. obrázek 46.

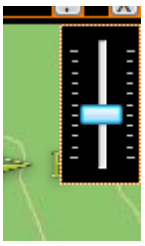
Obr. 46: Detail informace, znázorňující aktivní sledování vozidla

10.4.5 Práce s mapovým podkladem

Pro práci s mapovým podkladem slouží gesta myši. Posouvání mapového podkladu se docílí uchopením (stálý stisk) levého tlačítka myši a tažením na jakoukoliv stranu.

Přibližování mapového podkladu se zajistí dvojklik levým tlačítkem myši. Pro oddálení mapového podkladu je určen dvojklik myši prostředním tlačítkem.

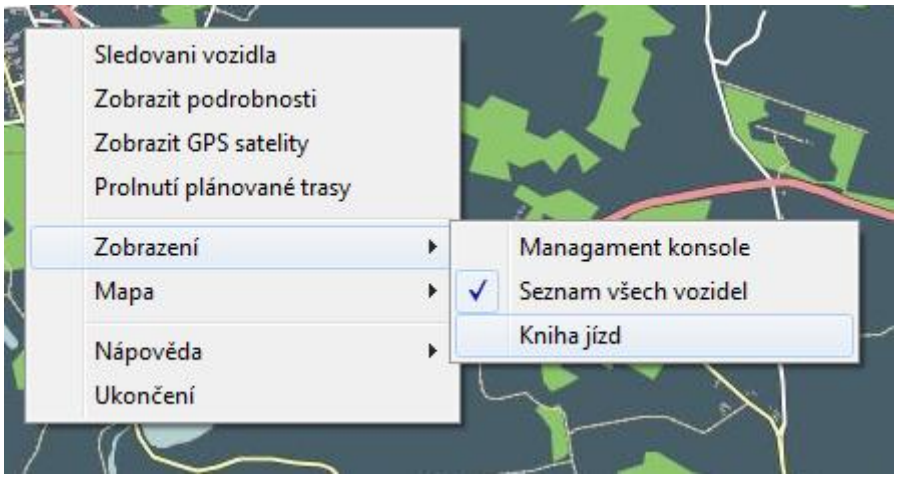
Pokud ovšem někomu nevyhovují gestikulace pomocí myši, je možné zobrazit na obrazovce posuvník přiblížení, který je zobrazen na obrázku 47. Tento posuvník je umístěn v pravém horním rohu. Zobrazení se provede pomocí vyskakovacího menu (obrázek 34) a v sekci *Mapa* se nachází zaškrtávací volba posuvníku.



Obr. 47: Posuvník pro ovládání přiblížení mapového podkladu.

10.4.6 Kniha jízd

Jednou z důležitých funkcí programu je *kniha jízd*. K otevření okna knihy jízd je potřeba ve vyskakovacím menu mapového podkladu (obr. 34) zvolit volbu *Zobrazení* a zaškrtnout možnost *Kniha jízd* (obr. 48). Tato volba je aktivní pouze v případě, kdy je zvoleno vozidlo, přesně podle kapitoly 10.4.3.



Obr. 48: Otevírání knihy jízd

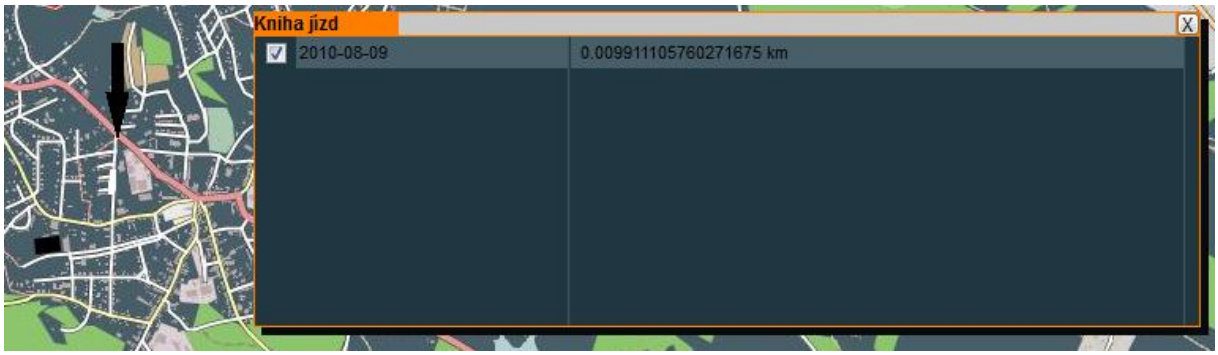
Kniha jízd (obr. 49) je tabulka zobrazující údaje o projetých trasách. Sloupce tabulky obsahují:

Datum	Zobrazuje den projetí trasy.
Délka	Sloupec zobrazuje délku trasy

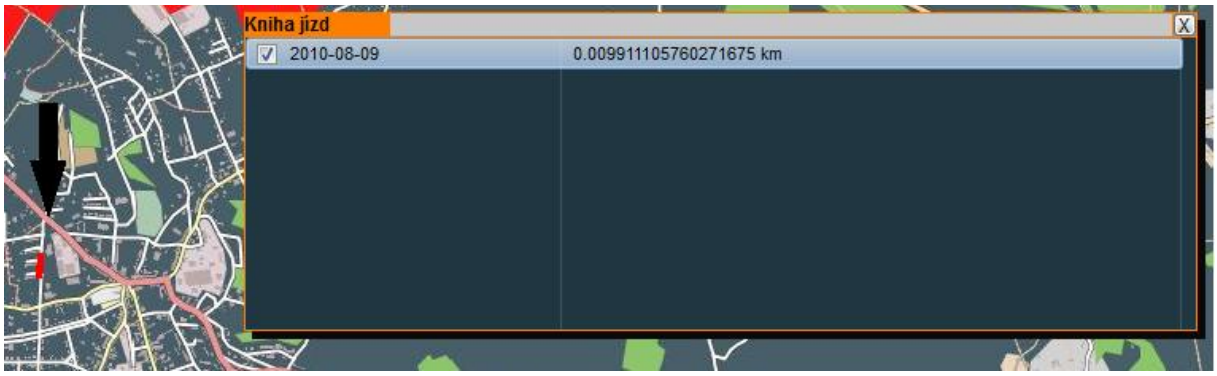
V tabulce tras jsou řádky podbarveny dvojí barvou. Světlejší zelená označuje soukromou cestu. Tuto cestu nelze nikterak vizualizovat. Tmavší zelená značí cestu služební. Ta umožňuje plnou vizualizaci. Kromě těchto dvou informací jsou v prvním sloupci znázorněny zaškrtávací boxy. Po jejich zaškrtnutí dojde k zobrazení projeté trasy přímo na mapovém podkladu.

Vizualizace tras je nastavena tak, že pokud se v tabulce knihy jízd označí poklepáním myši jakýkoliv řádek tabulky, bude tato trasa vykreslena na mapový podklad červenou barvou (obr. 50). Všechny ostatní zaškrtnuté řádky budou vykresleny barvou bílou.

Pokud budou zakšrtnuty některé trasy při uzavření *Knihy jízd*, pak ty budou i po uzavření *Knihy jízd* stále na mapovém podkladu zobrazeny. Pro odstranění projetých tras z mapového podkladu je potřeba u všech zrušit zaškrtnutí boxu.



