Univerzita Pardubice
Dopravní fakulta Jana Pernera

Systém pro plánování a evidenci jízd s možností dynamického rozšíření

Bc. Tomáš Růt

Diplomová práce
2010

Univerzita Pardubice
Dopravní fakulta Jana Pernera
Akademický rok: **2009/2010**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Tomáš RŮT**
Studijní program:	**N3708 Dopravní inženýrství a spoje**
Studijní obor:	**Aplikovaná informatika v dopravě**
Název tématu:	**Systém pro plánování a evidenci jízd vozidel s možností dynamického rozšíření**
Zadávající katedra:	**Katedra informatiky v dopravě**

Zásady pro vypracování:

- Diplomová práce řeší plánování jízd vozidel a evidenci jejich průběhů za pomoci mobilních technologií a systému GPS.
- Systém se skládá ze dvou částí: desktopové a mobilní.
- Desktopová část bude mít následující funkce:
  a. plánování jízd vozidel jednotlivých řidičů s využitím mapového podkladu,
  b. uchování dat v databázi Microsoft SQL Server,
  c. sledování probíhajících jízd,
  d. kontrola uskutečněných jízd,
  e. generování výstupních sestav,
  f. rozšiřitelnost o zásuvné moduly v podobě knihoven.
- Mobilní část poskytne následující funkce:
  a. stahování naplánovaných cest ze vzdáleného serveru,
  b. zaznamenávaní průběhu cesty s využitím modulu GPS,
  c. aktualizace dat na vzdáleném serveru.
- Aplikace bude vyvinuta v programovacím jazyku C#.

Rozsah grafických prací:

Rozsah pracovní zprávy: **50 normostran**

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

Seznam odborné literatury:

1. **NAGEL Ch. et al.** *C# 2005 : Programujeme profesionálně*. **Brno : Computer Press, 2006. ISBN 80-251-1181-4.**

2. **TROELSEN, A.** *C# a .NET 2.0 profesionálně*. **Brno : Zoner Press, 2007. ISBN 80-86815-38-2.**

Vedoucí diplomové práce: **Ing. Karel Greiner, Ph.D.**
Katedra informatiky v dopravě

Datum zadání diplomové práce: **24. listopadu 2009**

Termín odevzdání diplomové práce: **24. května 2010**

prof. Ing. Bohumil Culek, CSc.
děkan

L.S.

doc. Ing. Josef Volek, CSc.
vedoucí katedry

V Pardubicích dne 20. listopadu 2009

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 9. 5. 2010

Bc. Tomáš Růt

SOUHRN

Předmětem diplomové práce je UML analýza a vývoj softwaru sloužícího pro plánování a evidenci jízd řidičů a vozidel a jejich online sledování za pomoci GPS a mobilních technologií. Software dále umožňuje dynamické rozšíření funkcionality pomocí knihoven.

Program je vytvořen ve vývojovém prostředí Visual Studio 2008 Professional za pomoci programovacího jazyka C#. Desktopová část aplikace je určena pro prostředí Windows XP a vyšší, mobilní část pro prostředí Windows Mobile 6.0 až 6.5.

Databáze je typu Microsoft SQL Server 2008. Aplikace je řešena jako online s přístupem pomocí uživatelského jména a hesla. Tyto údaje jsou uloženy v zašifrované podobě na databázovém serveru. Každý uživatel má umožněny pouze operace dle nastaveného oprávnění, které může měnit pouze administrátor.

Prostředí je vytvořeno jako vícejazyčné (angličtina, čeština), ovládané myší a klávesnicí. Aplikace dále umožňuje generovat výstupní sestavy dle výběru uživatele.

KLÍČOVÁ SLOVA

evidence; plánování; správa; sledování; GPS; generování; jízda; uživatel; vozidlo; mapa; souřadnice; ověření

TITLE

System for planning and registration trips with the possibility of dynamic extension

ABSTRACT

The subject of the thesis is UML analysis and software development serving the planning and registration trips of drivers such as vehicles and their online tracking by GPS and mobile technologies. The software also allows dynamic extensions of functionality through libraries.

The program is created in Visual Studio 2008 Professional using the programming language C #. Part of the desktop application is designed for Windows XP and later, part of mobile application for Windows Mobile 6.0 to 6.5.

The database type is Microsoft SQL Server 2008. The application is designed as online with access via user name and password. These data are stored in encrypted form on the database server. Each user is allowed only for operations set by permissions which can be changed only by administrator.

Environment is created as a multilingual (English, Czech), controlled by mouse and keyboard. The application also allows generating output reports according to user's choice.

KEYWORDS

registration; planning; management; monitoring; GPS; generating; user; vehicle; map; coordinates; validation**;**

OBSAH

Seznam obrázků

Seznam zkratek

SQL	Standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích. SQL je zkratka anglických slov Structured Query Language (strukturovaný dotazovací jazyk).
UML	Unified Modeling Language – unifikovaný modelovací jazyk. Univerzální modelovací jazyk pro vizuální modelování systémů. Má široké využití, nejčastěji je spojován s modelováním objektově orientovaných systémů [1].
.NET	Podle anglického dot NET = tečka NET (NET pochází ze slova network, tj. síť). Je zastřešující název pro soubor technologií v softwarových produktech, které tvoří celou platformu, která je dostupná nejen pro Web, Windows i Pocket PC. Common Language Infrastructure je standardizovaná specifikace jádra .NET [6].
MD5	Message-Digest algorithm je rozšířená rodina hashovacích funkcí, která vytváří ze vstupních dat výstup (otisk) fixní délky. Otisk je též označován jako miniatura, kontrolní součet (v zásadě nesprávné označení), fingerprint, hash (česky někdy psán i jako haš). Jeho hlavní vlastností je, že malá změna na vstupu vede k velké změně na výstupu, tj. k vytvoření zásadně odlišného otisku [5].
GPS	Global Positioning System, zkráceně GPS, je vojenský polohový družicový systém provozovaný Ministerstvem obrany Spojených států amerických, s jehož pomocí je možno určit polohu a přesný čas kdekoliv na Zemi nebo nad Zemí s přesností první desítky metrů. Přesnost GPS lze s použitím dalších metod ještě zvýšit až na jednotky centimetrů. Část služeb tohoto systému s omezenou přesností je volně k dispozici i civilním uživatelům [4].
GNSS	Globální družicový polohovací systém
RAPT	Evidence a plánování cest (*Registration and planning trips*). Název vyvíjeného systému.
RSS	RSS je rodina XML formátů určených pro čtení novinek na webových stránkách a obecněji indikaci obsahu. Technologie RSS umožňuje uživatelům Internetu přihlásit se k odběru novinek z webu, který nabízí RSS zdroj (*RSS feed*, též RSS kanál, *RSS channel*). Tento zdroj se většinou vyskytuje na stránkách, kde se obsah mění a přidává velmi [14].

XML	Extensible Markup Language (zkráceně XML, česky rozšiřitelný značkovací jazyk) je obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C. Je zjednodušenou podobou staršího jazyka SGML. Umožňuje snadné vytváření konkrétních značkovacích jazyků (tzv. aplikací) pro různé účely a různé typy dat. Používá se pro serializaci dat. Zpracování XML je podporováno řadou nástrojů a programovacích jazyků [15].
MVC	*Model View Controller* (Model pohled ovládání) odděluje část programu, mající na starosti předzpracování příkazů uživatele (tj. zjištění, co od programu vlastně chce) od částí zabezpečujících logiku programu, která uživatelovy příkazy zpracovává, a části, která má na starosti zobrazení výsledku. Jedná se spíše o architektonický vzor uplatňující se při celkovém návrhu architektury programu [8].
API	*Application Programming Interface* je rozhraní pro programování aplikací.
MDI	Aplikace využívající *Multiple Document Imaging* jsou takové, jejchž okna jsou zobrazována uvnitř rodičovského okna.

1 ÚVOD

K výběru tématu mne inspirovala praxe na úřadě, kde byly plánovány a evidovány cesty řidičů a vozidel ručním způsobem, bez možnosti jakékoliv kontroly průběhů těchto cest. Takovýto systém se dá velmi snadno obcházet a neumožňuje žádné dodatečné úkony.

Cílem mé práce je vytvořit systém, který bude umožňovat jednoduchou správu uživatelů a vozidel, pro něž bude možné jednoduše na základě mapového podkladu plánovat cesty a o těchto cestách generovat reporty. Dále tento systém bude umožňovat online zaznamenávat průběhy cest za pomocí mobilních zařízení vybavených systémem GPS a tím zprostředkovávat jejich případné sledování nebo pozdější kontrolu. Tento systém bude dále dynamicky rozšiřitelný o pluginy (zásuvné moduly), které budou ve formě *dll* knihoven automaticky načítány do programu.

Z výše uvedeného vyplývá, že systém bude obsahovat dvě části. Desktopovou, na které bude pracovat dispečer nebo administrátor, a mobilní, kterou bude využívat řidič. Úložiště dat musí být vytvořeno tak, aby bylo dostupné i pro mobilní zařízení.

Aplikace bude rozdělena na dvě části – desktopovou a mobilní. V desktopové části bude uživateli umožněno po přihlášení spravovat uživatele, vozidla, mapy, města a plánované cesty, případně bude mít zpřístupněny další funkcionality doplněné zásuvnými moduly.

Mobilní se bude nacházet na zařízení typu PDA nebo smartphone vybaveném systémem GPS. Uživatel bude schopen po přihlášení zjistit naplánované cesty a evidovat jejich průběh včetně zaznamenávání případně provedených údržeb.

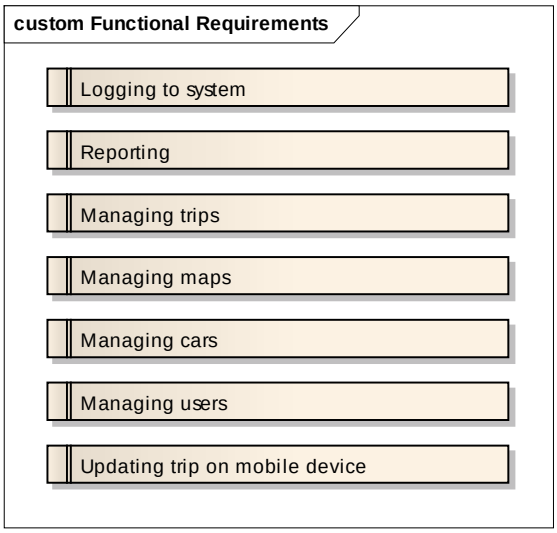
Pro úspěšný a efektivní vývoj bude zapotřebí zpracovat UML analýzu a dle této analýzy provést samotný vývoj.

2 Analýza

Analýza a návrh řešení jsou provedeny v jazyce UML (Unified Modeling Language) s využitím nástroje Enterprise Architect 7.5 od firmy SPARX. Spočívá ve zpracování funkčních a nefunkčních požadavků na systém a sestavení jednotlivých případu užití na základě těchto požadavků. Diagramy jsou vytvořeny v anglickém jazyce, další popis je již v jazyce českém.

2.1 Funkční požadavky

Tyto požadavky určují, co by měl vyvíjený systém dělat. Při zpracování funkčních požadavků je vycházeno ze zadání práce. Tyto se dají rozdělit na základní požadavky.



Obrázek 1 Základní funkční požadavky

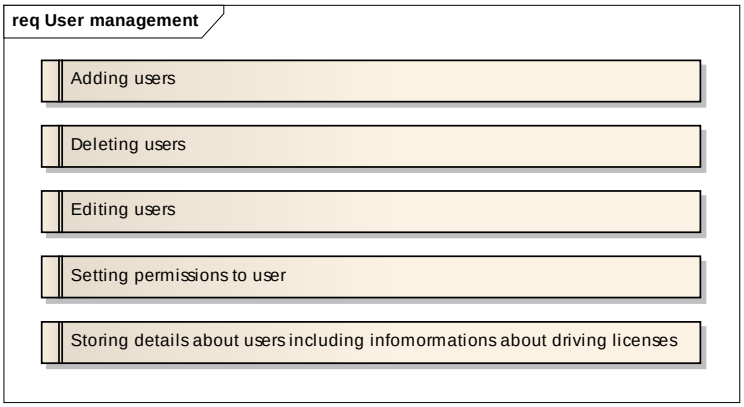
Požadavky jsou:

1. Přihlašování do systému
2. Reportování
3. Správa cest
4. Správa map
5. Správa vozidel
6. Správa uživatelů
7. Úprava informací o cestě na mobilním zařízení

Na základě těchto požadavků je práce rozdělena do jednotlivých modulů, ve kterých jsou požadavky definovány podrobně.

2.1.1 Modul správy uživatelů

Tento modul bude sloužit pro správu uživatelů, informací o nich a pro řízení jejich oprávnění.



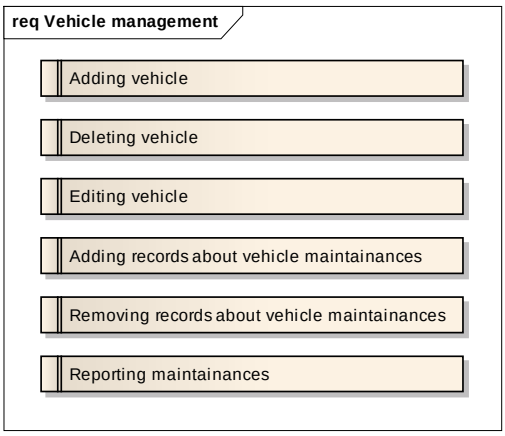
Obrázek 2 Funkční požadavky na modul správy uživatelů

Požadavky na modul správy uživatelů jsou:

1. Přidávaní uživatelů
2. Mazání uživatelů
3. Úprava uživatelů
4. Nastavování oprávnění jednotlivých uživatelů
5. Uchovávání informací o řidičských oprávněních uživatele

2.1.2 Modul správy vozidel

Pro modul určený pro správu vozidel jsou požadavky rozšířeny o uchovávání informací o kategoriích vozidel, díky kterým bude možné do budoucna řídit technické kontroly vozidel.



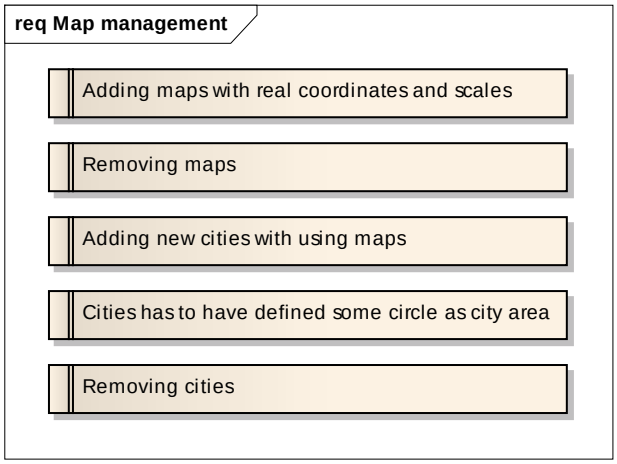
Obrázek 3 Funkční požadavky na modul správy vozidel

Požadavky jsou:

1. Přidávání vozidel
2. Odebírání vozidel
3. Úprava vozidel
4. Přidávání záznamů o údržbě vozidel
5. Odebírání záznamů o údržbě vozidel
6. Reportování záznamů o údržbě

2.1.3 Modul správy map

Modul bude sloužit pro nahrávání map do systému, a to včetně geografických informací. Na základě těchto map pak budou přidávány jednotlivé body obcí, a to za pomoci grafického nástroje.



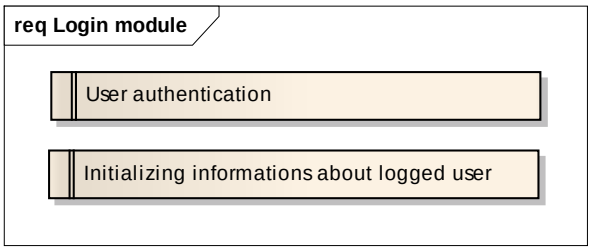
Obrázek 4 Funkční požadavky na modul správy map

Požadavky na modul správy map jsou:

1. Přidávání map s reálnými souřadnicemi a rozsahy
2. Odebírání map
3. Přidávání obcí s použitím map
4. Obce musí mít definován okruh jako plochu města
5. Odebírání obcí

2.1.4 Modul přihlášení

Pomocí tohoto modulu se bude moci uživatel přihlásit do systému.



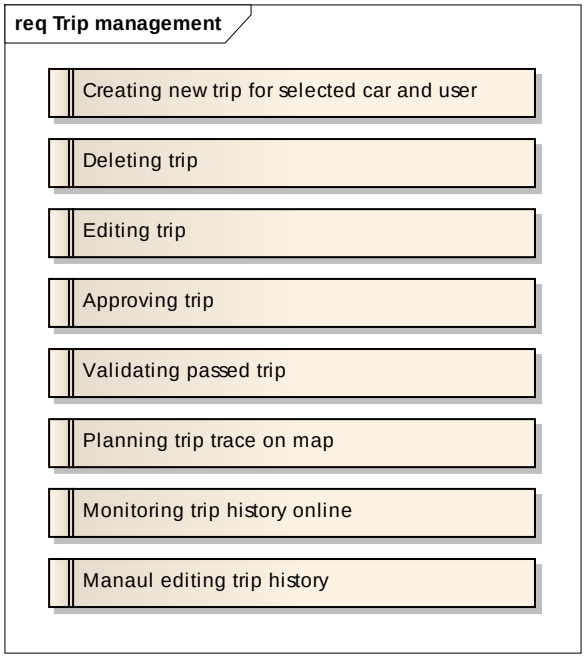
Obrázek 5 Funkční požadavky na modul přihlášení

Požadavky na tento modul jsou:

1. Autentizace uživatele
2. Inicializace informací o přihlášeném uživateli

2.1.5 Modul správy cest

Jedná se o nejdůležitější modul, kde bude probíhat plánování, zaznamenávání a sledování cest. Tento modul bude umožňovat vytvářet plán cest nad mapovým podkladem. Dále bude modul schopný vytvářet reporty o cestách definovaných uživatelem.



Obrázek 6 Funkční požadavky na modul správy cest

Požadavky na modul správy cest jsou:

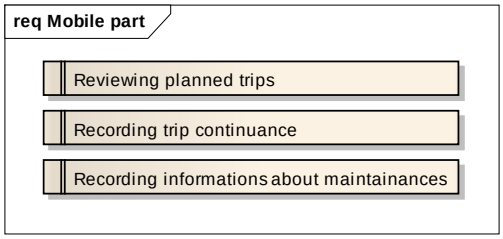
1. Vytváření cesty pro vybraného uživatele a vozidlo
2. Odebírání cesty
3. Úprava cesty
4. Schvalování cesty
5. Validace proběhlé cesty
6. Plánování trasy cesty na mapovém podkladu
7. Sledování průběhu cesty online
8. Ruční editace cesty

2.1.6 Modul přihlášení do mobilní části aplikace

Tento modul má stejné požadavky na funkcionalitu jako modul z kapitoly 2.1.4.

2.1.7 Mobilní část aplikace

Tato část aplikace bude spouštěna na mobilním zařízení vybaveném systémem GPS. Uživateli bude zobrazovat naplánované a schválené cesty a v případě vybrání některé této cesty k ní bude zaznamenávat informace o jejím průběhu a případně o provedených údržbách vozidla.



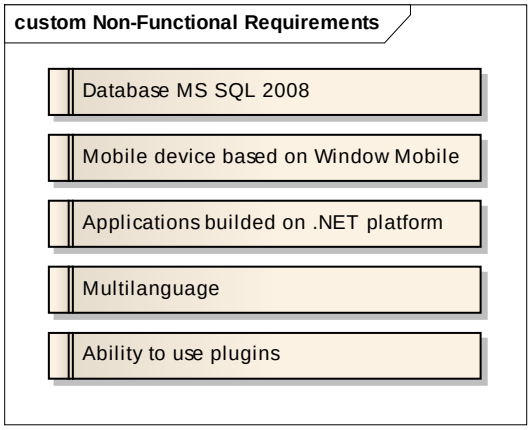
Obrázek 7 Funkční požadavky na mobilní část aplikace

Požadavky jsou:

1. Prohlížení naplánovaných cest
2. Zaznamenávání průběhu cest
3. Zaznamenávání informací o proběhlých údržbách

2.2 Nefunkční požadavky

Tyto požadavky definují omezení vytvářeného systému.



Obrázek 8 Nefunkční požadavky

Tyto požadavky jsou následující:

1. Databáze Microsoft SQL 2008
2. Mobilní zařízení bude se systémem Windows Mobile
3. Aplikace bude postavena pro platformu .NET
4. Vícejazyčnost
5. Schopnost používat zásuvné moduly

2.3 Případy užití

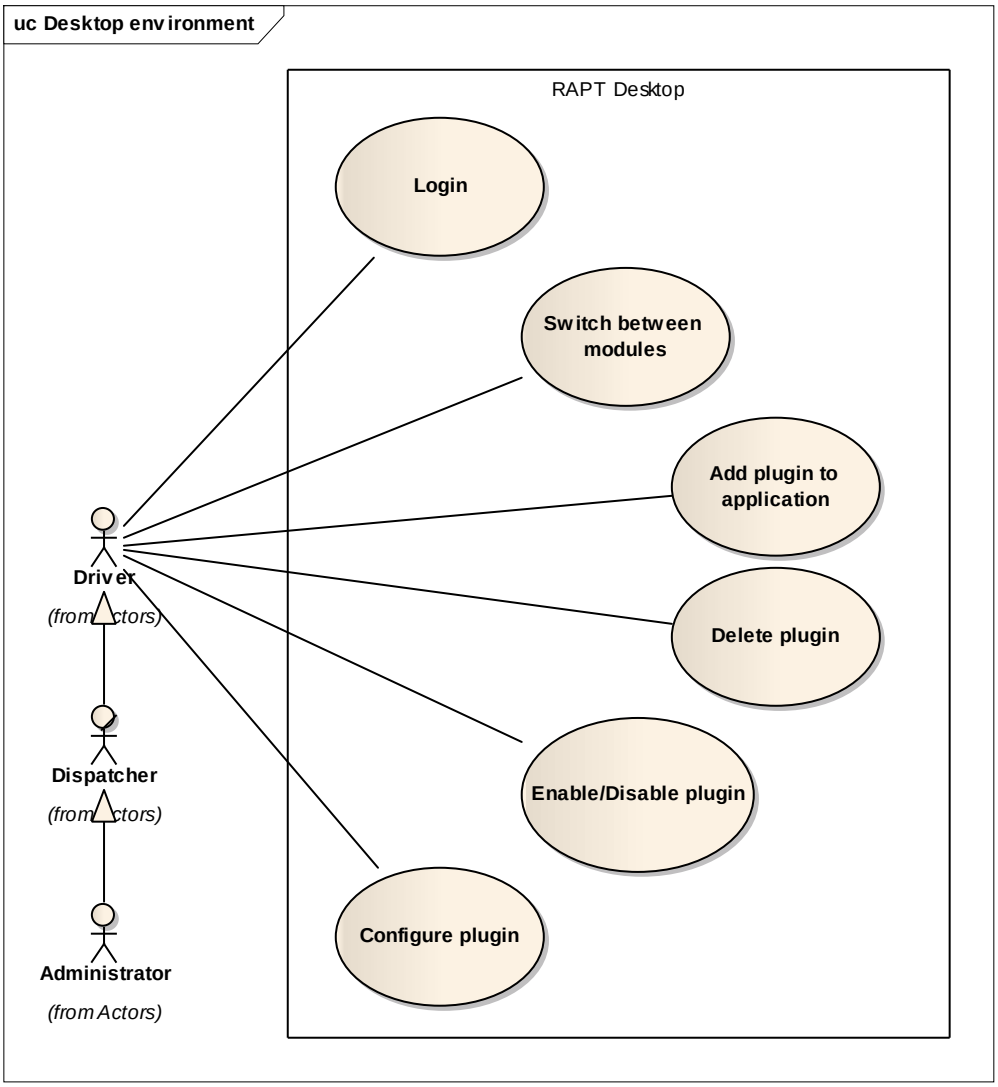
Modelování případů užití je jednou z forem inženýrství požadavků. Je jiným, doplňkovým způsobem získávání a dokumentování požadavků. Výstupem je model případu užití. Tento model obsahuje čtyři komponenty:

- Hranice systému *(system boundary)* – ohraničení zobrazené kolem případu užití, jež je vyznačením území nebo hranic modelovaného systému. Tomu se v UML 2 říká *subjekt*.
- Aktéři *(actors)* – jsou to role, přidělené osobám nebo předmětům používajícím daný systém.
- Případy užití *(use cases)* – činnosti, které mohou aktéři se systémem vykonávat.
- Relace *(relationships)* – smysluplné vztahy mezi aktéry a případy užití.

Modely případů užití navíc poskytují hlavní zdroj objektů a tříd. Jsou prvotním vstupem k modelování tříd.[1]

2.3.1 Digram desktopového prostředí

V tomto diagramu je namodelováno, co mohou aktéři dělat s prostředím desktopové části aplikace. Pro přístup do aplikace je nejprve nutné ověřit uživatele *přihlášením*. Po úspěšném přihlášení může uživatel *přepínat mezi jednotlivými moduly aplikace*. Dle požadavků má uživatel možnost rozšiřovat aplikaci o zásuvné moduly v podobě *dll* knihoven. Tyto moduly může *přidávat*, *odebírat*, *zapínat*, *vypínat* a *konfigurovat*.



Obrázek 9 Model případů užití desktopového prostředí

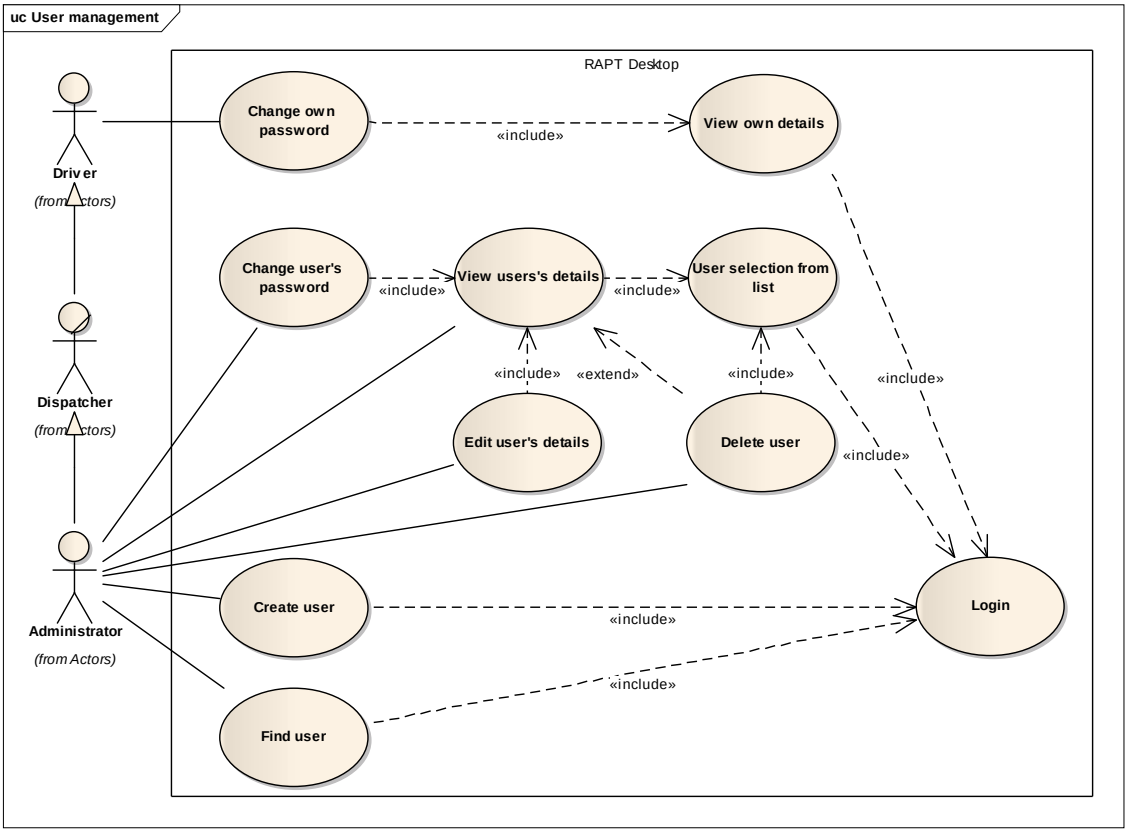
V diagramu je znázorněn vztah mezi jednotlivými aktéry, kdy od řidiče dědí vlastnosti dispečer, a od dispečera administrátor.

2.3.2 Diagram správy uživatelů

V tomto diagramu jsou podstatní uživatelé řidič a administrátor.

Řidič a dispečer nemají oprávnění prohlížet a spravovat uživatele. Mají však možnost *prohlížet vlastní informace* a *měnit své heslo*. K tomu je zapotřebí *přihlášení* do desktopového prostředí.

Administrátor má navíc možnost na *základě výběru uživatele ze seznamu prohlížet jeho detaily*, *upravovat jeho detaily*, *měnit heslo*, případně *uživatele smazat*. Dále může administrátor *vytvářet nové uživatele* a *vyhledávat existující uživatele*. Tyto možnosti jsou administrátorovi zpřístupněny po *přihlášení* do desktopového prostředí.



Obrázek 10 Model případů užití správy uživatelů

2.3.3 Diagram správy vozidel

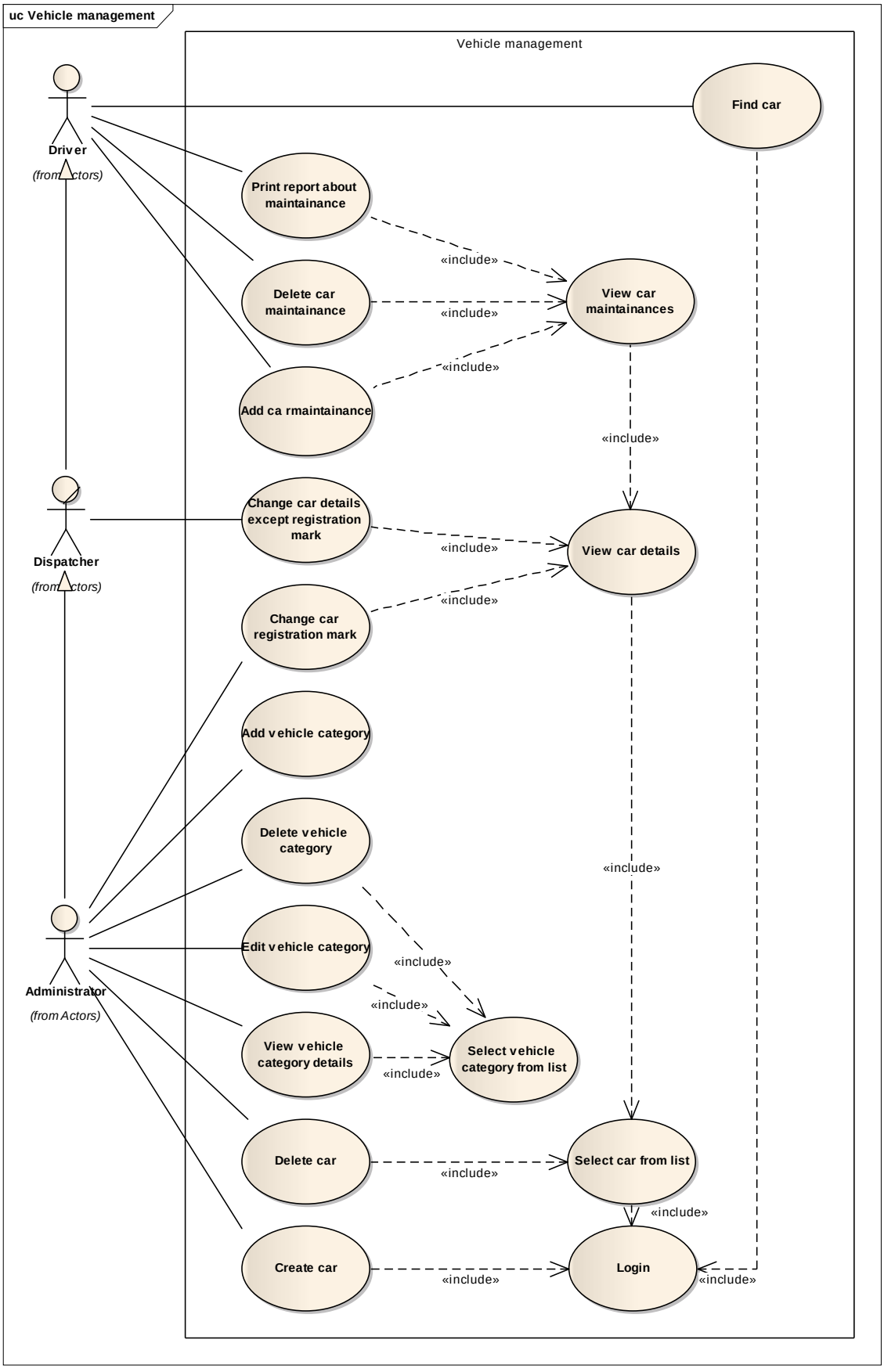
V tomto diagramu vystupují všichni tři aktéři, kteří se pro provádění veškerých činností musí nejprve *přihlásit do systému*.

Řidič má opět velmi omezené možnosti působení. Může u *vozidla vybraného ze seznamu prohlížet jeho informace*, *prohlížet údržby*, *odebírat* a *přidávat nové*. O těchto údržbách může *generovat reporty*. Dále může vozidla vyhledávat.

Dispečer má možnosti působnosti větší. Oproti řidiči může navíc *měnit údaje vybraného vozidla*, avšak kromě registrační značky.

Administrátor má navíc ještě možnost *změnit registrační značku vozidla*, *přidat nové vozidlo* a *smazat vybrané vozidlo*.

Vozidla mohou být různých kategorií. Tyto kategorie jsou definovatelné. Mohou být *přidávány*, *upravovány* a *mazány*. Tyto činnosti jsou přiděleny administrátorovi.



Obrázek 11 Model případů užití správy vozidel

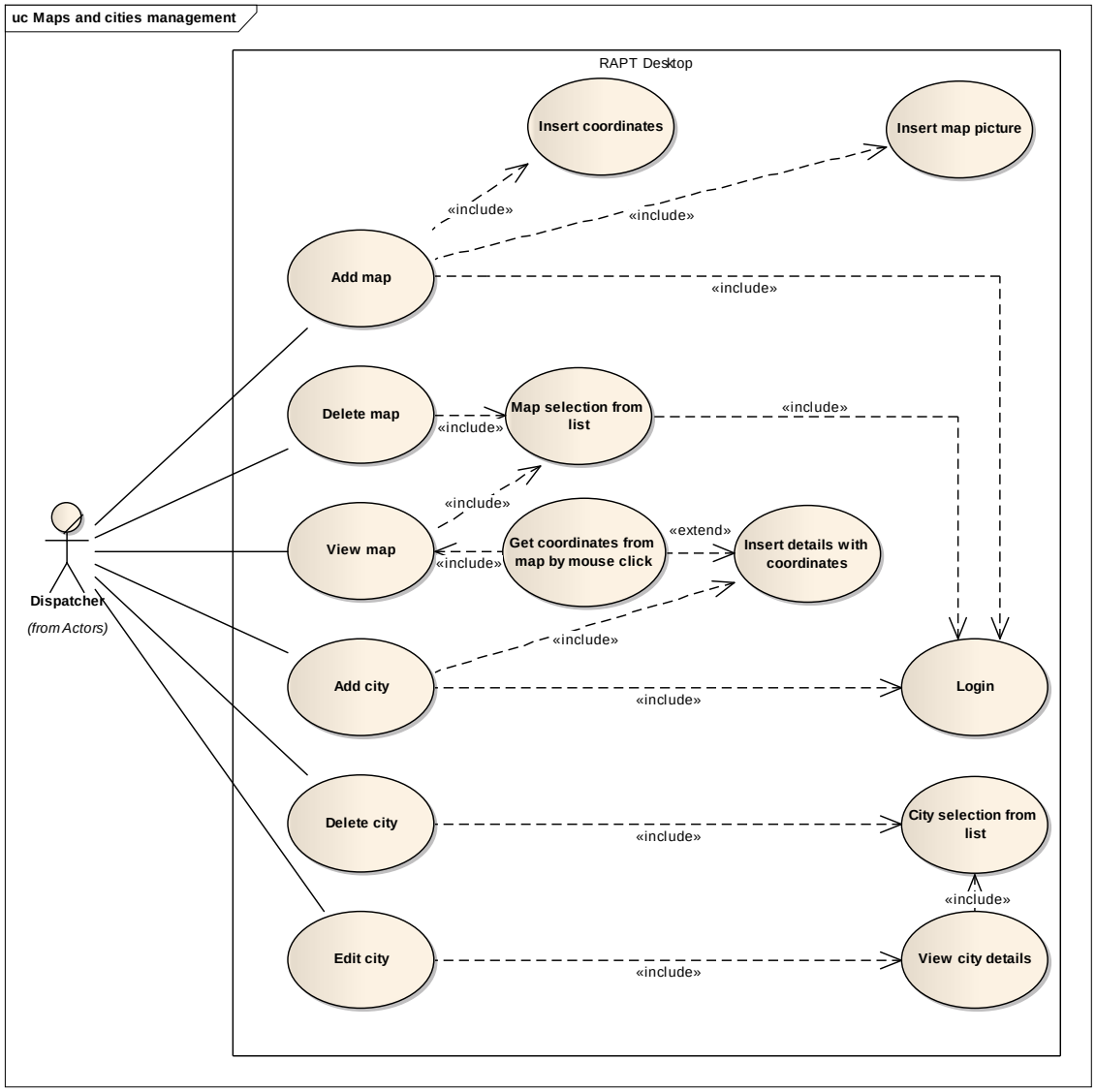
2.3.4 Diagram správy map a měst

Plánování cest by mělo být podpořeno možností používat mapové podklady. Následující diagram zobrazuje případy užití správy těchto map a pro ně definovaných měst.

V této části aplikace se vyskytuje aktér dispečer, od něhož dědí vlastnosti administrátor. Ke všem činnostem je opět nutné být přihlášen do systému.

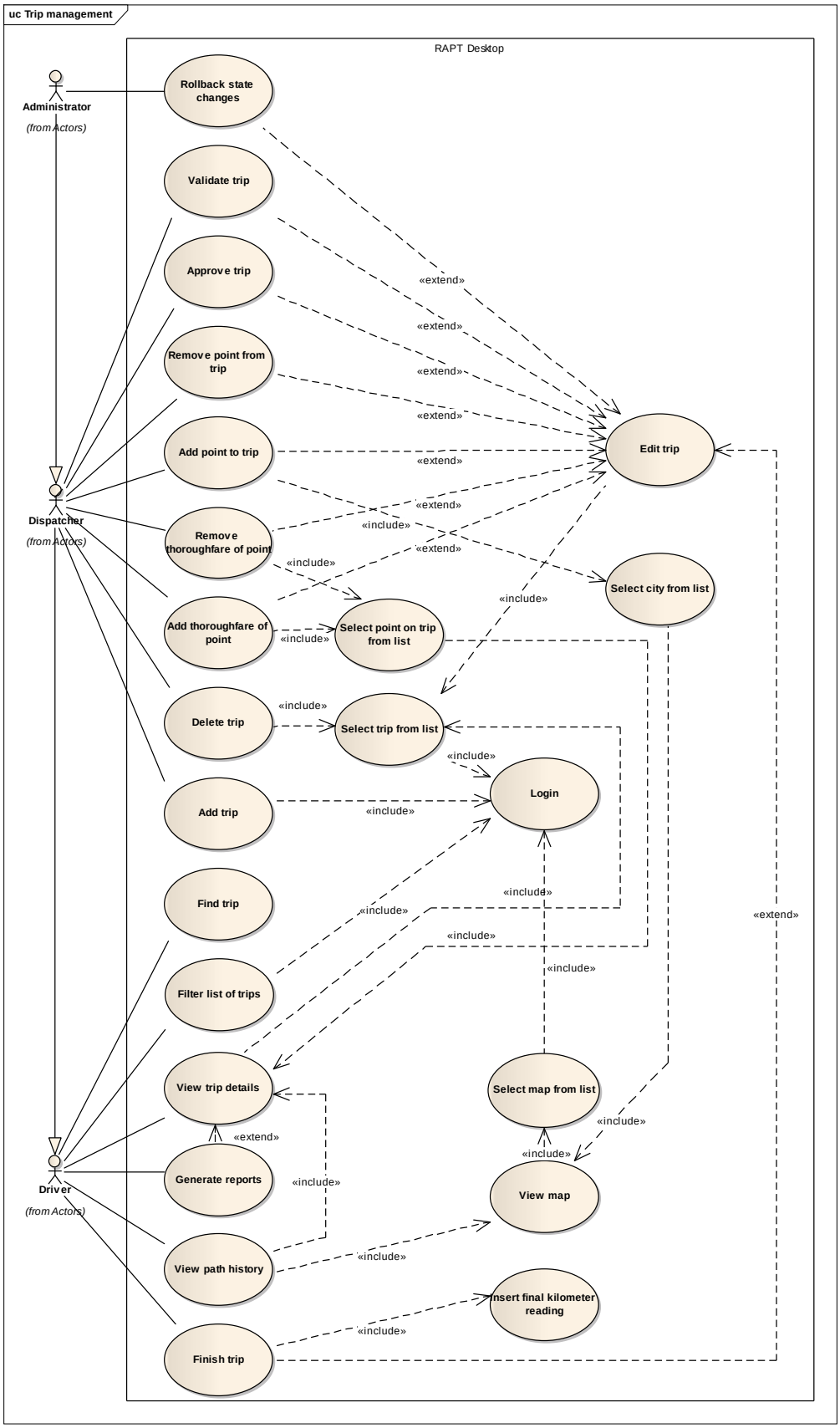
Dispečer může *přidávat mapy*. K tomu musí *vybrat obrázek mapy* a *zadat souřadnice*. Tyto mapy pak může dále *prohlížet*, nebo *smazat* po jejich *vybrání z rolovacího seznamu*.

Dále může *přidávat města zadáním jejich informací* a *souřadnic*. Tyto souřadnice může zadat buď ručně, nebo kliknutím na bod na načtené mapě. Dále může *města prohlížet*, *mazat* nebo *upravovat po jejich vybrání ze seznamu*.



Obrázek 12 Model případů užití správy map a měst

2.3.5 Diagram správy cest



Obrázek 13 Model případů užití správy cest

Tento digram znázorňuje případy užití nejdůležitější části aplikace – správy cest a jejich plánování. Jako u předchozích i u této správy je nutné se *přihlásit* do systému.

Řidič v této části aplikace může *zobrazovat cesty na základě předvoleného filtru*. Tento filtr umožňuje vybírat cesty na základě řidiče, vozidla, plánovaného data začátku a konce cesty a podle stavu, ve kterém se cesta nachází. Z tohoto výběru může dále *generovat report*. V cestách je možné *vyhledávat*.

V seznamu cest pak může *vybrat právě jednu požadovanou* a *prohlížet její detaily*. Kromě prohlížení detailů cesty si může na *mapě, kterou vybral ze seznamu map* a načetl, *prohlédnout i průběh cesty*. Na této mapě je možné vidět plánované body na cestě a skutečný průběh cesty. V případě, že cesta je schválená, ale ještě není validovaná a ukončená, může tuto cestu *ukončit* včetně *zadání konečného stavu kilometrů*. Tento stav se pak projeví v informacích o vozidle. K cestě dále může *generovat reporty*.

Dispečer je hlavní aktér plánování cesty. Po *výběru cesty ze seznamu* ji navíc může *smazat*. Hlavní činností dispečera je *vytváření cest*. Tuto cestu pak *upravuje* doplňováním dalších informací. Na cestě může *přidávat body vybráním ze seznamu bodů* k *vybrané mapě*, *odebírat* je a *zadávat nebo mazat průjezdy těmito body*. Po naplánování cesty může cestu *schválit* a tím zpřístupnit řidičovi. Po ukončení cesty může provést její *validaci*. Tím se cesta definitivně uzavře pro další změny.

Administrátor provádí všechny výše uvedené činnosti. Navíc má oprávnění *vracet přidělené stavy*. To znamená, že může vrátit cestu ze stavu „validovaná“ do stavu „ukončená“, ze stavu „ukončená“ do stavu „schválená“, případně i zrušit její schválení.

2.3.6 Diagram mobilní části

Na tomto diagramu jsou případy užití činností, které jsou dostupné řidičům po přihlášení do mobilní aplikace.

Řidič může i do této části systému *přidávat nebo odebírat zásuvné moduly* v podobě *dll* knihoven a dále je *zapínat, vypínat a konfigurovat*. Dále může řidič *zapnout nebo vypnout modul GPS*.

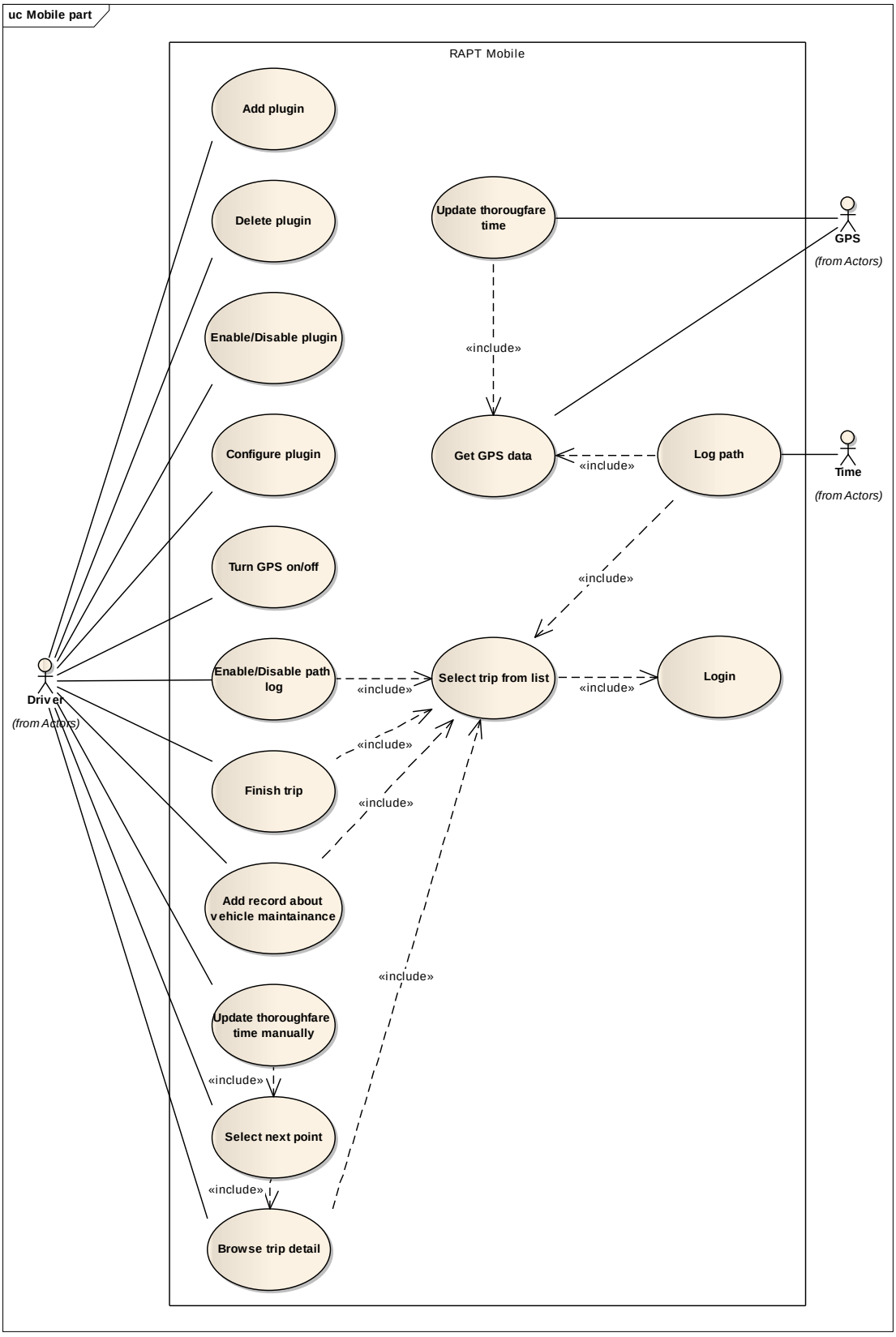
Po přihlášení má řidič možnost načíst jemu naplánované a schválené cesty pro dané datum. *Vybráním cesty* dojde k jejímu načtení a řidič může *prohlížet potřebné detaily*, zejména pak sekvenci naplánovaných bodů na trase. *V této sekvenci se může řidič posouvat* podle potřeby.