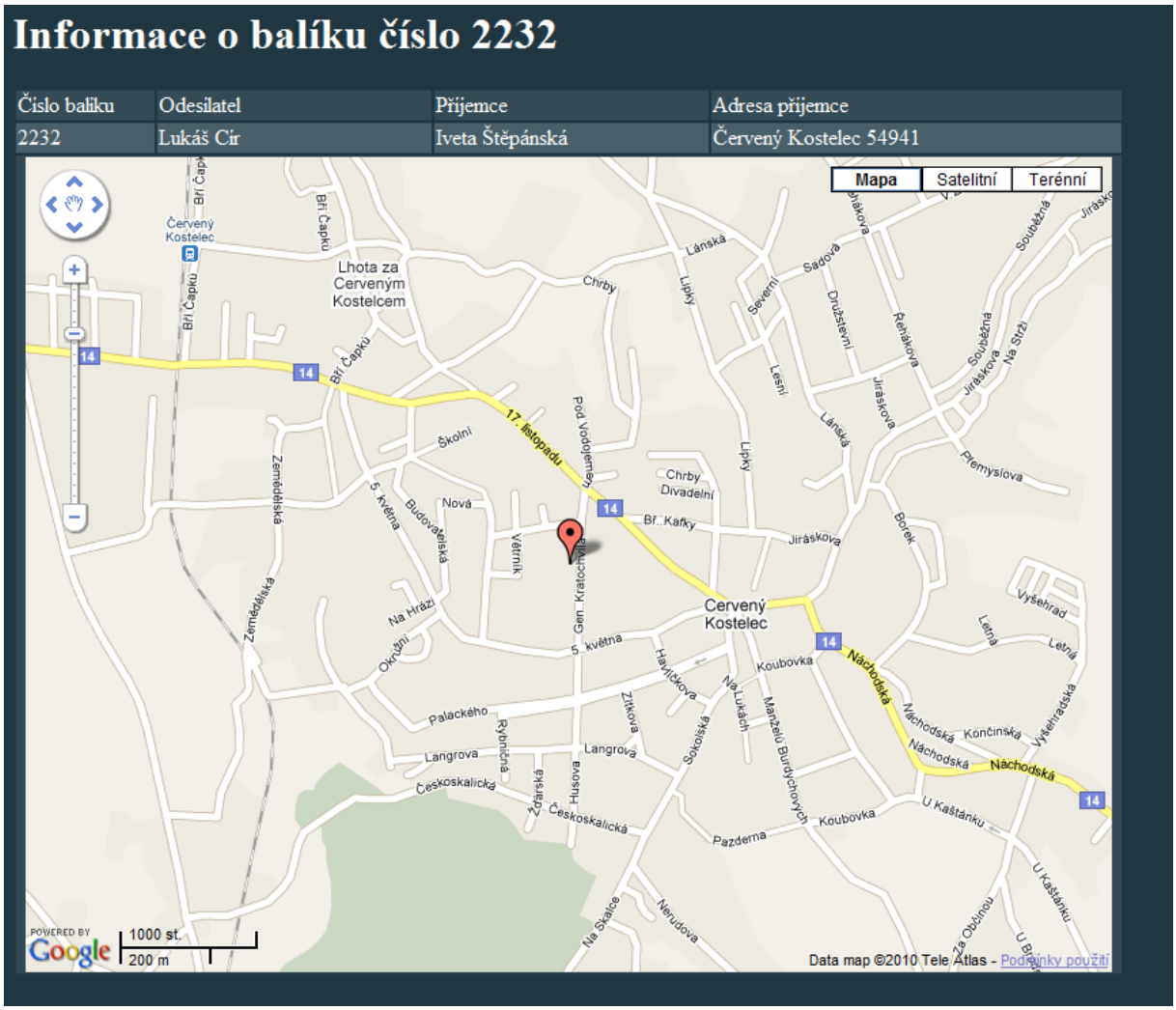
Obr. 49: Dialog Kniha jízd se zobrazenou projetou trasou



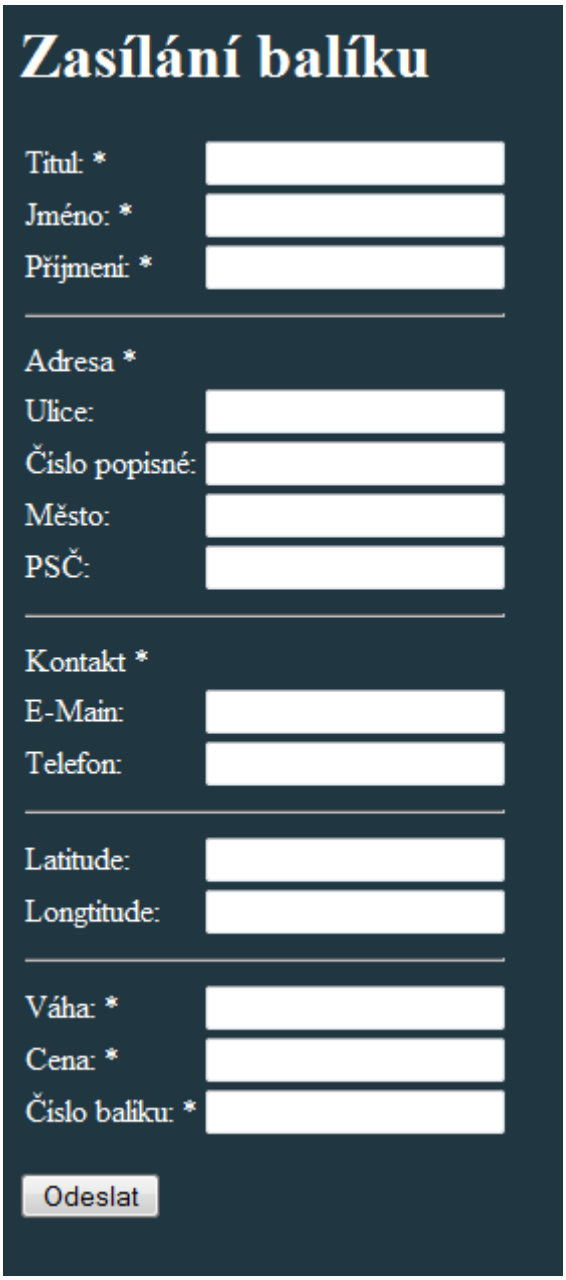
Obr. 50: Kniha jízd s označenou projetou trasou