Aplikace pomocí *GPS zaznamenává časy průjezdů* plánovanými body. Pokud není GPS informace dostupná, může řidič *zaznamenat čas ručně*.

Řidič má dále možnost *zapínat nebo vypínat zaznamenávání průběhu cesty*. V případě zapnutého zaznamenávání jsou *automaticky* v průběhu cesty *zaznamenávány zeměpisné souřadnice* podle modulu GPS. Tento záznam je možné poté prohlížet v desktopové části systému ve správě cest.

V průběhu cesty může řidič *uložit záznam o proběhlé údržbě*.

Na konci cesty pak může řidič *zaznamenat ukončení cesty* včetně *zadání konečného stavu kilometrů* na tachometru. Tento stav se projeví v detailech vozidla, pro které byla cesta naplánována.



Obrázek 14 Model případů užití mobilní aplikace

3 Návrh řešení

I pro návrh řešení je využit nástroj Enterprise Architect verze 7.5. Návrh se skládá z modelu tříd a modelu databázové struktury. Systém je navržen jako modulární z důvodu přehlednosti při práci a pro možnost pozdější implementace automatických aktualizací programu. Této modulárnosti bude odpovídat i zobrazení modelů.

Vzhledem k tomu, že aplikace se nachází na zařízeních s různými operačními systémy (Microsoft Windows XP a vyšší a Microsoft Windows Mobile 6.0 a vyšší), je návrh pro dobrou přenositelnost kódu vytvořen pro platformu .NET 3.5.

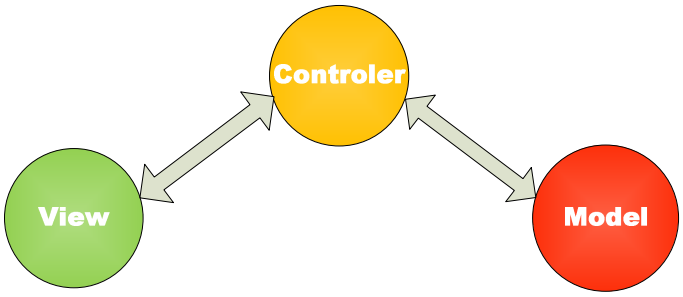
3.1 Návrhové vzory

V návrhu je využito známých návrhových vzorů *Model View Controller* (Model pohled ovládání) pro další rozčlenění aplikace, *Singleton* (Jedináček) pro databázové připojení a *Observer* (Pozorovatel) pro šíření informací od zařízení GPS.

3.1.1 Model View Controller

Model View Controller (Model pohled ovládání) [9] odděluje část programu, mající na starosti předzpracování příkazů uživatele (tj. zjištění, co od programu vlastně chce) od částí zabezpečujících logiku programu, která uživatelovy příkazy zpracovává, a části, která má na starosti zobrazení výsledku. Jedná se spíše o architektonický vzor uplatňující se při celkovém návrhu architektury programu.

Tento vzor explicitně odděluje část programu zabývající se vstupem a předzpracováním zadávaných příkazů od části reakcí na zadané příkazy a části mající na starosti výstup výsledků. V čisté verzi má každou z těchto činností na starosti samostatný objekt, resp. skupina objektů (samostatný modul). Tímto oddělením se dá např. zabezpečit, aby aplikace reagovala jednotně, zadá-li uživatel svůj příkaz z klávesnice, myši nebo prostřednictvím nějakého hlasového vstupu. Další výhodou takovéhoto rozdělení je usnadnění případných budoucích změn. Změny požadavků na způsob zadávání požadavků a prezentaci výsledků patří přitom k nejčastějším.



Obrázek 15 Model view controller

Aplikace vzoru usnadňuje přenositelnost mezi platformami. V jednodušších aplikacích se části ale často slučují [8].

Obecně řečeno, vytváření aplikací s využitím architektury MVC vyžaduje vytvoření tří komponent, mezi které patří:

- **Model (model)**, což je doménově specifická reprezentace informací, s nimiž aplikace pracuje.
- **View (pohled)**, který převádí data, reprezentovaná modelem, do podoby vhodné k interaktivní prezentaci uživateli.
- **Controller (ovládání)**, který reaguje na události (typicky pocházející od uživatele) a zajišťuje změny v modelu nebo v pohledu.

3.1.2 Singleton

Návrhový vzor Singleton (Jedináček) [8] zabezpečuje, že třída nebude mít více než jednu instanci. Standardní implementace využívá soukromý konstruktor, soukromý statický atribut odkazující na tuto instanci, a veřejnou jednoduchou tovární metodu, která vrací uchovávaný odkaz.

Konstruktor lze zavolat přímo v deklaraci statického atributu, který je pak možno deklarovat jako konstantu. V některých případech je výhodné definovat konstantní atribut jako veřejný a vyhnout se tím nutnosti používání tovární metody. Obecně se však takovéto řešení příliš nedoporučuje.

Je-li vytvoření instance náročnější, může tovární metoda iniciovat její vytvoření až při první žádosti o odkaz na instanci. Tím zabezpečí, že se instance vytvoří pouze v případě, kdy ji bude doopravdy někdo potřebovat. Toto se nazývá odložená inicializace (lazy initialization).

Ve vícevláknových aplikacích je třeba u odložené inicializace ošetřit, aby inicializaci jedináčka nemohla souběžně vyvolat dvě různá vlákna.

Třída singleton by neměla být serializovatelná. Je-li její serializovatelnost žádoucí, je třeba zabezpečit, aby metody načítající instanci ze vstupního proudu vracely odkaz na existující singleton.

Při použití několika zavaděčů tříd je třeba ošetřit, aby každý zavaděč nezavedl svoji verzi singleton třídy a nevytvořil tak i vlastní instanci singleton.

Třída singleton může mít řadu potomků a tovární metoda vracející odkaz na singleton se může při jeho inicializaci na poslední chvíli rozhodnout, která třída ten správný singleton dodá.

3.1.3 Observer

Návrhový vzor *Observer* (Pozorovatel) [8] zavádí vztah mezi objekty (pozorovateli) reagujícími na změnu stavu (pozorovaného) objektu nebo na jím sledované události. Pozorovatelé se u pozorovaného objektu přihlásí a ten je pak na každou změnu svého stavu či výskyt události upozorní.

Vydavatel je v anglickém originále referovaný jako *subject, Předplatitel* jako *observer*.

Mezi předplatiteli a vydavateli zavádí vztah N:1 (jeden vydavatel má mnoho předplatitelů). V případě, že na straně vydavatele dojde ke změně vztahu, či výskytu nějaké události, upozorní na to automaticky všechny své předplatitele. Vydavatel zná své předplatitele a může jich mít neomezené množství. Vydavatel má také veřejné rozhraní, skrze které se předplatitelé mohou přihlašovat nebo odhlašovat. Předplatitel má veřejné rozhraní, které má být použito k jeho upozornění na změnu na straně vydavatele. Toto veřejné rozhraní předá vydavateli při přihlašování. Alternativně může předplatitel vydavateli předat sám sebe jako parametr.

Při rozdělování systému na soubor spolupracujících tříd je třeba zachovat konzistenci mezi souvisejícími objekty. Řešení konzistence prostřednictvím těsného svázání objektů není žádoucí, protože snižuje znovupoužitelnost objektů.

Při upozorňování předplatitelů na změny na straně vydavatele je možné implementovat dvě různé strategie – tažnou a tlačnou.

U tažné strategie vydavatel informuje předplatitele o změně svého stavu. Předplatitel si následně vyžádá od vydavatele parametry, které ho zajímají. Nevýhodou této strategie je, že předplatitel musí mít funkci, která bude vydavatele žádat o parametry. Strategii je vhodné použít v případech, kdy náklady vynaložené na následnou komunikaci jsou menší než náklady, které by byly vynaložené pro rozeslání všech parametrů všem předplatitelům.

U tlačné strategie vydavatel zároveň s informací o změně svého stavu předá předplatitelům všechny své parametry. Předplatitel si vybere pouze ty, které ho zajímají. Tím je vzájemná komunikace omezena na minimum. Tuto strategii je vhodné použít v případech, kdy předplatitele zajímají všechny, nebo většina parametrů, které rozesílá vydavatel. Nebo v případech, kdy by objem následné komunikace (předplatitel si vyžádá parametry, vydavatel je zasílá) přesáhl celkový objem komunikace, kdy vydavatel posílá všechny své parametry přímo.

Použití vzoru má několik výhod. Abstraktní spojení mezi vydavatelem a předplatitelem. Každý vydavatel zná seznam svých předplatitelů, z nichž každý implementuje rozhraní abstraktní třídy *Observer*. Díky tomu je spojení mezi vydavatelem a předplatitelem volné (vydavatele nezajímá jaké třídy je předplatitel). Podpora všesměrového vysílání. Na rozdíl od běžné komunikace, kdy objekty přesně určí adresáta své zprávy, návrhový vzor *Observer* podporuje všesměrovou komunikaci. Vydavatel neadresuje konkrétně každého svého předplatitele, ale automaticky posílá zprávu všem předplatitelům, kteří jsou k němu přihlášeny [10].

3.2 Normalizace databáze

Při návrhu databáze systému je nutné provést normalizaci tabulek. Normalizace databázového modelu je sada pravidel, jak by se mělo postupovat při transformaci struktury entit a relací ER modelu na strukturu fyzického uspořádání tabulek a relací v databázi.

Normalizace je odstranění redundantních (opakujících se) dat, omezení složitosti (rozložení složité relace na dvojrozměrné tabulky) a zabránění tzv. aktualizačním anomáliím (např. abychom smazáním všech knih autora nepřišli o data o autorovi). Což by mělo vést k databázi přehlednější, rozšiřitelnější a výkonnější.

Normalizace by měla vést ke vzniku tabulek, které lze snadno udržovat a efektivně se na ně dotazovat. Normalizované schéma musí zachovat všechny závislosti původního schématu a relace musí zachovat původní data, což znamená, že je nutné se pomocí přirozeného spojení dostat k původním datům.

Existují následující normální formy tabulek [13]:

- První normální forma
- Druhá normální forma
- Třetí normální forma
- Boyce-Coddova normální forma
- Čtvrtá normální forma
- Pátá normální forma

3.2.1 První normální forma

Relace je v první normální formě, pokud každý její atribut obsahuje jen atomické hodnoty. Tedy hodnoty z pohledu databáze již dále nedělitelné.

3.2.2 Druhá normální forma

Relace se nachází v druhé normální formě, jestliže je v první normální formě a každý neklíčový atribut je plně závislý na primárním klíči, a to na celém klíči a nejen na nějaké jeho podmnožině. Z čehož vyplývá, že druhou normální formu musíme řešit pouze v případě, že máme vícehodnotový primární klíč.

3.2.3 Třetí normální forma

V této formě se nachází tabulka, splňuje-li předcházející dvě formy a žádný z jejich atributů není tranzitivně závislý na klíči. Jiné vyjádření téhož říká, že relace je v 3. NF, pokud je ve 2. NF a všechny neklíčové atributy jsou navzájem nezávislé.

3.2.4 Boyce-Coddova normální forma

Boyce-Coddova normální forma se pokládá za variaci třetí normální formy a dokonce je původní definicí 3. NF tak jak byla publikována v 70 letech. Je vymezena stejnými pravidly jako 3. NF forma a říká, že musí platit i mezi hodnotami uvnitř složeného primárního klíče.

Relace se nachází v BCNF, jestliže pro každou netriviální závislost $X \rightarrow Y$ platí, že X je nadmnožinou nějakého klíče schématu R. Dále musí platit:

- Relace musí mít více kandidátních klíčů
- Minimálně 2 kandidátní klíče musí být složené z více atributů
- Některé složené kandidátní klíče musí mít společný atribut.

3.2.5 Čtvrtá normální forma

Tabulka je ve čtvrté normální formě, je-li v BCNF a popisuje pouze příčinnou souvislost (jeden fakt).

Existuje jiná definice: Relace je ve čtvrté normální formě, pokud je v BCNF, a navíc všechny vícehodnotové závislosti jsou zároveň funkčními závislostmi z kandidátních klíčů.

Pro lepší pochopení existuje ještě další definice: Relace je ve čtvrté normální formě tehdy, je-li v BCNF a všechny vícehodnotové závislosti obsažené v relaci jsou zároveň funkčními závislostmi. Vícehodnotovou závislost atributů lze definovat následovně: V relaci R, která je v BCNF, s atributy A, B, C nastává vícehodnotová závislost atributu B na atributu A právě tehdy, jestliže množina hodnot B přiřazená dvojici hodnot A, C závisí jen na hodnotě atributu A a je nezávislá na hodnotě atributu C.

Čtvrtá normální forma se zabývá vztahy uvnitř složeného primárního klíče. Pokud je v tabulce složený primární klíč, může se stát, že některé hodnoty tohoto klíče jsou na sobě nezávislé, ale tím, že spolu tvoří klíč, vzniká falešná souvislost mezi těmito hodnotami a nemohou existovat nezávisle na sobě, což není v souladu s modelovanou realitou. 4. NF proto vyžaduje, aby klíč tvořily jen ty hodnoty, které mají skutečnou vzájemnou souvislost.

3.2.6 Pátá normální forma

Relace je v páté normální formě, pokud je ve čtvrté a není možné do ní přidat další atribut (skupinu atributů) tak, aby se vlivem skrytých závislostí rozpadla na několik dílčích relací.

Další definice zní: Relace je v páté normální formě jestliže je ve 4. NF a nemůže-li být dále bezeztrátově rozložena. Jinými slovy relace, která má n klíčových atributů ($n >= 3$) a která se rozloží na relace o ($n - 1$) klíčových atributech, nemůže být opětovně spojena operací přirozeného spojení do jedné relace, aniž by došlo ke ztrátě informace.

3.3 Systém GPS

V mobilní části systému bude využíván GPS modul za pomocí GPS API. Global Positioning System, zkráceně GPS, je vojenský polohový družicový systém provozovaný Ministerstvem obrany Spojených států amerických, s jehož pomocí je možno určit polohu a přesný čas kdekoliv na Zemi nebo nad Zemí s přesností desítky metrů. Přesnost GPS lze s použitím dalších metod ještě zvýšit až na jednotky centimetrů. Část služeb tohoto systému s omezenou přesností je volně k dispozici i civilním uživatelům.

V současné době se systém využívá v mnoha oborech lidské činnosti. Na provoz GPS se ročně vynakládá přibližně 600 až 900 milionů (2006–2008) amerických dolarů z rozpočtu USA.

3.3.1 Funkce

Zjednodušeně lze družicový polohový systém popsat jako *družicový radiový dálkoměrný* systém:

- **Dálkoměrný** systém je takový, kdy se poloha nějakého objektu určuje ze vzdáleností od bodů se známou polohou. Např. v krajině lze určit polohu pomocí mapy a dalekohledu, který umí změřit vzdálenost od pozorovaného objektu. Dalekohledem změříme vzdálenost ke dvěma význačným objektům a kružítkem na mapě nakreslíme kolem každého objektu kružnici o změřeném poloměru. Zjišťovaná poloha je v jednom z průsečíků obou kružnic.
- **Rádiový** systém pro měření určitého parametru využívá rádiových vln. „Rádiový dálkoměrný“ systém k měření vzdálenosti využívá radiových vln takto. Do bodu se známou polohou je umístěn vysílač, který vysílá rádiové vlny s časovými značkami. V bodě, jehož poloha se měří, umístíme přijímač, který porovnává časové značky se svými „hodinami“. Tím je možno změřit *zpoždění*, tj. jak dlouho trvalo rádiové vlně, než k přijímači dorazila. Protože se radiové vlny pohybují známou rychlostí, stačí pro výpočet požadované vzdálenosti vynásobit změřené zpoždění touto rychlostí.
- **Družicový** je systém označován proto, že body se známou polohou jsou družice obíhající Zemi. Aby bylo možno určit polohu družic, musí být v jejich vysílání nejen časové značky, ale i parametry dráhy dané družice.

3.3.1.1 Radiové vysílání

Systém GPS využívá kódové radiové vysílání - CDMA *(Code Division Multiple Access)* [11]. Každá družice vysílá různé kódy na stejné frekvenci, které se svou charakteristikou blíží náhodnému kódu, a proto se označují za PRN (*Pseudo Random Numbers*). Přijímač pak na základě znalosti tohoto kódu snadno zesílí hledaný signál a odfiltruje jako šum ostatní.

3.3.1.2 Vztažné soustavy a určování polohy a času

Pro charakteristiku Země se jako vztažné těleso využívá geoid, který je ale pro matematický popis nevhodný. Proto používáme jeho aproximaci prvního stupně – koule, nebo druhého stupně - elipsoid. Pro potřeby uživatelů GPS je nejčastěji užívaný geografický referenční systém WGS 84, známý také pod kódem EPSG:4326, který se skládá z:

- geodetického data: elipsoid s poloosami přibližně 6 378 a 6 356 km s počátkem ve středu Země
- systému zeměpisných souřadnic (zeměpisná šířka a délka)

Pro výpočty se používá geocentrický referenční systém WGS 84 se shodným datem ale s kartézskými souřadnicemi v systému ECEF (*Earth-Centered, Earth-Fixed*).

GPS čas je měřen na týdny s maximem 1024, díky čemu dochází k jeho vynulování, což bylo naposledy 22. srpna 1999. Další časová značka je pořadí podrámce v navigační zprávě, který nabývá hodnot s maximem 100800, dále slova podrámce a jeho datové bity, které mají délku 0,02 s. Poslední podrobný časový otisk je samotný kód. C/A kód rozděluje čas po bitech dlouhých ~10^{-6} s a P kód na ~10^{-7} s. Porovnáním vzestupných a sestupných hran PRN kódů modulovaných s frekvencí nosné vlny může moderní elektrotechnika změřit rozdíl až na tisíciny času bitu. Za předpokladu přesnosti 1% bitu je to přibližně 10 ns (10^{-8} s) pro C/A kód a 1 ns (10^{-9} s) pro P(Y). Protože signál GPS se šíří rychlostí blízkou rychlosti světla blíží se krok měření při 1% délky bitu řádově ~3 m u C/A kódu, u P(Y) ~0,3 m.

Rychlost světla je definována 299 792 458 m/s Odeslaný signál má při přijetí zpoždění mezi 67 ms při elevaci družice 90° a 86 ms při elevaci 0°.

GNSS systémy jsou obvykle navrženy k jednomu principiálně jednoduchému způsobu výpočtu polohy, přesto je však možno ve speciálních aplikacích uplatnit jiné metody:

- Kódová
- Fázová
- Dopplerovská
- Úhloměrná

GPS využívá kódovou metodu. Měření jsou jednoduchá a spolehlivá. Na základě časových značek a známé pozice vysílačů (družic) je možno spočítat polohu a čas v místě přijímače.

Po přijetí rádiového signálu jsou v přijímači dekódovány:

1. Časové značky při odeslání signálu každé družice (t).
2. Polohy každé družice v prostoru, tzv. efemeridy (x, y, z).

Hledáme-li pozici uživatele v prostoru, musíme ji popsat třemi souřadnicemi např. v kartézském systému *ECEF*(X, Y, Z). Protože čas v přijímači není pro potřeby výpočtu přesný a synchronní, je čas uživatele také proměnná (T). Neznámé jsou tedy (X, Y, Z, T) a proto můžeme sestavit 4 rovnice koule $(X - x_n)^2 + (Y - y_n)^2 + (Z - z_n)^2 = [(T - t_n)c]^2$ o 4 neznámých, kde c je rychlost světla a za předpokladu, že známe (x, y, z, t) pro 4 družice (n = 1, 2, 3, 4), je řešením rovnice poloha a čas uživatele.

Pro převod do zeměpisných souřadnic a občanského času se využívá definovaných matematických vztahů.

Např. platí:

$$ECEF(X, Y, Z, T) \rightarrow WGS84(lat, lon, HAE, UTC)$$

kde:

- lat – zeměpisná šířka
- lon – zeměpisná délka
- HAE – výška nad elipsoidem (*Height Above Ellipsoid*)
- UTC – čas (*Coordinated Universal Time*)

Pro získání *výšky vztažené k hladině moře* (*MSL, Mean Sea Level*) je třeba opravit výšku HAE o hodnotu separace geoidu. Na území Česka se jedná u WGS 84 řádově o hodnoty -40 až -50 m. Výpočetní jednotka GPS přijímače již přibližný model obsahuje a opravu provádí automaticky. Uživatel má obvykle k dispozici obě hodnoty výšek HAE i MSL.

Pro výpočet se používají pouze družice, které jsou nad obzorem výše, než limitní hodnota, běžně 5° až 10°. Toto opatření se nazývá elevační maska a používá se proto, že radiový signál nízko nad obzorem delší dráhou více ovlivňuje atmosféra, než družice ve vyšších pozicích a má náchylnost k vícecestnému šíření.

V případě příjmu signálu z více než 4 družic je poloha váženým průměrem, tak aby výhodná geometrická poloha družice a kvalitní radiový signál hráli významnější roli, čímž může být výsledek výrazně stabilnější a přesnější. Pokud jsou ve výpočtu jen 3 družice, je určena poloha pouze na povrchu elipsoidu (*lat*, *lon*, *HAE* = 0, *UTC*), často označovaná jako neplnohodnotná navigace *2D*. Pokud jsou ve výpočtu jen 2 družice, lze teoreticky určit výšku nad elipsoidem (*lat* = 0, *lon* = 0, *HAE*, *UTC*), toto řešení se však nepoužívá, neboť skutečná nadmořská výška je polohově závislá.

Výsledek výpočtu předává přijímač dále ke zpracování pomocí standardizovaných formátů zpráv (NMEA, RTCM, RINEX, SiRF) skrze komunikační rozhraní (Bluetooth, sériový port).

3.3.1.3 Efemeridy

Efemeridy[4] jsou predikované polohy družic na oběžných drahách. Protože se pohybují po téměř kruhových, mírně elipsovitých drahách velkou rychlostí a ve velké vzdálenosti od Země, jsou jejich dráhy stabilní a dobře matematicky popsatelné. Přesto se vlivem kolísání tíhových sil Země, Slunce a Měsíce a sluneční jaderné aktivity jejich dráha mírně mění. Předpoklad vývoje trajektorie je popsán v navigační zprávě.

3.3.2 Rozšiřující systémy GNSS

GNSS mohou být dále rozšířeny systémy uvedenými v dalších podkapitolách [11].

3.3.2.1 SBAS

SBAS *(Satellite Based Augmentation Systems)* je obecný název pro systém pozemních monitorovacích stanic rozšiřujících původní monitorovací segment GNSS, které v reálném čase vyhodnocují aktuální stav kosmického segmentu GNSS (typicky GPS+GLONASS) a stav ionosféry. Vypočítávají korekce těchto vlivů a tyto data v malém časovém zpoždění vysílají k uživatelům skrze družice na geostacionární dráze. Geostacionární družice mají čísla #ID nad 32. Nevýhodou systému je umístění družic nad rovníkem (v Česku nízko nad jižním horizontem) se slabým vysílacím výkonem, který určuje použití jen pro leteckou, případně námořní dopravu.

3.3.2.2 GBAS

GBAS (*Ground Based Augmentation Systems*), označované někdy jako GRAS (*Ground-based Regional Augmentation Systems*) je obecný název pro systém pozemních referenčních stanic, které v reálném čase vyhodnocují aktuální stav kosmického segmentu GNSS (typicky GPS+GLONASS). Vypočítávají korekce vzhledem ke své absolutní poloze a poskytují je uživatelům pomocí mobilních sítí, radiových vysílání nebo až zpětně pro korekce prováděné po skončení měření.

3.3.2.3 IGS

IGS (*International GNSS Service*) je mezinárodní organizace, která sleduje a vyhodnocuje kosmické segmenty GNSS. Klíčovými produkty tří stovek stanic jsou zpětně dopočtené a velmi přesné:

- efemeridy družic GPS/GLONASS ~5 cm/15 cm (predikce v navigační zprávě GPS: ~160 cm)
- přesné korekce pro palubní hodiny GPS ~ 0,1 ns (predikce v navigační zprávě GPS: ~ 7 ns)
- ionosférické a troposférické zpoždění
- 3D souřadnice monitorovacích stanic ~3 mm a jejich pohyb ~2 mm/rok
- parametry rotace Země *(Earth Rotation Parameters*)

3.3.2.4 ILRS

ILRS (*International Laser Ranging Service*) je vyvíjená služba umožňující nezávislé zjišťování polohy družic na oběžné dráze za pomoci laserových měřidel (bez ohledu na vysílaný radiový signál). Pro měření je nutné vybavit družici odražečem a jejich nasazení je plánované pro GNSS II. generace.

3.3.3 Protokol NMEA

NMEA [12] je v kombinace elektronické a datové specifikace pro komunikaci mezi námořními elektronickými zařízeními, jako je echolot, sonary, anemometr (měří rychlost a směr větru), kompas, autopilot, GPS přijímače a mnoho dalších podobných přístrojů. Byl založen a nyní je řízen asociací National Marine Electronics Association.

NMEA 0183 standard používá jednoduchý ASCII sériový komunikační protokol, který definuje, jakým způsobem se data předávají ve *větě* od jednoho *mluvčího* více *posluchačům* v čase. Pomocí dílčích expandérů může řečník mít jednosměrný rozhovor s téměř neomezeným počtem posluchačů, a za použití multiplexerů může více přijímačů komunikovat s jedním portem počítače. Aplikační vrstva definuje jednotlivé položky každého typu věty (zprávy), takže posluchači vědí, jak správně rozebrat zprávu.

3.3.3.1 Sériové nastavení (datová vrstva)

Přenos dat je zajištěn při následujícím nastavení:

- Přenosová rychlost – 4800 b/s
- Počet bitů – 8
- Parita – žádná
- Stop bit – 1
- Řízení toku dat – žádné

3.3.3.2 Pravidla aplikační vrstvy

- Každá zpráva začíná znakem dollaru.
- Dalších pět znaků identifikují odesílatele (dva znaky) a typ zprávy (tři znaky).
- Všechna datová pole, která následují, jsou oddělena čárkou.
- Pokud jsou data nedostupná, korespondující pole obsahuje *NULL byte* (příklad: „123,,513" znamená, že druhé datové pole je nedostupné).
- První bezprostředně následující charakter, který následuje je hvězdička.
- Hvězdička je dále následována dvouciferným kontrolním součtem, reprezentující číslo v hexadecimálním tvaru. Tento kontrolní součet je exklusivní disjunkcí všech čísel mezi dolarem a hvězdičkou. Podle oficiální specifikace je kontrolní součet volitelný pro všechny datové věty, ale je povinný pro RMA, RMB a RMC
- Zpráva je zakončena <CR><LF>

Příklad

```
$GPAAM,A,A,0.10,N,WPTNME*32 $ GPAAM, A, A, 0,10, N, WPTNME *
32
```

kde:

GP	ID mluvčího (*GP* pro GPS jednotku, *GL* pro GLONASS)
AAM	Alarm
A	Vstup do příjezdového kruhu
A	Průchod přes kolmou přímku
0.10	Poloměr
N	Námořních mil
WPTNME	Název průjezdního bodu
*32	Kontrolní součet data

Sloučený řetězec z přijímače NR203 GPS obsahuje více zpráv: Dekódovaná zpráva obsahuje:

- ZDA – standard NMEA $..ZDA čas a datum zprávy
- GLL – standard NMEA $..GLL geografická poloha - zeměpisná šířka / délka
- NSV – NMEA zpráva obsahující individuální satelitní informace.

Typický ASCII řetězec

```
$<CR><LF>
MRK,0<CR><LF>
ZDA,123336.8069,17,06,2001,13.0<CR><LF>
GLL,4916.45,N,12311.12,W,225444,A,*1D<CR><LF>
VTG,218.7,T,2.38,H,0.18,V<CR><LF>
SGD,-1.0,G,-1.0,M<CR><LF>
SYS,3T,9<CR><LF>
ZEV,0.28745E-006<CR><LF>
NSV,2,00,000,00,0.0,00.0,00,00,D<CR><LF>
NSV,7,00,000,00,0.0,00.0,00,00,D<CR><LF>
NSV,28,00,000,00,0.0,00.0,00,00,N<CR><LF>
NSV,1,00,000,00,0.0,00.0,00,00,D<CR><LF>
NSV,13,00,000,00,0.0,00.0,00,00,D<CR><LF>
NSV,4,00,000,00,0.0,00.0,00,00,N<CR><LF>
NSV,25,00,000,00,0.0,00.0,00,00,N<CR><LF>
NSV,0,00,000,00,0.0,00.0,00,00,N<CR><LF>
NSV,11,00,000,00,0.0,00.0,00,00,D<CR><LF>
NSV,0,00,000,00,0.0,00.0,00,00,N<CR><LF>
&
```

3.4 Návrh modelu tříd

Návrh modelu tříd se skládá ze dvou hlavních fází. Návrhu analytických tříd a návrhu návrhových tříd [1].

Analytické třídy vyjadřují velmi přesné definované zobecnění (abstrakci) entity problémové domény. Problémovou doménou je doména, z níž vešel požadavek na nový softwarový systém. Analytické třídy by měly být jednoznačným způsobem mapovány na pojem používaný ve skutečném světě. Obchodní pojmy je třeba si během analýzy dobře ujasnit.

Analytický model obsahuje pouze analytické třídy. Tento model nesmí obsahovat žádné třídy vzniklé z návrhových úvah. Třídy obsahují množinu hlavních kandidátů na atributy a množinu hlavních operací.

Dobrá analytická třída v názvu zrcadlí její účel. Je výlučným zobecněním, které modeluje jeden specifický prvek problémové domény. Třída je mapována na jasně a zřetelně definovanou vlastnost problémové domény a vlastní malou, přesně definovanou množinu odpovědnosti. Tato odpovědnost je dohodou nebo závazkem třídy vůči klientům. Je to sémanticky soudržná množina operací a každé třídě bychom měli přidělovat tři až pět odpovědností. Dobrá analytická třída je dále vysoce soudržná. Všechny charakteristické vlastnosti třídy by měly pomáhat v určení jejího účelu. Vyznačuje se počtem vazeb na okolí třídy. Taková třída by měla spolupracovat jen s několika třídami, s jejichž pomocí může plnit zadané úlohy.

Třídy můžeme hledat následujícími způsoby:

1. Analýza podstatných jmen a sloves
   - Podstatná jména a jejich spojení jsou kandidáti na třídy a atributy.
   - Slovesa nebo slovesné fráze jsou kandidáti na odpovědnosti nebo operace.
2. Metoda štítků CRC
   - Důležité předměty problémové domény jsou zapisovány na lístečky.
   - Každý lísteček je rozdělen na tři části – třída, odpovědnost, spolupracovníci.
3. Rozvaha o dalších zdrojích nebo třídách, jako jsou fyzické objekty, kancelářské pomůcky, rozvahy k vnějšímu světu a koncepční entity.

Po provedení výše uvedených úkonů se porovnají jejich jednotlivé výsledky a na jejich základě je vytvořen prvotní analytický model.

Návrhové třídy jsou třídy, jejichž specifikace je na takovém stupni, že je lze implementovat.

Během analýzy je zdrojem tříd problémová doména. Je to množina požadavků, která popisuje problém, jež se snažíme vyřešit. Návrhové třídy lze získat ze dvou zdrojů:

- Z problémové domény prostřednictvím upřesňování analytických tříd – součástí upřesňování je rovněž doplňování implementačních detailů. Během této činnosti se často stává, že je třeba koncepční analytickou třídu rozbít na několik podrobných návrhových tříd.
- Z domény řešení. Doména řešení je sférou knihoven užitkových tříd a znovupoužitelných komponent, jako jsou `DateTime`, `String`, kolekce apod. Je to rovněž místo, v němž existuje střední vrstva. Doména řešení obsahuje rovněž pracovní prostředí grafického uživatelského rozhraní *(GUI frameworks).* Tato doména poskytuje technické nástroje, jež umožňují implementaci systému.

Návrhový model je základ pro samotné programování a vytváření zdrojového kódu. Kód může být generován přímo z modelu, za předpokladu že modelovací nástroj takovou funkci obsahuje. Enterprise Architect 7.5 mezi tyto nástroje patří, nicméně při návrhu systému, který je tématem této práce, byla celá implementace provedena ručně. Návrhové třídy musí být precizně specifikovány. Jednou z částí procesu je specifikace, zda jsou třídy správně formulovány.

Při tvorbě návrhové třídy je důležité nahlížet na třídu z pohledu jejích potencionálních klientů tak, jak ji budou vidět oni. Správně formulované třídy splňují následující podmínky:

- Úplnost a dostatečnost.
- Jednoduchost.
- Vysoká soudržnost.
- Minimalizace vazeb.

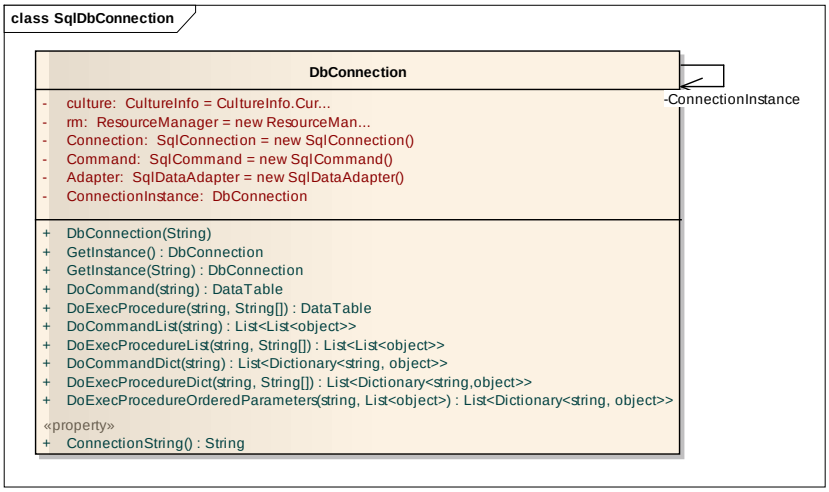
Na základě takovéhoto návrhu lze dobře vytvořit odpovídající zdrojový kód s očekávanou funkcionalitou.

Dle výše uvedených postupů byly vytvořeny dále uvedené modely tříd. Všechny třídy byly navrženy tak, aby splňovaly požadavek na vícejazyčnost aplikace.

3.4.1 Třída databázového připojení

Tato třída bude sloužit k vytvoření instance databázového připojení, a to za dodržení pravidel návrhového vzoru *Singleton*.

Pro zjištění instance databázového připojení se bude vždy volat metoda `GetInstance`, které nejprve zjistí, zda již neexistuje dříve vytvořená instance připojení, a pokud ne, vytvoří dle zadaných vstupních parametrů instanci novou. Instance bude uchována ve statickém atributu třídy.



Obrázek 16 Třída databázového připojení

Použitím tohoto návrhového vzoru bude zaručeno, že v celé aplikaci bude vytvořena a používána pouze jedna instance připojení. Třída bude obsahovat několik odlišných metod pro volání příkazů a získávání dat z databáze:

- `DoCommand` – pouze provede zadaný příkaz a vrátí výsledek.
- `DoExecProcedure` – spustí zadanou databázovou proceduru, včetně parametrů předaných v podobě pole textových řetězců a vrátí výsledek.
- `DoCommandList` – bude pracovat stejně jako `DoCommand` s tím rozdílem, že výsledek bude vracen v kolekci objektů.
- `DoExecProcedureList` – bude pracovat stejně jako `DoExecProcedure`, s tím rozdílem, že výsledek bude vrácen v kolekci objektů.
- `DoCommandDict` – bude pracovat stejně jako `DoCommand` s tím rozdílem, že výsledek bude vracet jako seznam slovníků. Index seznamu pak bude ukazovat na řádek a klíč bude ukazovat na jednotlivý sloupec.
- `DoExecProcedureDict` – bude pracovat stejně jako `DoExecProcedure` s tím rozdílem, že výsledek bude vracet opět jako seznam slovníků.
- `DoExecProcedureOrderedParameters` – tato metoda bude ze všech ostatních nejvhodnější k volání procedur. Název procedury jí bude předán jako textový řetězec. Parametry procedury budou předány opět v poli, ale tentokrát již libovolného typu, použitelného v databázi. Jedinou podmínkou bude dodržení pořadí parametrů očekávaných procedurou. Pro navrácení výsledku bude opět použit seznam slovníků.

Stejná třída bude využita jak v desktopové, tak v mobilní části systému. Z důvodu jiných referencí na systémové knihovny však budou vytvořeny dvě různé knihovny v závislosti na cílové platformě.

3.4.2 Model základních tříd

V tomto modelu jsou zobrazeny základní třídy, které jsou stavebními kameny pro tvorbu jednotlivých modulů aplikace.

Na nejvyšší úrovni bude třída `Item`. Tato třída bude rodičem všech ostatních tříd základního modelu. Bude sloužit pro uchování informací o uživatelích a časech, kdy bude záznam vytvořen a editován. Dále bude obsahovat informaci o platnosti záznamu.

Třída `User` bude uchovávat informace o uživateli. Jako potřebné informace byly vybrány přihlašovací jméno, příjmení, jméno, titul, ulice, číslo popisné, město, PSČ, e-mail, telefonní číslo, rodné číslo, heslo, fotografie a seznam řidičských oprávnění. Dále bude k uživateli zaznamenán příznak s jeho stupni oprávnění. Uživatel bude moci být administrátor, dispečer nebo řidič. Podle těchto oprávnění se dále budou řídit jeho možnosti v systému.

Třída `Vehicle` bude sloužit pro uchování informací o vozidle. Vozidlo bude mít jako své atributy registrační značku, značku, typ, kategorii, stav tachometru, seznam potřebných řidičských oprávnění, datum poslední technické kontroly, číslo pojištění, rok výroby a seznam provedených údržeb.

Výše uvedená kategorie bude instancí třídy `VehicleCategory`. Tato třída nám umožní zaznamenat pro jednotlivé kategorie vozidel jejich popis a intervaly technických kontrol.

Dále byly zmíněny údržby vozidel. K jejich evidenci bude vytvořena třída `VehicleMaintainance`, ve které bude zaznamenán řidič, který údržbu provedl, datum a čas údržby, stav tachometru a poznámky o provedených úkonech.

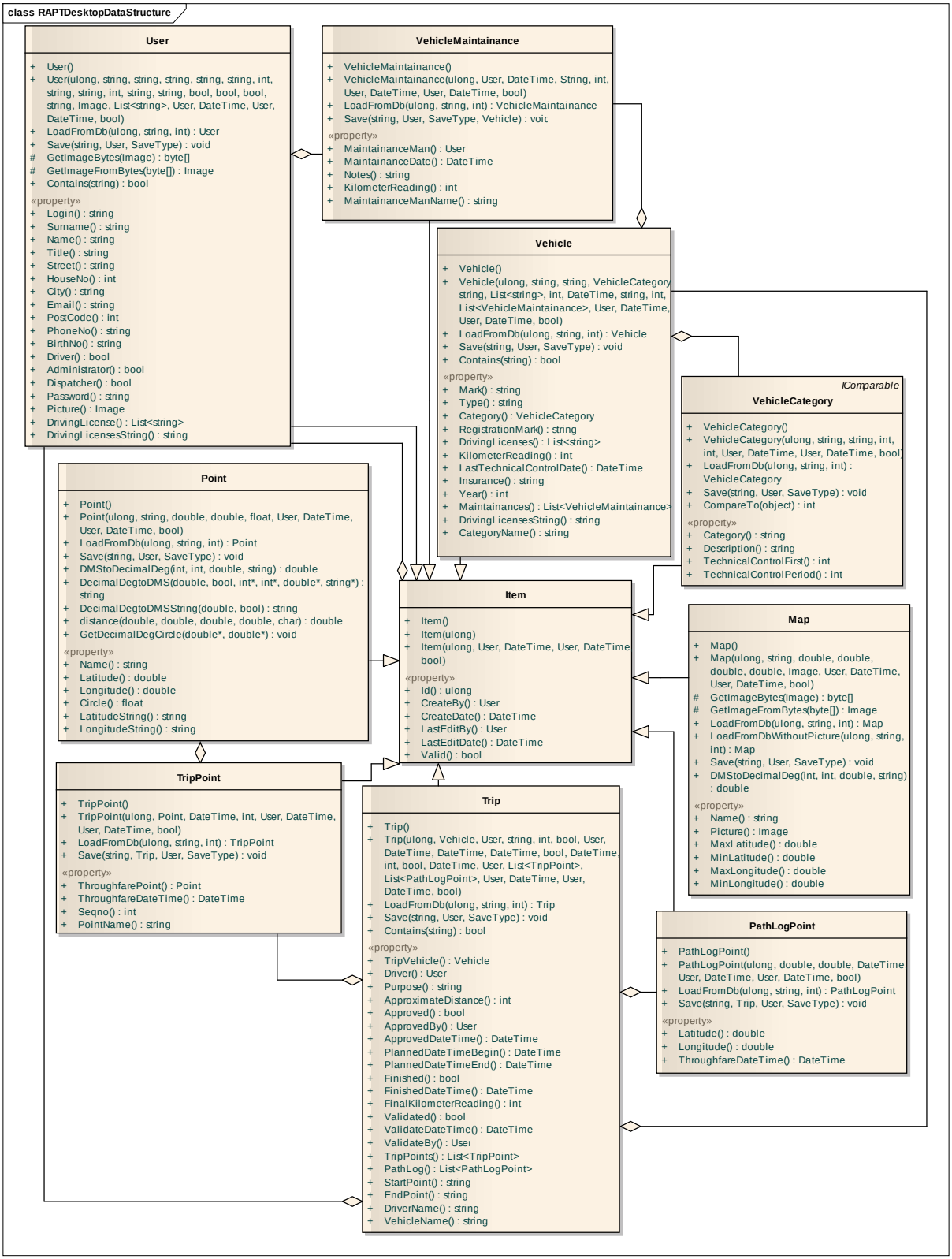
Další třída v diagramu je třída `Map`. Díky této třídě budeme moci používat mapový podklad. Bude uchovávat obrázek mapy, její název a informace o rozsazích. Tato informace bude zajištěna uložením nejmenší a největší zeměpisné šířky a délky v desetinném formátu.

Mapa sama o sobě nám ale neumožní plánovat trasy. K tomu bude ještě zapotřebí mít nadefinované body na mapě – města, části měst apod. Pro tyto účely bude sloužit třída `Point`. V této třídě budeme moct uchovat informaci o názvu bodu, jeho souřadnicích a okruhu. Tento okruh bude sloužit pro zjištění rozsahu kolem souřadnice, ve kterém se území obce rozlehá.

Tuto třídu využijeme ve třídě `TripPoint`, kde bude jedním z atributů. Třída `TripPoint` bude představovat bod na naplánované cestě. K tomuto bodu budeme moci zaznamenat informace o pořadí bodu na naplánované cestě a datum čas průjezdu tímto bodem.

Dalším třídou pro zaznamenávání cesty bude `PathLogPoint`. Tato třída bude sloužit pouze k zaznamenávání průběhu cesty v podobě souřadnic a časů. Umožní neustálé logování cesty.

Nad výše napsanými třemi třídami bude třída `Trip` představující samotnou cestu. Pro kompletní naplánování cesty a zaznamenání jejího průběhu bylo potřeba vybrat větší množství atributů. Budou to řidič, jemuž bude cesta naplánována, vozidlo, se kterým by měl cestu provést, účel cesty, plánovaná délka trasy, plánovaný datum a čas začátku cesty, plánovaný datum a čas konce cesty, seznam průjezdních bodů, seznam bodů záznamu cesty, příznak, zda je cesta schválena včetně uživatele, který cestu schválil a data a času schválení, příznak, zda je cesta ukončena včetně data a času ukončení a konečného stavu tachometru a příznak, zda je cesta ověřena včetně uživatele, který ji ověřil a data a času ověření.



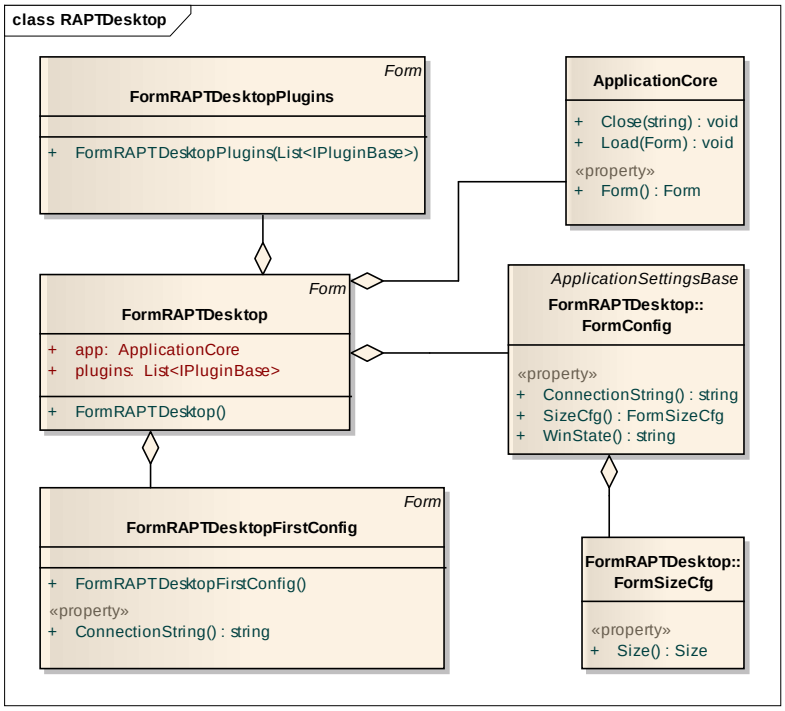
Obrázek 17 Model základních tříd

3.4.3 Model tříd hlavního okna

Hlavní okno bude MDI formulář poskytující prostředí pro dílčí moduly. Bude obsahovat hlavní tlačítkové menu, na kterém budou zobrazena tlačítka pro jednotlivé moduly.