10.5 Obrázky webových aplikací

10.5.1 Webová aplikace pro zákazníky

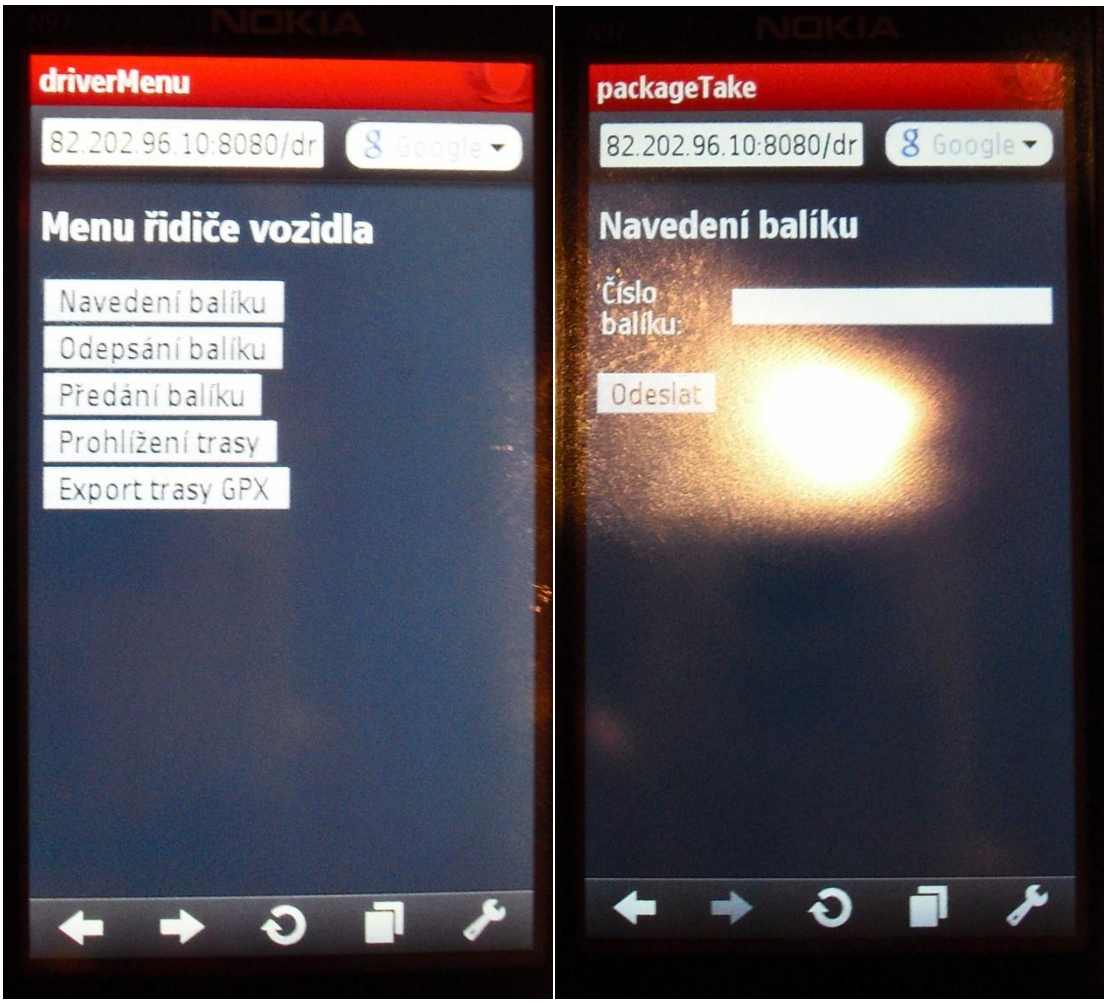


Obr. 51: Stránka příjemce balíku



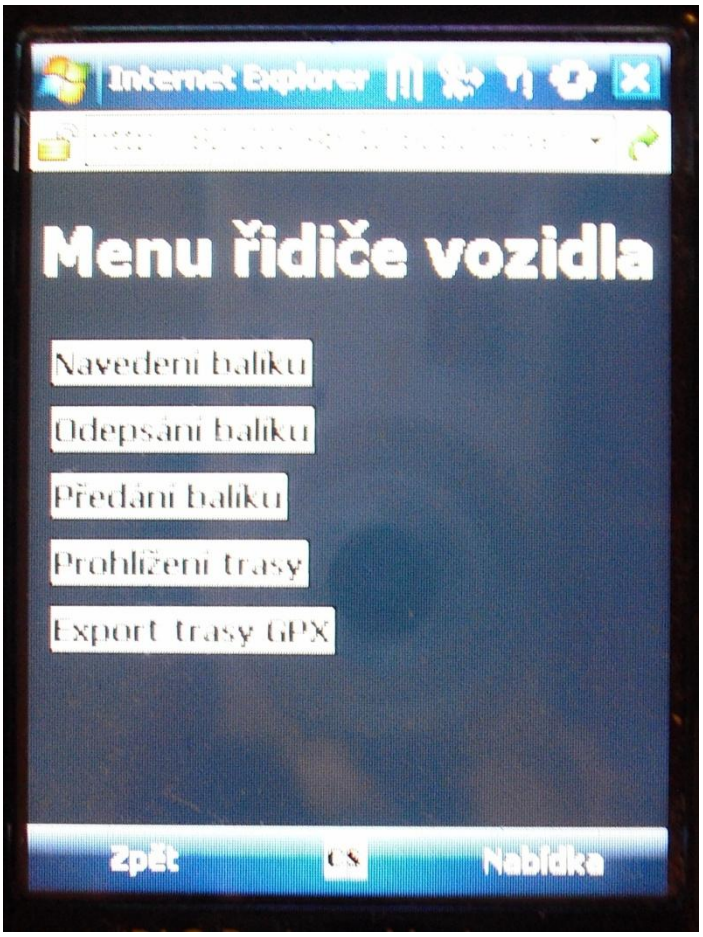
Obr. 52: Formulář pro registraci požadavku na odeslání balíku

10.5.2 Webová aplikace pro řidiče a operátory



Obr. 53 a 54: Hlavní menu řidiče a navedení balíku

(mobilní telefon s operačním systémem Symbian S60 páté generace)



Obr. 55: Menu řidiče v mobilním telefonu s operačním systémem Windows Mobile

10.6 Přepočet GPS souřadnic na obrazové souřadnice

Pro přepočet GPS souřadnic na obrazové souřadnice se musí nejdříve vypočítat konstanty *(latitudeConstant, longtitudeConstant)*. Počítají se z polohy dvou bodů a velikosti plochy mezi těmito dvěma body, které leží na úhlopříčce zmíněné plochy.

Dalším krokem je výpočet GPS souřadnic levého horního rohu *(xmin,ymin)* kreslící plochy (obrazová souřadnice 0,0). Nakonec se v závislosti na GPS souřadnicích levého horního rohu (0,0) vypočítají obrazové souřadnice zobrazovaného bodu, zadaného GPS souřadnicemi *(NovaGPSx, NovaGPSy)*.

```
public void CountConstant()
{
      double latitudeBottom = 50.398;
      double longtitudeBottom = 16.1881;
      double latitudeTop = 50.4991;
      double longtitudeTop = 15.9828;
      int imageHigth = (int)(1753 * zoomNasobek);
      int imageWidth = (int)(2267 * zoomNasobek);

this.latitudeConstant = (latitudeTop - latitudeBottom) / imageHigth;
this.longtitudeConstant = (longtitudeBottom - longtitudeTop) / imageWidth;
}


this.CountConstant();
double ymin = (this.ymax - ((canvasSize.y * this.latitudeConstant) ));
double xmin = (((canvasSize.x * this.longtitudeConstant) ) + this.xmax);


int xObraz = (int)((NovaGPSx - xmax)/ longtitudeConstant);
int yObraz = (int)((ymax - NovaGPSy)/ latitudeConstant);
```