Okno bude integrovat také zásuvné moduly uložené ve složce *Plugins*. K tomu bude mít vytvořenu třídu jádra, kterou bude modulům předávat. Dále bude umožňovat zobrazovat nastavení jednotlivých modulů pomocí formuláře pro nastavení modulů.

Součástí této části systému bude okno pro nastavení připojení k databázi při prvním spuštění. Toto okno umožní nejen uložit tyto informace do uživatelského nastavení, ale také umožní vytvořit databázovou strukturu.



Obrázek 18 Model tříd hlavního okna

3.4.4 Model tříd modulu přihlášení do desktopové části

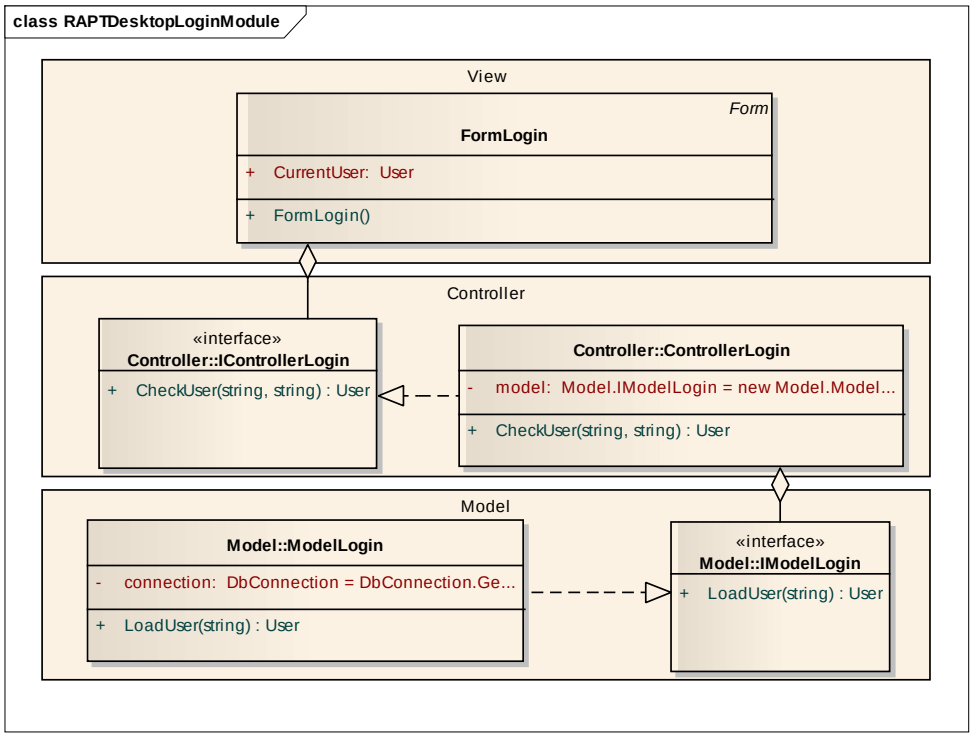
Při tvorbě modulů byla dodržována pravidla stavby aplikací podle návrhového vzoru *Model-View-Controller*. Dle těchto částí je rozdělen i diagram na obrázku.

V části *View* bude třída formulářového okna pro zadání přihlašovacích informací. Po jejich zadání formulář pomocí instance třídy s rozhraním `IControllerLogin` ověří uživatele.

`IControllerLogin` je rozhraní, které bude aplikováno třídou `ControllerLogin`. Ta bude obsahovat metodu pro ověření uživatele. Heslo uživatele bude šifrováno metodou *MD5*. V případě, že ověření proběhne správně, vrátí *controller* instanci uživatele. K tomu využije modelové třídy s rozhraním `IModelLogin`.

`IModelLogin` předepisuje třídě `ModelLogin` obsahovat metodu pro načtení informací uživatele z databáze.

Tento modul bude využíván vždy při spuštění desktopové aplikace.



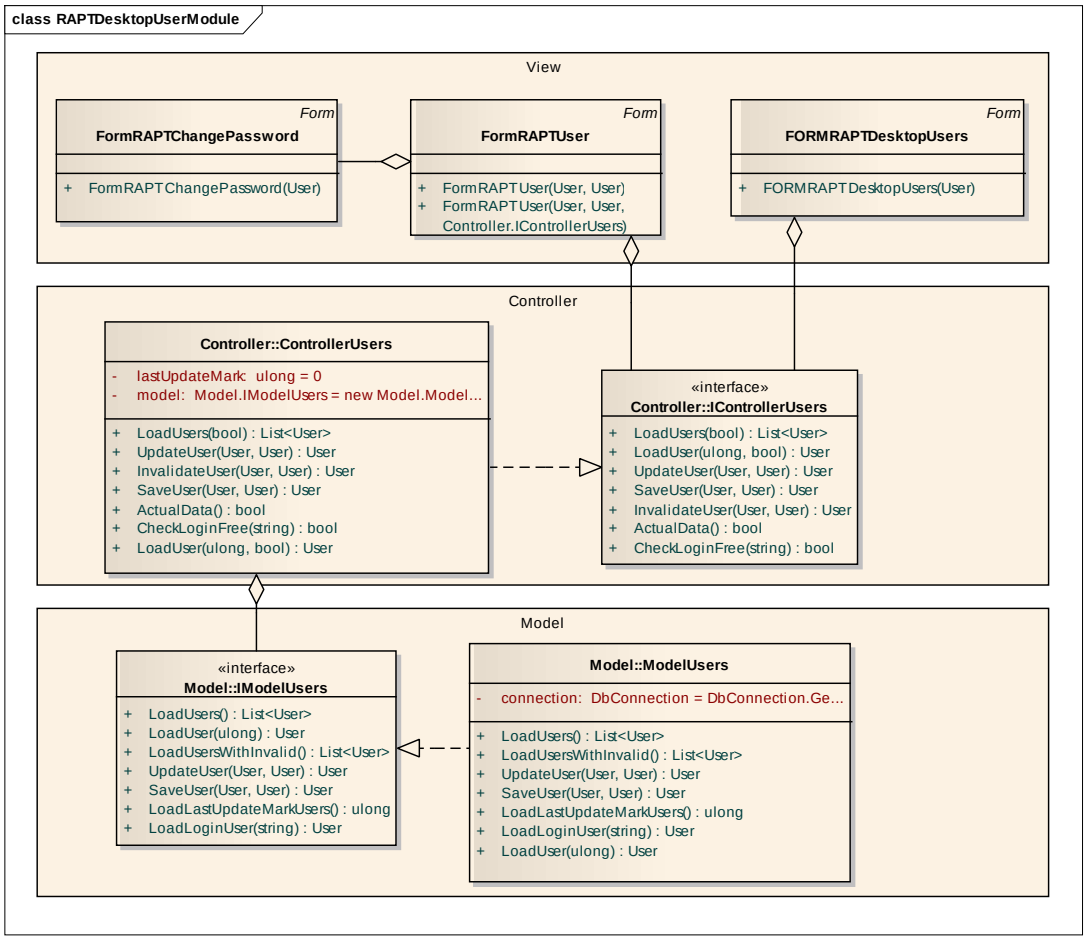
Obrázek 19 Model tříd modulu přihlášení do desktopové části

3.4.5 Model tříd modulu pro správu uživatelů

Tento modul má zabezpečit administrátorovi spravovat jednotlivé uživatele. Dle návrhového vzoru *Model-View-Controller* bude modul opět členěn na tři části.

Část *View* se bude skládat ze tří formulářů. Formulář uživatelů, který má zabezpečovat výpisy uživatelů a spouštění akcí pro jejich editaci, mazání, vyhledávání a přidávání. Dalším formulářem bude formulář s detaily uživatele, sloužící pro změny jeho jednotlivých údajů. Tento formulář má umožňovat pouze změny poplatné uživateli dle jeho oprávnění. Prohlížet své vlastní detaily a změnit si heslo bude moci každý uživatel. Třetím formulářem bude formulář pro změnu hesla.

Dva první formuláře budou využívat *Controller* aplikující rozhraní `IControllerUsers`. Toto rozhraní předepisuje metody pro načtení všech uživatelů, načtení jednoho uživatele, jeho editaci a smazání. Dále musí *Controller* hlídat aktuálnost editovaných dat. Toto rozhraní bude aplikováno třídou `ControllerUsers`.



Obrázek 20 Model tříd modulu pro správu uživatelů

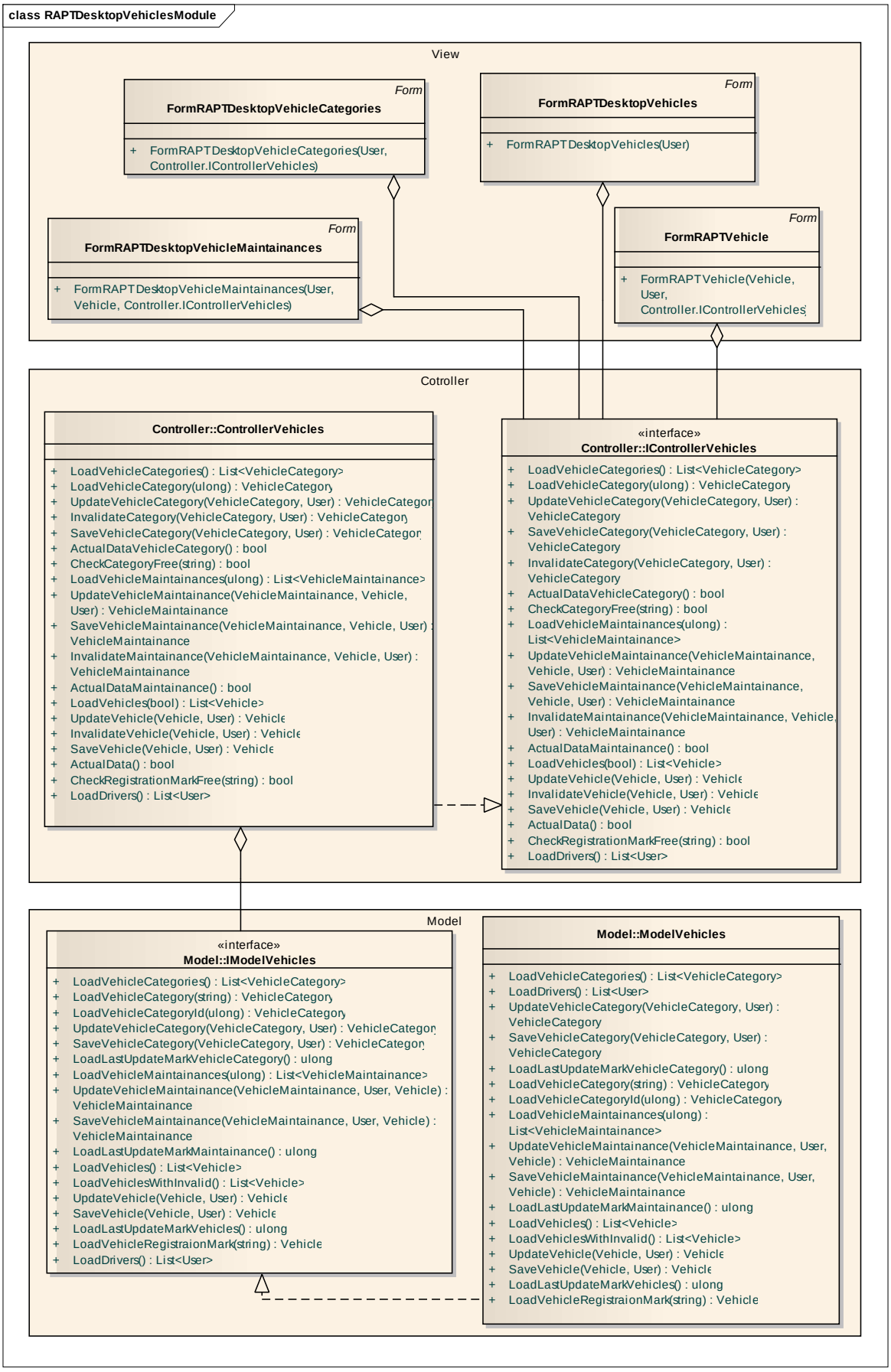
Tato třída bude využívat *Model* aplikující rozhraní `IModelUsers`. Tímto rozhraním jsou předepsány metody pro načítání uživatelů, aktualizaci jejich dat, ukládání nových uživatelů a načítání aktualizačních značek.

3.4.6 Model tříd modulu pro správu vozidel

I modul správy vozidel bude rozčleněn dle návrhového vzoru *Model-View-Controller*.

Část *View* bude obsahovat čtyři formuláře. Hlavní formulář budou sloužit pro výpis vozů a jejich výběr pro další editaci nebo mazání. Dále bude umožňovat přidávat nové vozy a vyhledávat. Druhý formulář bude sloužit pro zobrazení a editaci kategorií vozidel. Třetí bude sloužit pro zobrazení a editaci informací o vybraném vozidle. Dále bude možné z tohoto formuláře volat čtvrtý formulář, sloužící pro přidávání informací o údržbách vozidla.

Tyto formuláře budou využívat instanci třídy aplikující rozhraní `IControllerVehicles`. Toto rozhraní ukládá třídám obsahovat metody pro načítání vozidel, jejich detailů a údržeb, editaci údajů, vytváření a mazání. Dále metody pro načtení, editaci, vytváření a mazání kategorií vozidel a pak další metody pro načtení, přidávání a mazání údržeb vozidla. *Controller* musí také hlídat aktuálnost editovaných dat. Tyto předepsané metody budou implementovány ve třídě `ControllerVehicles`.

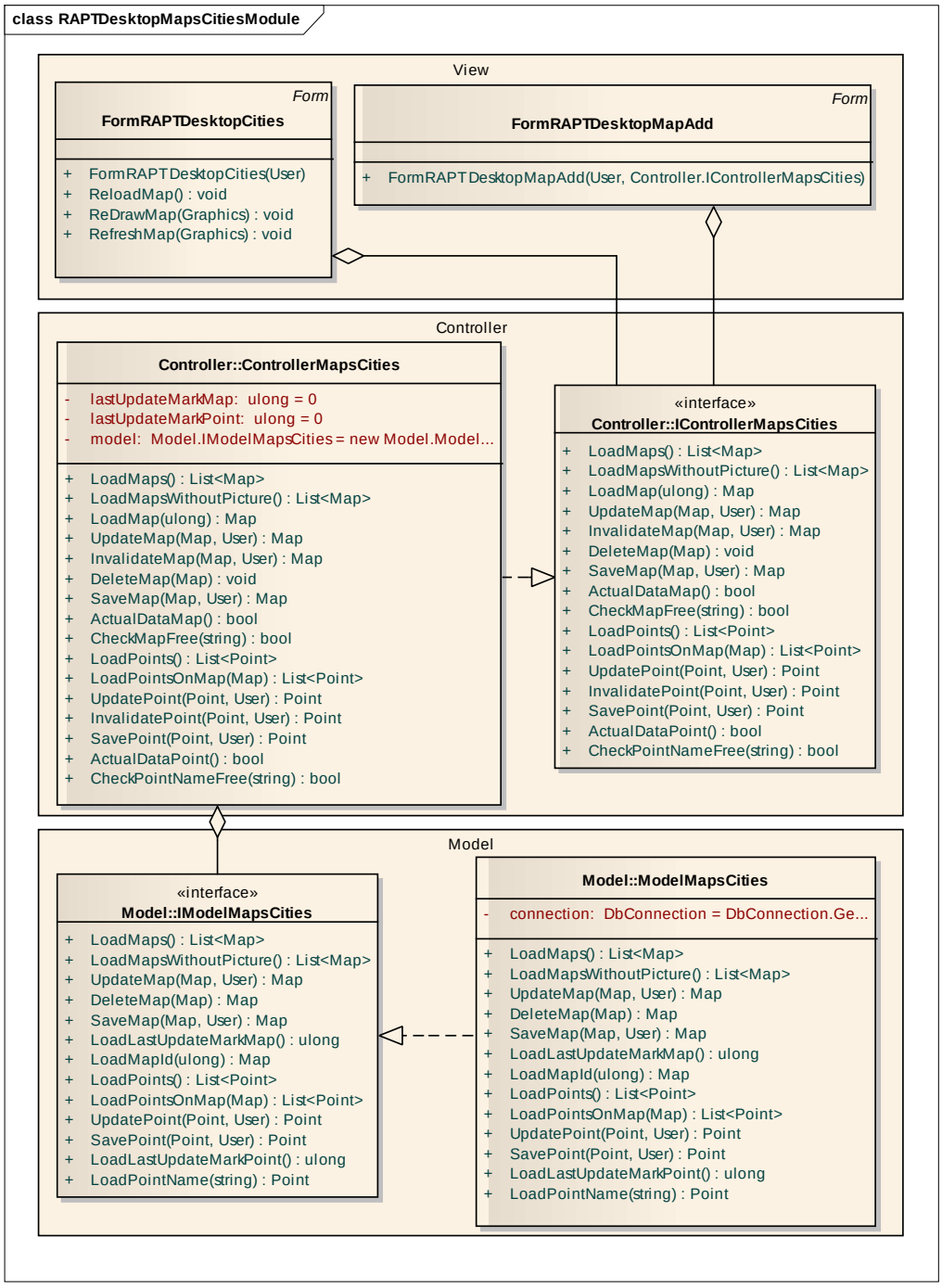


Obrázek 21 Model tříd modulu pro správu vozidel

Část *Model* bude reprezentována třídou `ModelVehicles`, která bude aplikovat rozhraní `IModelVehicles`, předepisující metody pro práci s daty vozidel, údržeb a kategorií nad databází. Funkcionalita bude stejná jako u předchozích modelů, čili načítaní, mazání a aktualizace údajů příslušných tříd v databázi.

3.4.7 Model tříd modulu pro správu map a měst

V této části aplikace bylo nutné navrhnout import mapových podkladů a umožnit nad nimi nadefinovat jednotlivé body, použitelné pro plánování cest. K tomu bude zapotřebí kvalitní grafické uživatelské rozhraní.



Obrázek 22 Model tříd modulu správy map a měst

V části *View* k zabezpečení výše uvedených funkcionalit budou sloužit dvě formulářová okna. První z nich umožní načítat a prohlížet mapy a v případě dostatečných oprávnění bude možné mapu smazat. Z načtené mapy bude možno vybrat souřadnice a do nich přidat nový bod. Existující body v rozsahu mapy bude dále možné prohlížet, nebo na základě výběru editovat a mazat. K přidání mapy bude sloužit druhé okno, které umožní načíst mapu, zadat její název a geografické rozsahy souřadnic.

Controller pomocí třídy, aplikující rozhraní `IControllerMapsCities`, zabezpečí předání požadavků z *View* do *Modelu*. K tomu jsou předepsány metody pro načtení, mazání a vytváření nových map a pro načtení, editaci, vytváření a mazání bodu na mapě. Načítání bodů musí umožňovat načíst body v rozsahu souřadnic načtené mapy.

3.4.8 Model tříd modulu pro správu cest

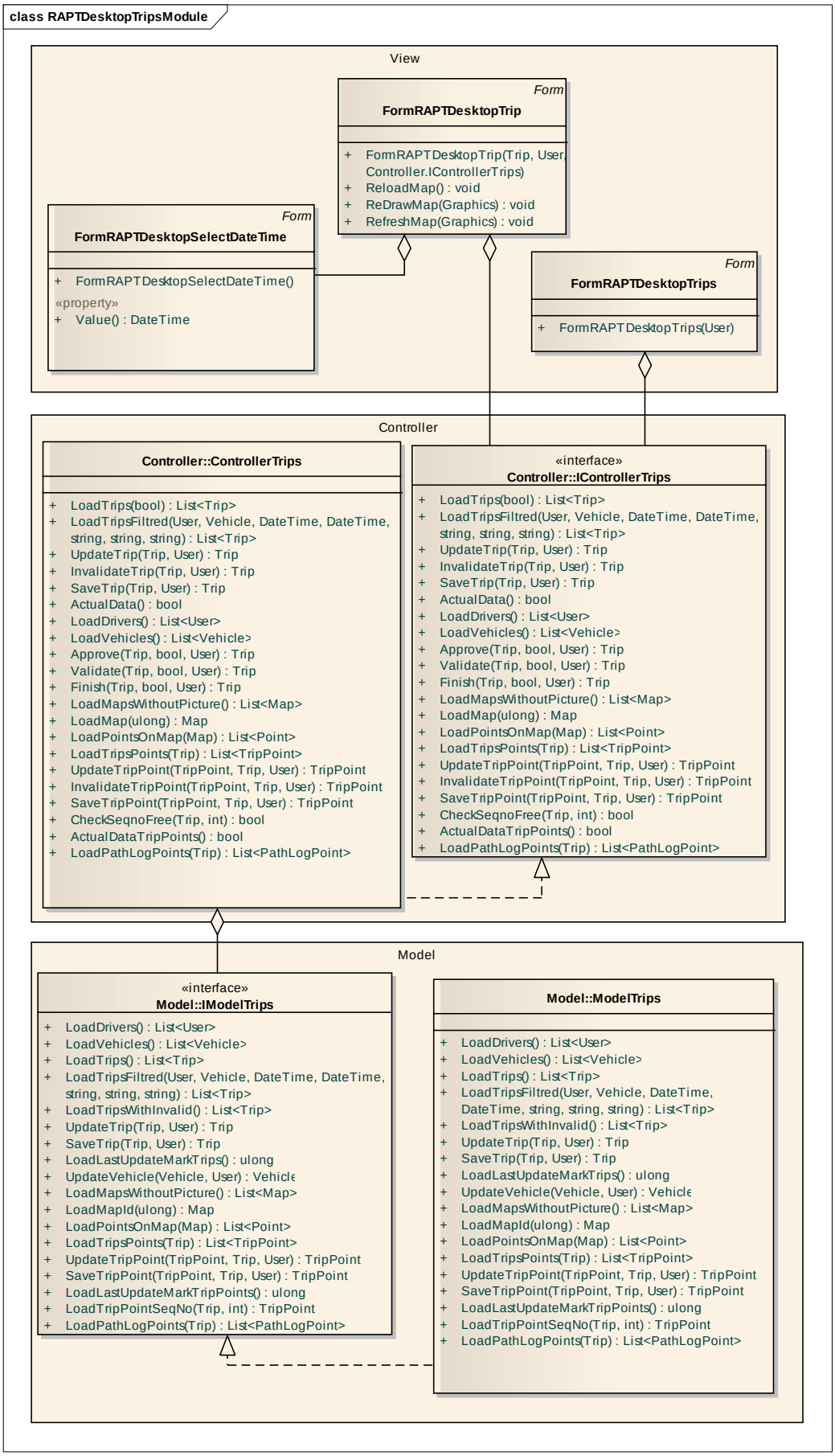
Tento modul se stane nejdůležitější částí aplikace. Bude v něm probíhat kompletní plánování cesty a evidování jejího průběhu. I v tomto modulu bude uplatněn návrhový vzor *Model-View-Controller*. Formuláře části *View* budou tři.

Prvním je hlavní okno. Toto okno bude umožňovat vypisovat cesty. Cest ovšem bude velké množství s různými stavy apod. Pro účely plánovaní pouhý výpis a vyhledávání nebude dostatečný. Proto okno bude schopno filtrovat cesty na základě řidiče, vozidla, plánovaného data a času začátku a konce a podle stavu, ve kterém cesta je. Po výběru umožní formulář cestu smazat, nebo otevřít pro další úpravy.

Třetí formulář bude sloužit pro zadávání data a času v případě ručního zadávání průjezdů naplánovanými body.

Požadovanou funkcionalitu bude zajišťovat třída `ControllerTrips` aplikující rozhraní `IControllerTrips`. To předepisuje, že třída bude mít metody pro načítání cest dle filtru, pro editaci a mazání cest, pro přidávání, editaci a mazání průjezdních bodů, pro načítání map a bodů a pro načítání průběhů cest.

Tyto operace bude *Controller* provádět za pomocí třídy `ModelTrips` aplikující rozhraní `IModelTrips`. Toto rozhraní předepisuje metody pro načítání a aktualizaci dat v databázi pro příslušné třídy.



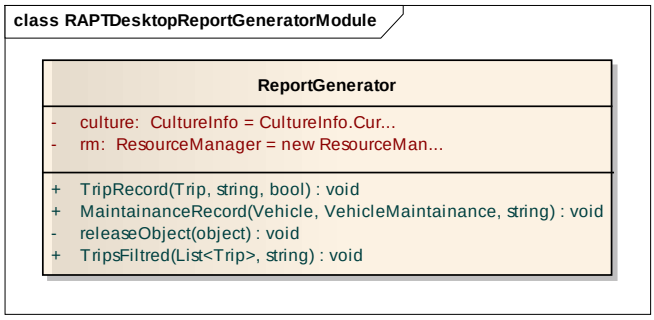
Obrázek 23 Model tříd modulu pro správu cest

3.4.9 Třída generátoru reportů

Tato třída bude sloužit ke generování reportů. Tyto reporty budou generovány na základě šablon do souborů formátu aplikace Microsoft Excel verze 2003. Bude obsahovat metodu pro generování tří reportů:

1. Záznam o cestě (jednoduchý a detailní)
2. Výpis cest dle vybraného filtru
3. Záznam o údržbě vozidla

I tato část bude v samostatném modulu pro možnost pozdější snadné aktualizovatelnosti. Vzhledem k velikosti a jednoduchosti modulu nebude tento aplikovat návrhový vzor *Model-View-Controller*.



Obrázek 24 Třída generátoru reportů

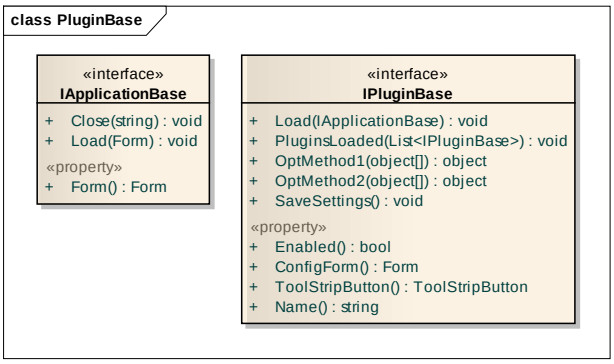
3.4.10 Model rozhraní modulu zásuvných modulů

Pro splnění požadavku na dynamickou rozšiřitelnost aplikace o zásuvné moduly v podobě *dll* knihoven byla navržena rozhraní, která budou muset tyto zásuvné moduly aplikovat.

Třída integrující zásuvné moduly bude muset obsahovat metody pro „násilné" ukončení aplikace a pro inicializaci nadřazeným formulářem.

Integrovaná třída bude muset obsahovat metodu pro inicializaci modulu, metodu pro předání ostatních načtených modulů a metodu pro uložení nastavení modulu. Dále bude muset třída obsahovat vlastnosti, a to konkrétně příznak zapnutí/vypnutí modulu, konfigurační formulář, tlačítko do menu a název. Tyto metody a vlastnosti budou volány integrujícím systémem.

V aplikaci budou muset být zásuvné moduly nahrány do složky *Plugins* ve složce aplikace.

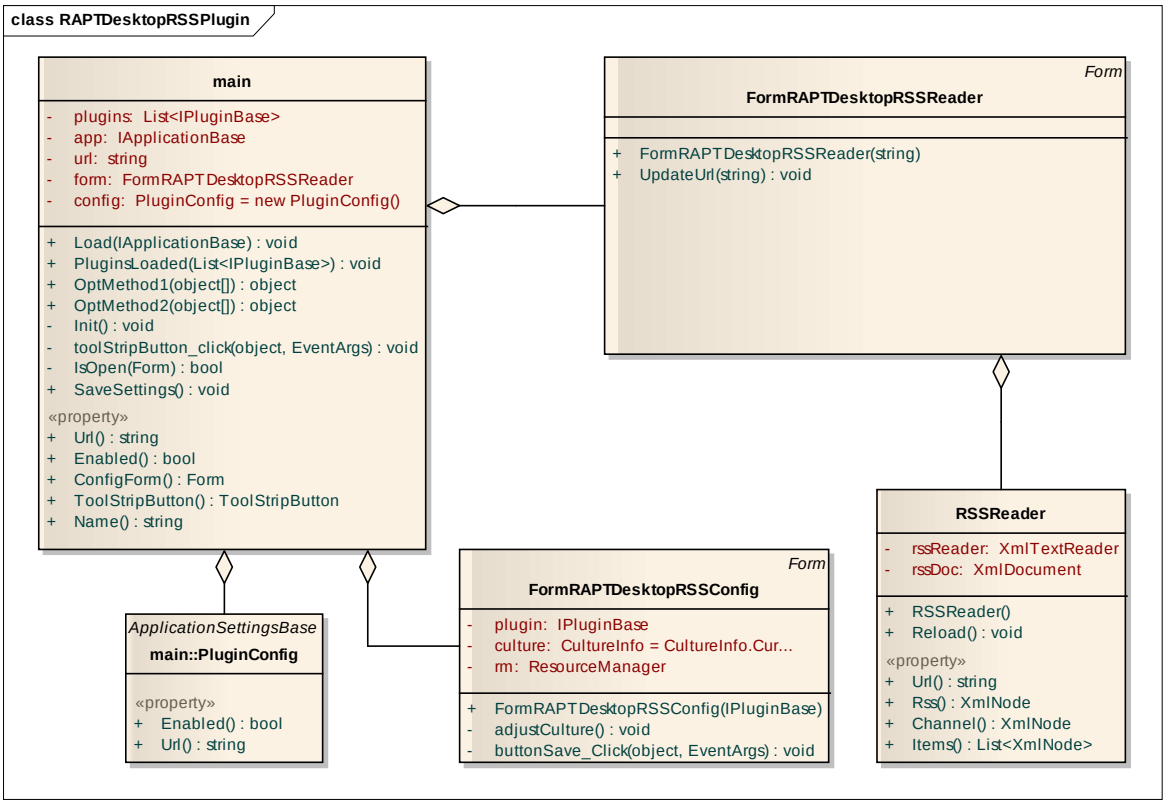


Obrázek 25 Model rozhraní modulu zásuvných modulů

3.4.11 Model tříd zásuvného modulu pro čtení RSS kanálu

Jako ukázkový zásuvný modul byl navržen modul pro čtení kanálu RSS. Tento modul bude umožňovat pomocí formulářového okna zobrazovat zprávy z nastaveného RSS kanálu. Adresu toho kanálu bude možné nastavit přes konfigurační formulář. Modul bude přístupný pomocí tlačítka v hlavním menu. Toto tlačítko bude hlavnímu formuláři aplikace předáno přes povinnou vlastnost zásuvného modulu.

Modul bude schopný ukládat své uživatelské nastavení. Tato nastavení budou obsahovat příznak, zda je modul zapnut a adresu RSS kanálu.

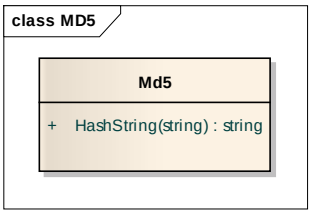


Obrázek 26 Model tříd zásuvného modulu pro čtení RSS

3.4.12 Třída pro šifrování MD5

Vzhledem k širším možnostem využití tohoto šifrování bude třída vytvořena v separátní knihovně.

Tato třída bude využívána pro šifrování hesel.



Obrázek 27 Třída pro šifrování MD5

3.4.13 Model tříd mobilní části systému

Tato část systému bude sloužit řidičům k zaznamenávání údajů o probíhající cestě. Bude využívat stejnou datovou strukturu a stejné databázové připojení jako desktopová část. To nám je umožněno díky technologii .NET. Modul je opět rozdělen dle návrhového vzoru *Model-View-Controller*.

Stěžejní část bude hlavní formulářové okno. Toto okno bude mít dvě rozvržení. Jedno rozvržení pro výběr cest a druhé pro práci s probíhající cestou. Předpokladem je, že aplikace bude spouštěna na dotykovém zařízení a k tomu bude muset být prostředí upraveno.

V průběhu cesty bude hlavní okno zobrazovat informace o stavu GPS modulu, který bude umožňovat vypínat a zapínat. Pro zpřístupnění těchto funkcionalit bude okno implementovat rozhraní `Observer`, pomocí kterého se zaregistruje k přijímání dat z `Observable` objektu – GPS modulu.

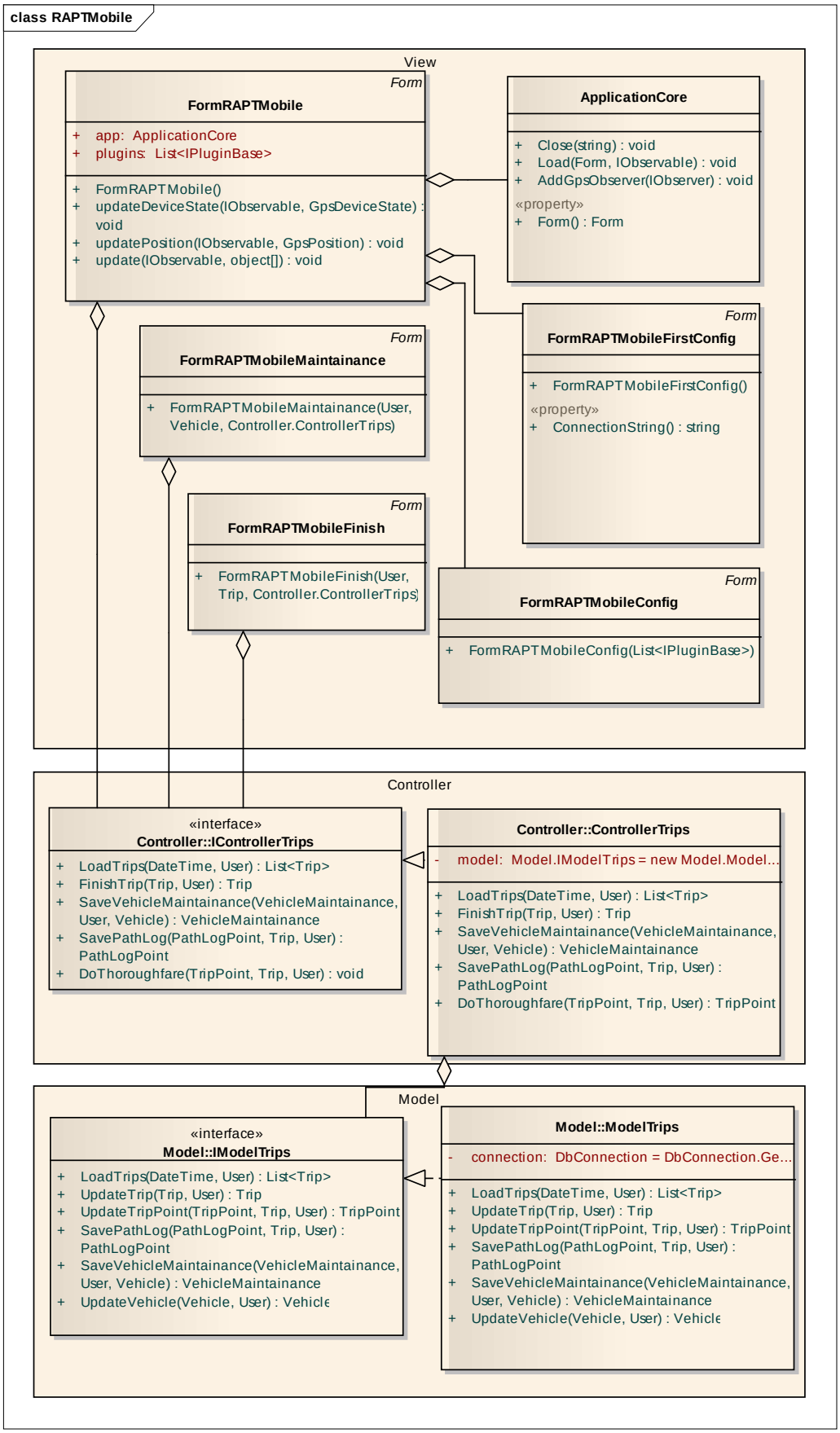
Formulář bude integrovat zásuvné moduly. Těmto modulům bude poskytovat instanci jádra aplikujícího potřebné rozhraní (viz dále) a formulář umožňující volat nastavení jednotlivých modulů.

Další okno mobilní části bude sloužit pro nastavení připojení k databázi při prvním spuštění aplikace. Toto nastavení bude uloženo v uživatelské části registrů operačního systému.

Dále modul bude obsahovat okno pro ukončení cesty a pro zaznamenání údržby, provedené v průběhu cesty.

Práci okolo zaznamenávání průběhu cesty bude plnit *Controller* dle rozhraní `IControllerTrip`. Toto rozhraní předepisuje metody pro načtení cest přihlášeného uživatele, ukládání průjezdů naplánovanými body, zaznamenávání průběhu cesty, zaznamenávání údržeb a pro ukončení cesty. Všechny tyto operace bude zajišťovat *Controller* pomocí *Modelu* dle aplikačního rozhraní `IModelTrips`.

Toto rozhraní bude aplikováno třídou `ModelTrips`, která bude provádět online zaznamenávání na vzdáleném serveru. Podmínkou bude funkční připojení k síti Internet.



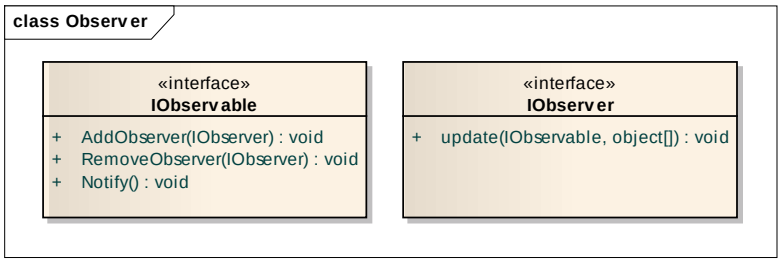
Obrázek 28 Model tříd mobilní části systému

3.4.14 Model rozhraní Observer

Observer bude implementován jako separátní knihovna z důvodu možnosti využití v jiných projektech.

Rozhraní `IObservable` předepisuje třídám implementovat metody pro přidání instancí aplikujících rozhraní `IObserver`, dále metody pro jejich odebírání a metodu pro informování instancí o změně stavu pozorovaného objektu. Předání informací bude prováděno tlačnou metodou.

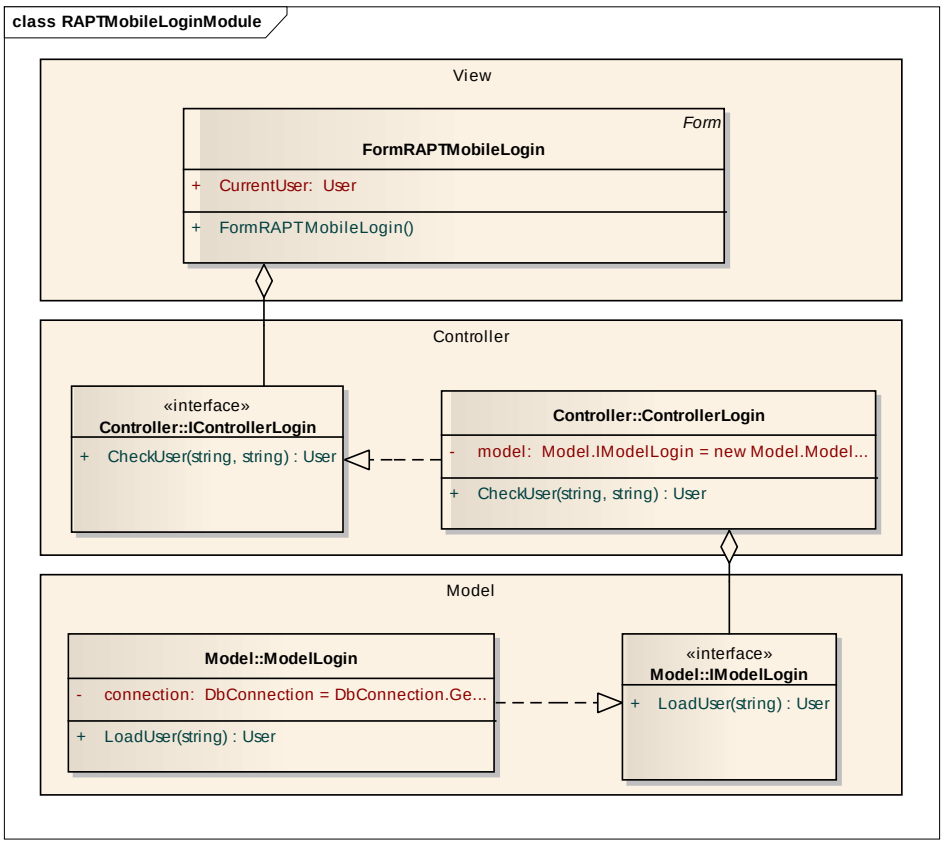
`IObserver` předepisuje třídám implementovat metodu pro příjem těchto upozornění.



Obrázek 29 Model rozhraní Observeru

3.4.15 Model tříd modulu přihlášení do mobilní části aplikace

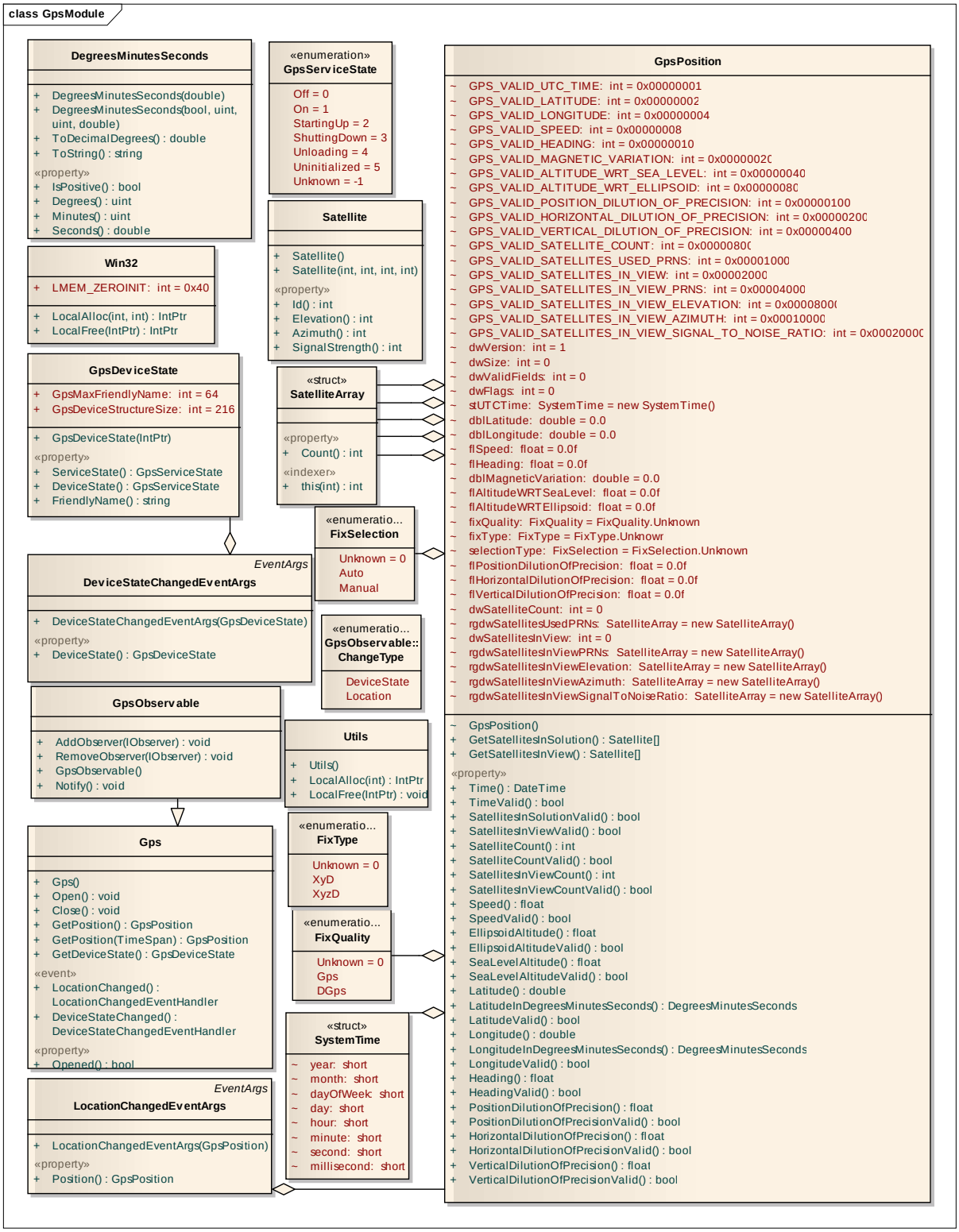
Tento modul pracuje stejně jako stejný modul v desktopové části.



Obrázek 30 Model tříd modulu přihlášení do mobilní části aplikace

3.4.16 Model tříd GPS modulu

Modul GPS bude zpracovávat zprávy poskytované z GPS API [16] a poskytovat je pomocí rozhraní `IObserverable` zaregistrovaným pozorovatelům. Jedná se o rozdělené zprávy o pozici a stavu zařízení a doplňující informace o GPS modulu.

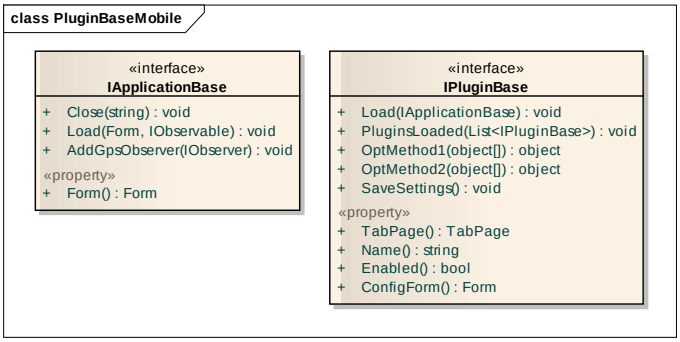


Obrázek 31 Model tříd GPS modulu

3.4.17 Model rozhraní zásuvných modulů mobilní části

Modul rozhraní bude postaven podobným způsobem, jako v části desktopové. Metody rozhraní integrujícího objektu však budou rozšířeny o metodu umožňující zaregistrovat `Observer`. Dále je inicializace objektu doplněna o předání objektu implementujícího rozhraní `IObservable`, konkrétně modulu GPS.

Integrovaný objekt oproti tomu musí navíc implementovat rozhraní IObserver. Další rozdíl bude v předávaném objektu pro přístup uživatele k modulu. Tím nebude tlačítko menu, ale záložka.



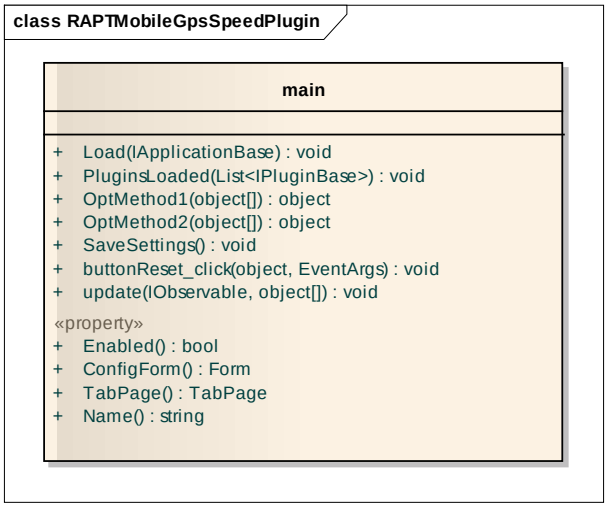
Obrázek 32 Model rozhraní zásuvných modulů mobilní části

3.4.18 Model tříd zásuvného modulu pro čtení RSS kanálu

Tento modul bude pracovat stejně jako stejný modul v desktopové části systému. V případě dostupného přístupu k síti internet umožní uživateli zobrazovat informace z RSS kanálu, nastaveného v konfiguračním formuláři.

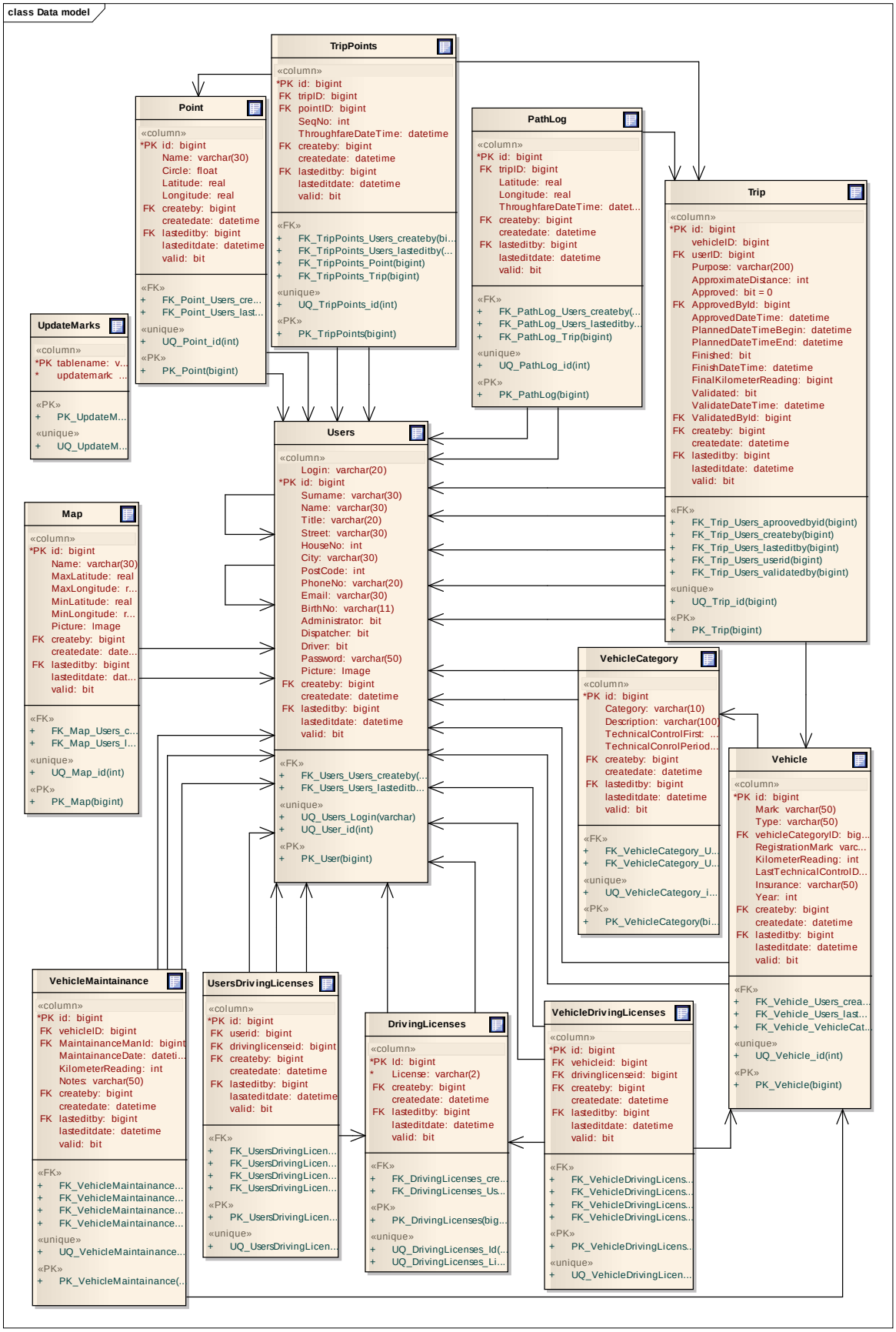
3.4.19 Model tříd zásuvného modulu pro zobrazení rychlosti

V tomto modulu bude na základě rozdílů GPS souřadnic řidiči zobrazována aktuální rychlost jízdy a celková průměrná rychlost.



Obrázek 33 Model tříd zásuvného modulu pro zobrazení rychlosti

3.5 Návrh modelu databáze



Obrázek 34 Model databáze

Pro uchování datové struktury systému bylo zapotřebí vytvořit databázový model, který vychází z modelu základních tříd. Názvy jednotlivých tabulek odpovídají názvům tříd, jejichž informace má daná tabulka obsahovat.

V systému je potřeba řešit platnost načtených údajů. K těmto účelům bude sloužit tabulka *UpdateMarks*, ve které budou uloženy unikátní čísla posledních změn pro každou tabulku.

Pro zachování vztahů v případě odstraňování údajů byl zvolen přístup, kdy z databáze nebudou údaje fyzicky mazány, ale všechny údaje budou doplněny o příznak, zda jsou platné či nikoliv. V budoucnu však bude potřeba tato data postupně archivovat.

Kompletní diagram databázového modelu je v Příloha F.

4 Řešení

Na základě návrhu řešení byla provedena implementace, jejímž výsledkem je systém, který dle požadavků obsahuje dvě části. Desktopovou část a mobilní část. Aplikace byla vytvořena jako vícejazyčná, kdy jazyk je vybrán dle regionálního nastavení operačního systému.

V následujících odstavcích bude popsán systém popořadě tak, jak se s ním bude postupně při používání setkávat uživatel.

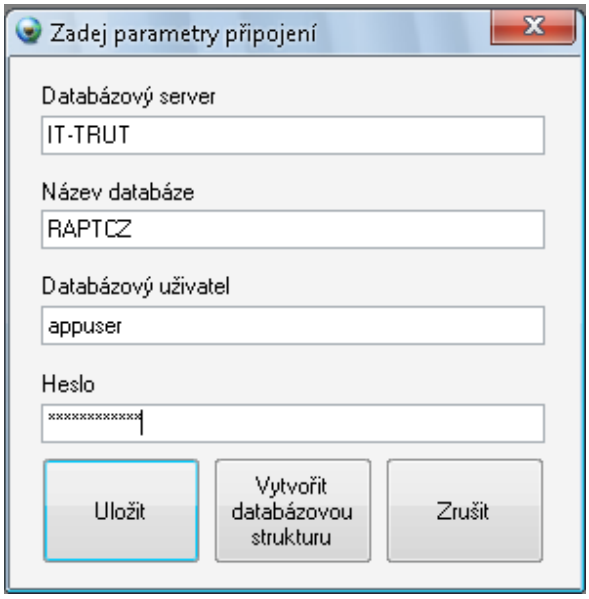
4.1 Desktopová část

Tato část bude sloužit zejména administrátorům pro správu systému, dispečerům pro plánování cest a okrajově řidičům pro dodávání doplňujících informací.

Desktopová aplikace dostala název RAPT Desktop (***R**egistration **A**nd **P**lanning **T**rips on **Desktop***).

4.1.1 Přihlášení do systému

V případě prvního spuštění aplikace se jako první okno zobrazí uživateli okno pro zadání přístupových údajů k databázi.



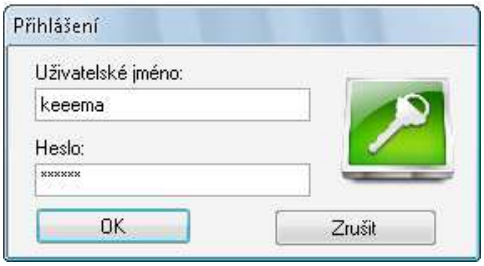
Obrázek 35 Okno pro zadání parametrů připojení

Na tomto okně je nutné zadat adresu databázového serveru, název databáze, databázové přihlašovací jméno a heslo pro přístup k databázi. Všechny údaje jsou povinné. Informace jsou v případě správného zadání uloženy do uživatelského nastavení uživatele přihlášeného do operačního systému.

Okno dále umožňuje vytvoření databázové struktury v databázi. Toto se dá využít v případě zavádění systému. V takovém případě je také vytvořen administrátorský uživatel „admin“ s heslem „admin“ pro první přístup do systému, při kterém je možné nastavit skutečného administrátora.

Po uložení informací se zobrazí okno pro přihlášení do systému. V tomto okně je nutné zadat uživatelské jméno a přístupové heslo. Oba údaje jsou povinné.

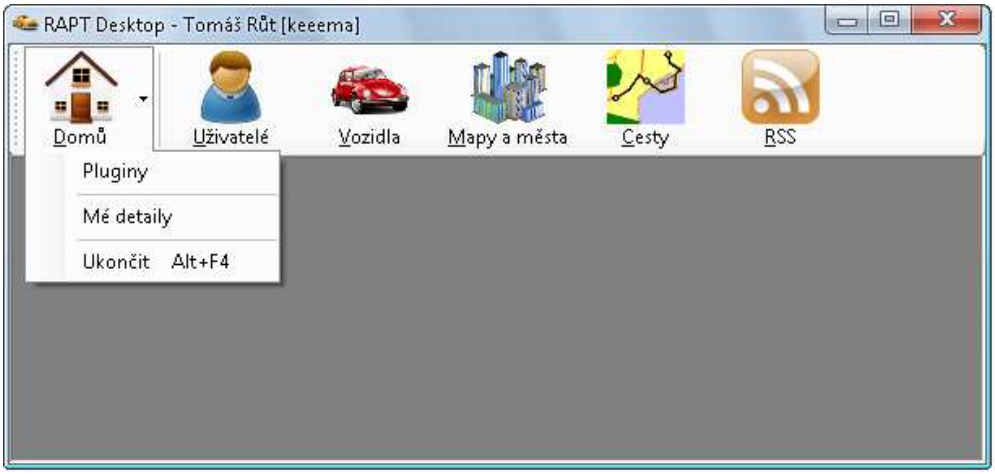
Po potvrzení tlačítkem *OK* je provedeno ověření uživatele v databázi. V případě špatného zadání je uživatel upozorněn na chybu a je mu umožněn další pokus o přístup. V případě správného zadání je okno uzavřeno a uživateli se zobrazí hlavní okno aplikace.



Obrázek 36 Okno přihlášení do systému

4.1.2 Hlavní okno

Hlavní okno aplikace je rozděleno na dvě části. Těmi jsou hlavní menu a prostor pro jednotlivé moduly (okna potomků).



Obrázek 37 Hlavní okno

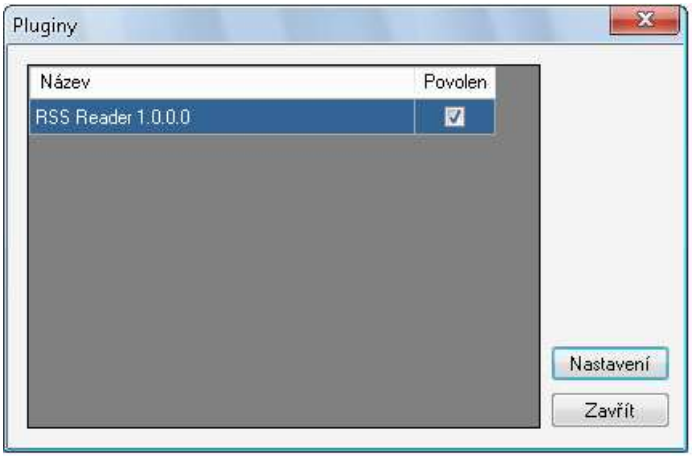
V hlavičce okna je vypsán název aplikace, jméno uživatele a jeho přihlašovací jméno. Hlavní menu obsahuje tlačítko *Domů*, tlačítka jednotlivých modulů systému a tlačítka zásuvných modulů.

Po otevření menu tlačítka *Domů* můžeme nastavovat zásuvné moduly, zobrazit vlastní detaily a ukončit aplikaci.

Okno si po vypnutí zapamatuje své rozměry, pozici a stav.

4.1.3 Okno nastavení zásuvných modulů

Okno nastavení zásuvných modulů umožňuje uživateli pomocí zaškrtávacího políčka vypnout nebo zapnout jednotlivé zásuvné moduly, vypsané v seznamu. Dále, pokud to zásuvný modul umožňuje, může uživatel otevřít jeho nastavení.



Obrázek 38 Okno pro nastavení zásuvných modulů

4.1.4 Okno správy uživatelů

Tento modul se zobrazí po stisknutí tlačítka *Uživatelé*, a to buď myší, přechodem na tlačítko pomocí šipek nebo klávesovou kombinací.

Vzhledem k citlivosti osobních údajů je správa uživatelů přístupná pouze administrátorům. Ostatní uživatelé tlačítko správy vůbec nemají k dispozici.

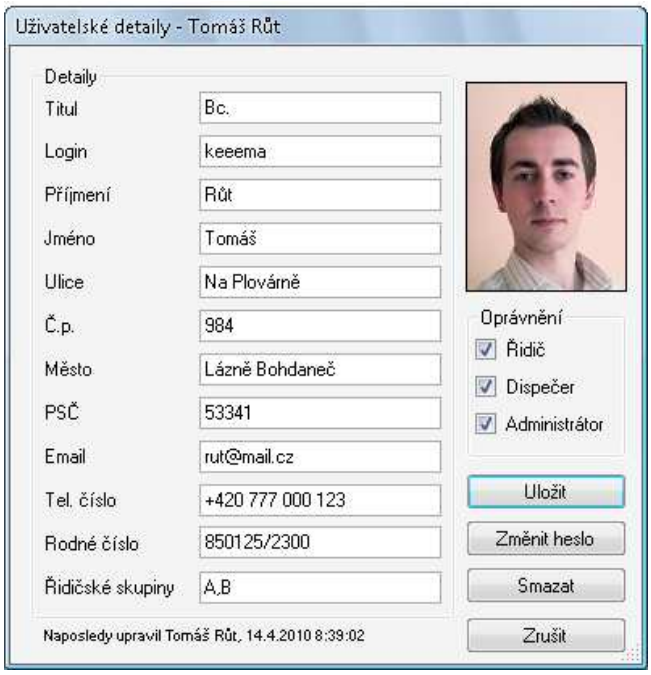
Obrázek 39 Okno správy uživatelů

Okno správy uživatelů se skládá z menu po levé straně okna, a z tabulky uživatelů. Tato tabulka nám zobrazuje všechny informace o uživatelích (id, obrázek, příjmení, jméno, titul, ulici, č. p., město, PSČ, email, telefonní číslo, rodné číslo, řidičské skupiny a příznaky oprávnění). Tabulku je možné řadit podle sloupce kliknutím myši na hlavičku sloupce. Uživatele je možné editovat kliknutím na konkrétní řádek. Menu obsahuje tlačítka pro přidání, úpravu, mazání a hledání uživatele a tlačítko pro znovunačtení dat z databáze.

4.1.5 Okno uživatele

Po otevření vlastních detailů, případně po otevření detailů jiného uživatele administrátorem ve správě uživatelů, se zobrazí okno s detaily uživatele.

V okně je možné změnit jednotlivé informace. To znamená obrázek, příjmení, jméno, titul, ulici, č. p., město, PSČ, email, telefonní číslo, rodné číslo, řidičské skupiny a heslo. Obrázek se dá změnit poklepáním myši. Po té se zobrazí standardní dialogové okno pro výběr obrázku. Tyto informace může uživatel měnit v případě, že je administrátor nebo se jedná o jeho vlastní údaje.



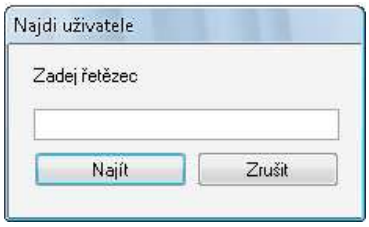
Obrázek 40 Okno uživatele

Smazání uživatele a změna jeho oprávnění je umožněna pouze administrátorovi. V případě, že upravující uživatel nemá oprávnění k některým změnám, nemá tyto položky vůbec dostupné.

V okně je také zobrazen údaj, kdo a kdy naposledy uživatele editoval. Toto slouží pro sledování změn.

4.1.6 Okno vyhledávání

Toto okno je univerzální a je použito pro vyhledávání ve všech modulech. Vyhledávání probíhá vždy od aktuální pozice v tabulce směrem dolů. Hledaná hodnota zůstává uložena v paměti a při opětovném pokusu o hledání je automaticky opět vyplněna. Výskyt hodnoty se hledá ve všech hodnotách řádku.



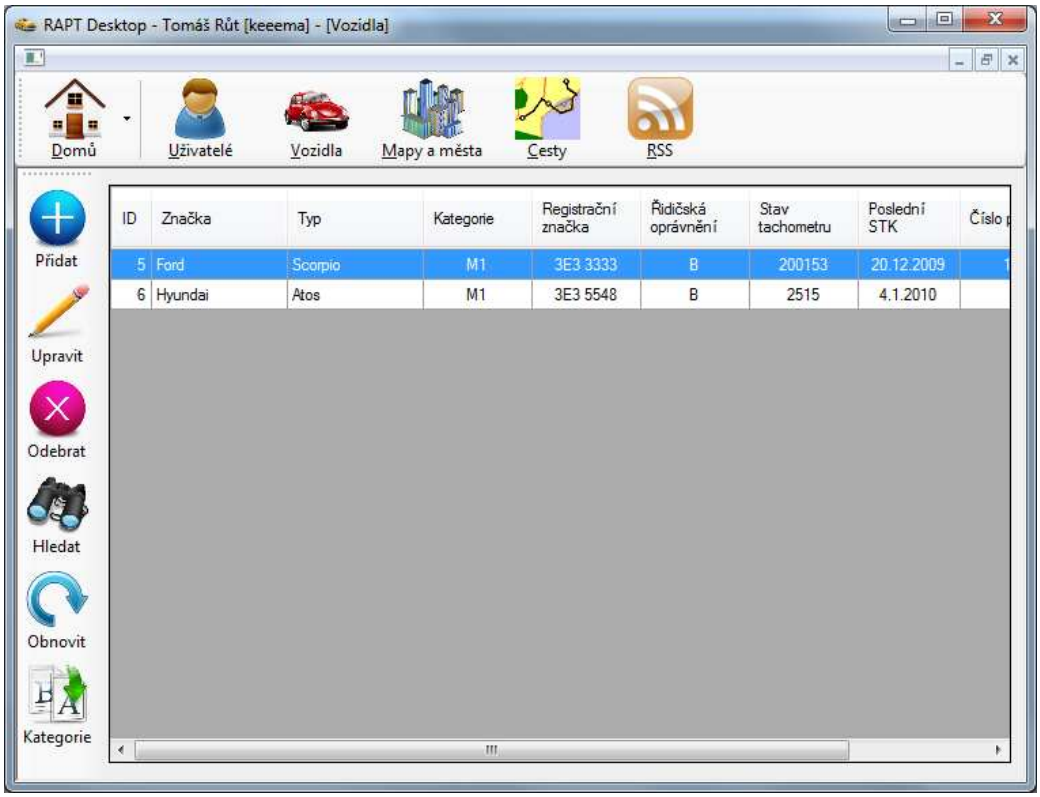
Obrázek 41 Okno vyhledávání

4.1.7 Okno správy vozidel

Tento modul je možné otevřít tlačítkem *Vozidla*. Vzhled je koncipován stejně jako u správy modulů, čili na levé straně menu a na pravé straně tabulka s údaji. Modul je přístupný všem uživatelům. Dispečer a řidič ovšem mají určitá omezení.

Tabulka obsahuje údaje vozidla (id, značku, typ, kategorii, registrační značku, potřebná řidičská oprávnění, stav tachometru, datum poslední STK, číslo pojištění a rok první registrace). Tabulka se dá opět řadit dle jednotlivých sloupců kliknutím na jejich hlavičku.

Menu umožňuje přidat (pouze administrátorovi), upravit, smazat (pouze administrátorovi) a hledat vozidla, obnovit aktuální načtená data a spravovat kategorie vozidel.



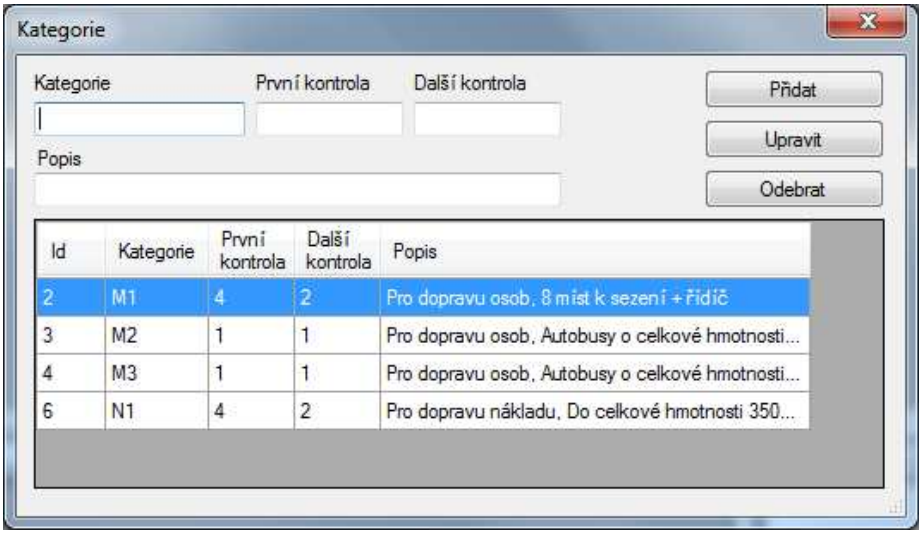
Obrázek 42 Okno správy vozidel

4.1.8 Okno kategorií vozidel

Po stisknutí tlačítka *Kategorie* v menu správy vozidel se zobrazí okno kategorií. Tento nástroj má k dispozici pouze administrátor a umožňuje přidávat, upravovat a mazat jednotlivé kategorie vozidel. Okno se skládá z editační části a z tabulky.

Tabulka vypisuje id, kategorii, interval první technické kontroly, interval následných kontrol a popis kategorie. Kliknutím na konkrétní řádek se předvyplní textová pole editační části a je možné daný řádek upravit.

Tato část byla vytvořena pro použití do budoucna, kdy například pomocí nějakého zásuvného modulu bude možné systém doplnit o hlídání termínu prohlídek STK jednotlivých vozidel atd.



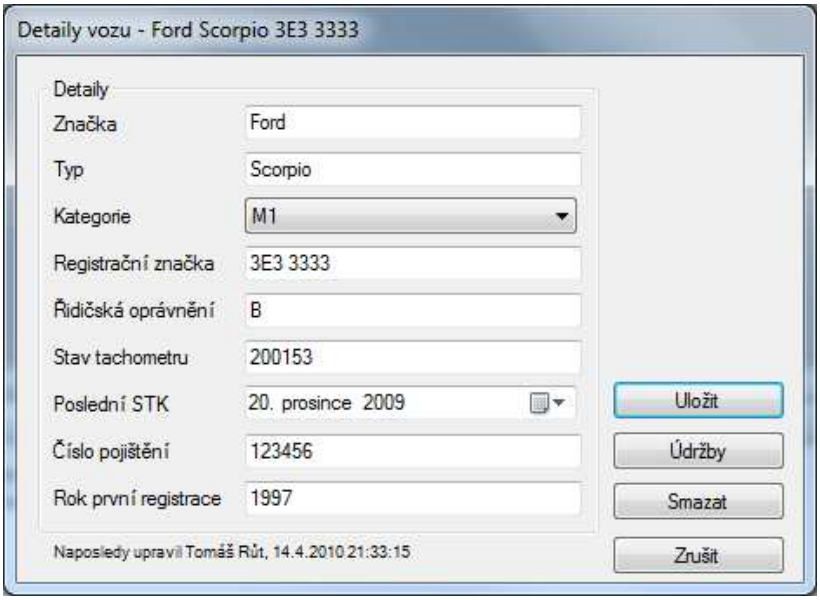
Obrázek 43 Okno kategorií vozidel

4.1.9 Okno vozidla

Po otevření detailů vozidla se zobrazí okno s detaily vozu. Administrátor může využívat veškeré funkce okna, dispečer nemá povoleno vůz smazat a řidič má právo pouze přidávat údržby vozidla.

K vozidlu lze uložit značku, typ, kategorii, registrační značku, potřebná řidičská oprávnění, stav tachometru, datum poslední STK, číslo pojištění, rok první registrace a údržby.

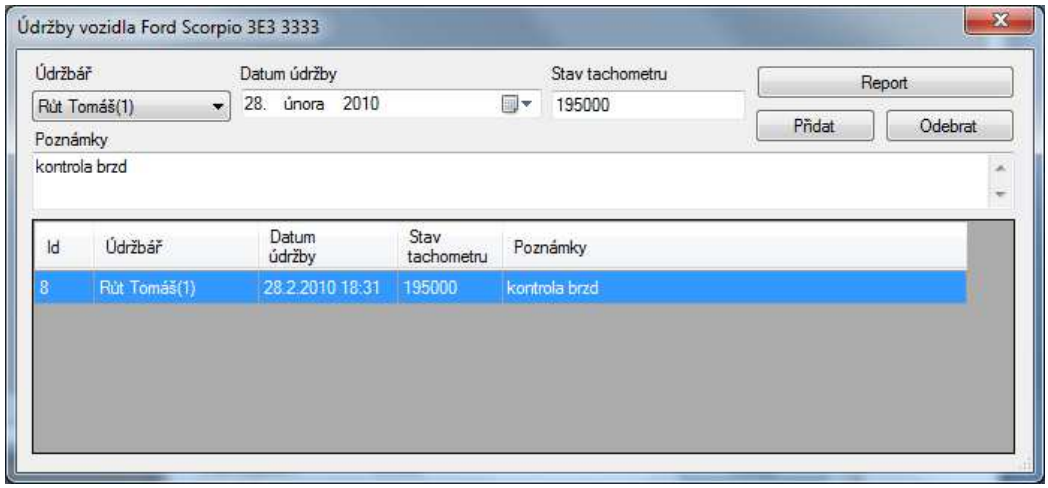
Okno stejně jako okno uživatele zobrazuje pro sledování historie změn datum a uživatele, který naposledy záznam měnil.



Obrázek 44 Okno vozidla

4.1.10 Okno údržeb vozidla

Po stisknutí tlačítka *Údržby* v okně vozidla se zobrazí okno s jednotlivými údržbami vozidla.



Obrázek 45 Okno údržeb vozidla

Okno je rozděleno na tabulku a editační část. Záznam údržby se skládá z údržbáře, data údržby, stavu tachometru a poznámek o provedených úkonech. Administrátor a dispečer mohou záznamy přidávat a odebírat. Řidič může pouze přidávat záznamy, a to pouze se svým jménem. Okno dále umožňuje vygenerovat „Záznam o údržbě“ v podobě dokumentu aplikace Microsoft Excel (Příloha B).

4.1.11 Okno správy map a měst

Stisknutím tlačítka *Mapy a města* v hlavním okně se zobrazí okno modulu správy cest. Toto okno umožňuje dispečerům a administrátorům spravovat mapové podklady a města určená pro plánování. Řidiči nemají k tomuto modulu přístup.

Okno je rozděleno na čtyři části. První část okna slouží pro výběr mapového podkladu, případně jeho přidání nebo smazání. Mazání map není dispečerům povoleno. Pro přidání mapy je vytvořeno okno, které bude popsáno v další kapitole.

Druhá část okna slouží pro zobrazení mapy. Tato část zobrazuje mapu v rozlišení obrázku mapy. Pro posun na mapě je tato část okna vybavena posuvníky.

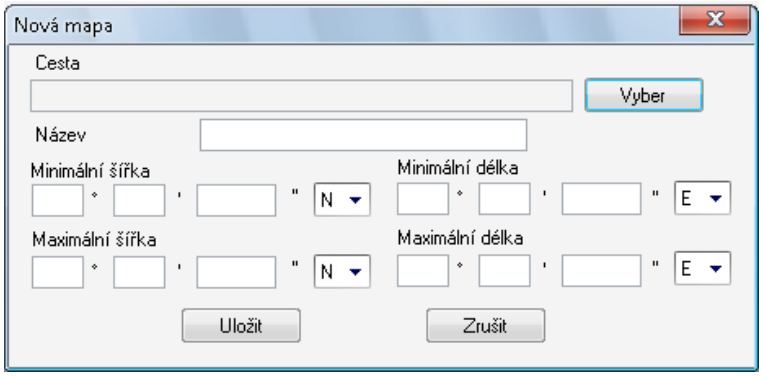
Třetí část okna je tvořena tabulkou. Tato tabulka vypisuje obce, které se nacházejí v rozsahu mapy. Ve výpisu jsou hodnoty název, zeměpisná šířka, zeměpisná délka a okruh obce kolem souřadnic. Vybráním obce v tabulce dojde k vyplnění editačních polí části čtvrté a k vykreslení obce na mapě. Červený bod označuje souřadnice obce. Kolem tohoto bodu je pak také zvýrazněn okruh okolo souřadnic obce.

Čtvrtá část se skládá z editačních polí pro přidávání, úpravy nebo mazání obcí. Zeměpisné souřadnice je možné do políček automaticky vyplnit kliknutím myši na pozici na mapovém podkladu.

Obrázek 46 Okno správy map a měst

4.1.12 Okno pro přidání mapy

Po stisknutí tlačítka *Přidat* ve správě map a měst, části *Mapy*, se zobrazí okno pro přidání mapového podkladu.



Obrázek 47 Okno pro přidání mapy

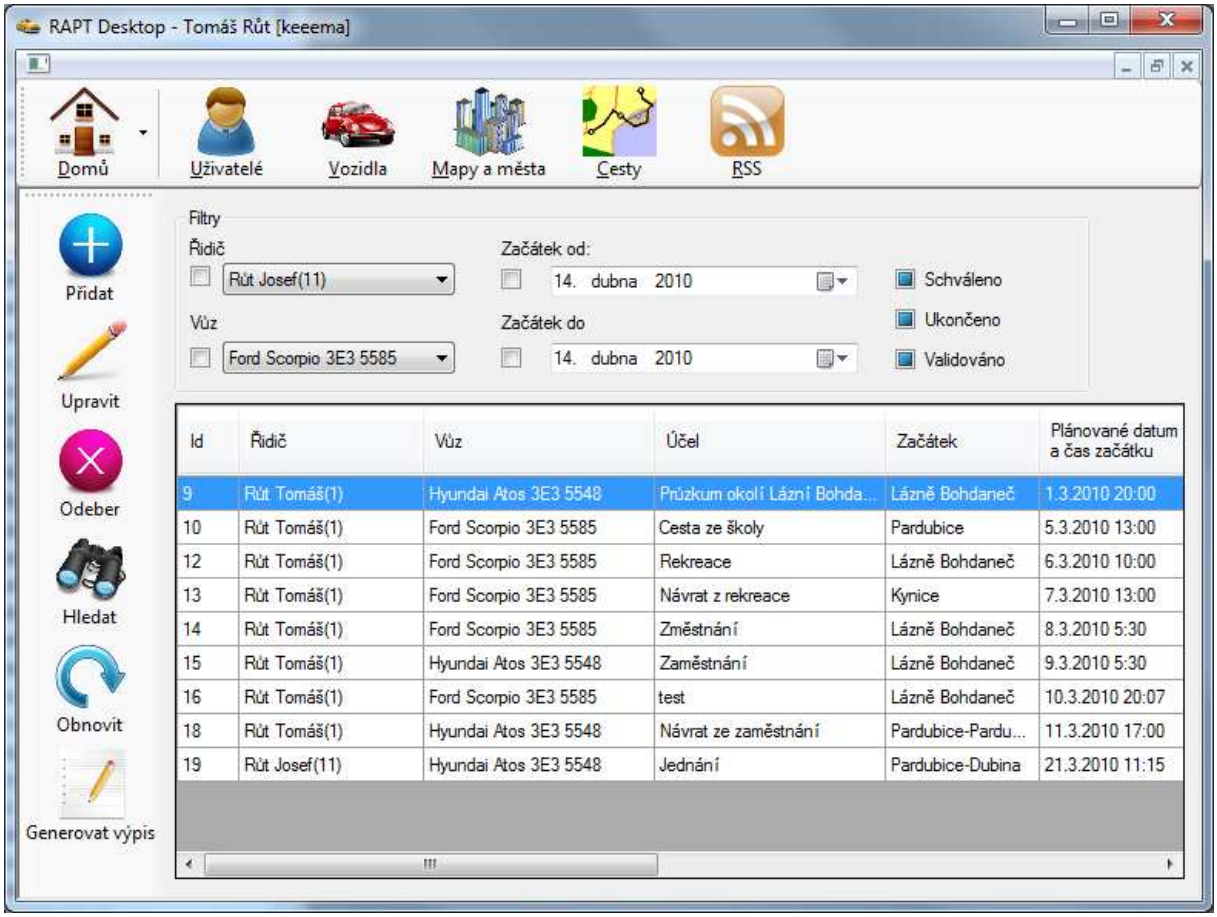
Toto okno nám umožňuje pomocí dialogového okna vybrat soubor mapového podkladu. Tato mapa musí být pojmenována a musí být určen její rozsah pro následné výpočty. Rozsah se zadává souřadnicemi jihozápadního a severovýchodního rohu mapy.

4.1.13 Okno správy cest

Stisknutím tlačítka *Cesty* v hlavním okně aplikace se zobrazí okno správy cest. Toto okno je rozděleno na tři části. Na levé straně menu, vpravo nahoře filtr výpisu a vpravo dole tabulka pro výpis cest.

Filtr umožňuje uživateli omezit výpis pouze na požadované záznamy. Cesty se dají filtrovat dle řidiče, vozidla, plánovaného data začátku cesty, plánovaného data ukončení cesty a podle stavu cesty. Výpis je prováděn do tabulky v dolní části okna. V této tabulce je možné řadit řádky podle sloupců kliknutím na jejich hlavičku.

Menu obsahuje tlačítka pro přidávání, úpravu, mazání a hledání cest. Dále tlačítko pro obnovení načtených dat a tlačítko pro generování „výpisu cest“. Tento výpis (Příloha C) je vygenerován do dokumentu typu Microsoft Excel a odpovídá aktuálně zobrazeným cestám.



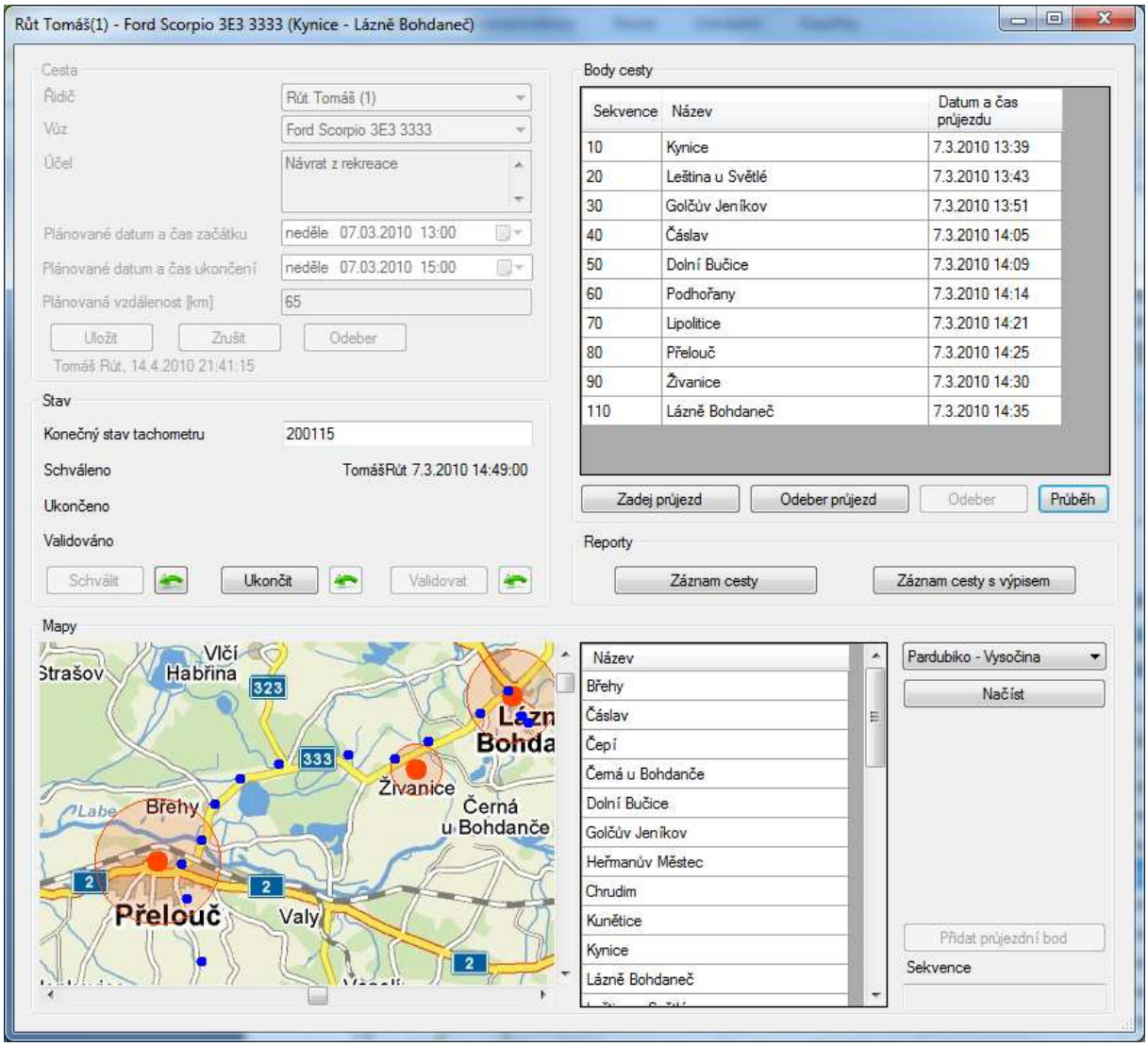
Obrázek 48 Okno správy cest

4.1.14 Okno cesty

Plánovaná cesta je vytvářena v okně detailu cesty. V tomto okně má nejprve dispečer pouze možnost zadat základní informace o cestě (řidič, vozidlo, účel cesty, plánovaný datum a čas začátku a konce cesty, plánovaná vzdálenost). Teprve po jejich uložení se zpřístupní zbývající funkcionalita okna.

V této fázi může dispečer ve spodní části okna vybrat mapu a z ní postupně vybírat body do plánované cesty. Bod se přidává výběrem z tabulky, zadáním pořadového čísla bodu na cestě a stisknutím tlačítka *Přidat průjezdní bod*. Tím se bod přidá do cesty a zařadí do tabulky v pravé horní části okna. Po kompletním naplánování cesty může dispečer cestu schválit. Tím se cesta zpřístupní řidiči, který může cestu provést.

Během probíhající cesty může dispečer v tomto okně při zobrazené mapě sledovat průběh cesty. Ta se mu po stisknutí tlačítka *Průběh* zobrazí na mapě. Vykreslení záznamu cesty je realizováno modrými tečkami. Tyto tečky představují souřadnice, kterými řidič projížděl. Dále jsou zobrazeny již projeté (červeně) a neprojeté (zeleně) průjezdní body. Dále jsou zvýrazněny body při výběru obce z některé z tabulek.



Obrázek 49 Okno cesty

Dispečer a řidič mohou cestu v této části aplikace ukončit. Dispečer navíc může manuálně zadat časy průjezdů naplánovanými body, nebo je naopak odebírat.

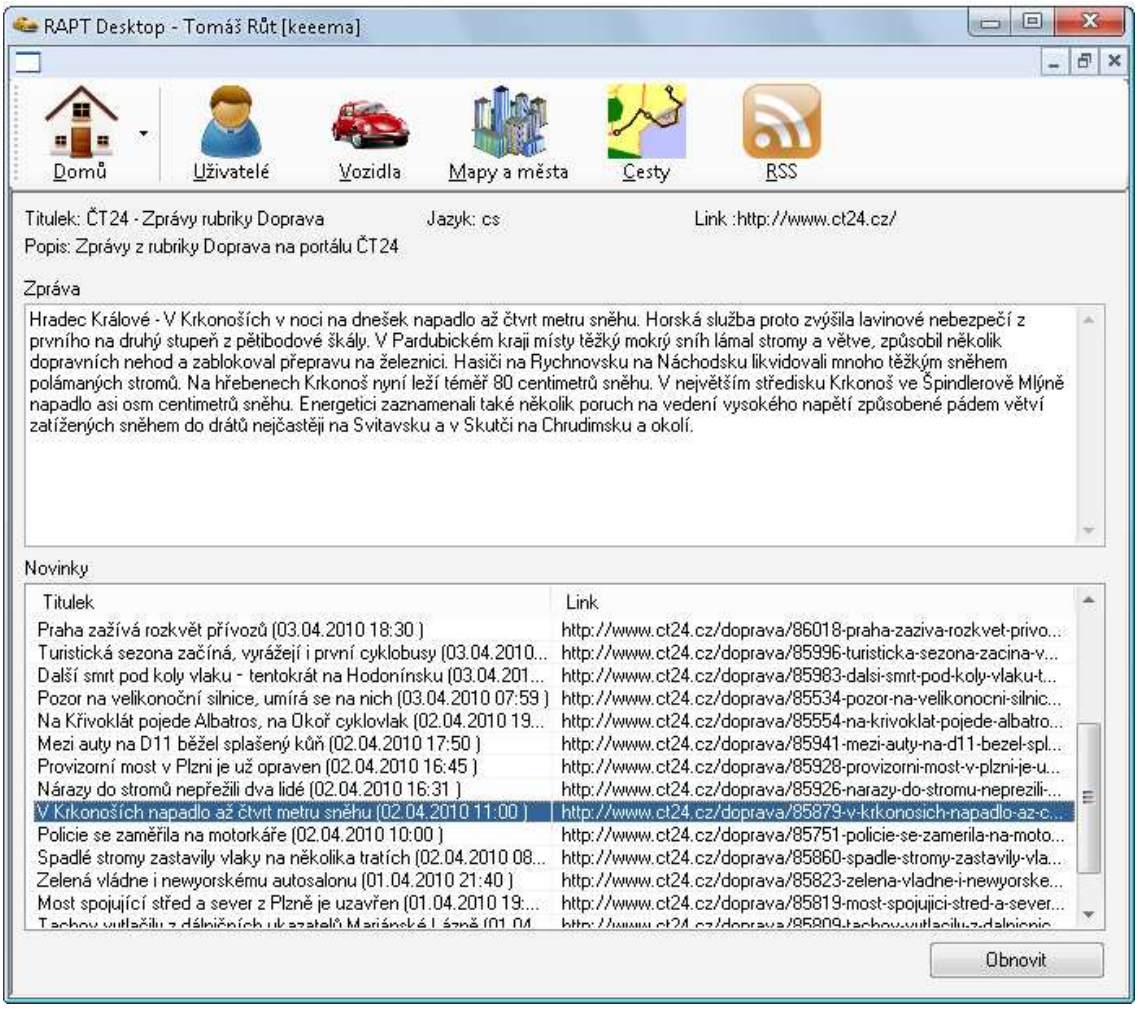
Po ukončení cesty je ještě zapotřebí cestu dispečerem validovat, zda proběhla správně, případně doplnit průjezdy. Po validaci již není možné s cestou dále pracovat, je možné pouze prohlížet její průběh a generovat reporty. Okno umožňuje generovat „Záznam o cestě“ (Příloha D) a „Záznam o cestě s výpisem“ (Příloha E).

Administrátor má stejné možnosti jako dispečer, navíc však má možnost vracet stavy cesty.

4.1.15 Okno zásuvného modulu čtečky RSS kanálu

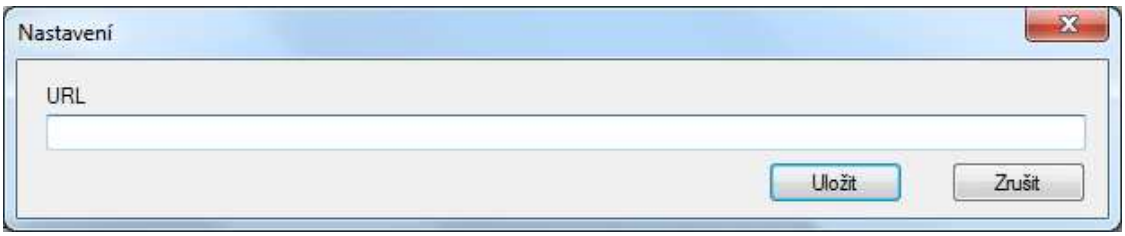
Tento zásuvný modul umožňuje čtení nastaveného RSS kanálu, a to po nahrání *dll* souboru modulu do složky *Plugins*. V aplikaci je přístupný pomocí tlačítka menu hlavního okna.

Po stisknutí tlačítka *Obnovit* čtečka přečte a zobrazí aktuální informace z nastaveného RSS kanálu.



Obrázek 50 Okno čtečky RSS kanálu

Adresu RSS kanálu je možné nastavit pomoci konfiguračního okna dostupného z menu *Domů→Pluginy→Nastavení*.



Obrázek 51 Okno nastavení čtečky RSS kanálu

4.2 Mobilní část

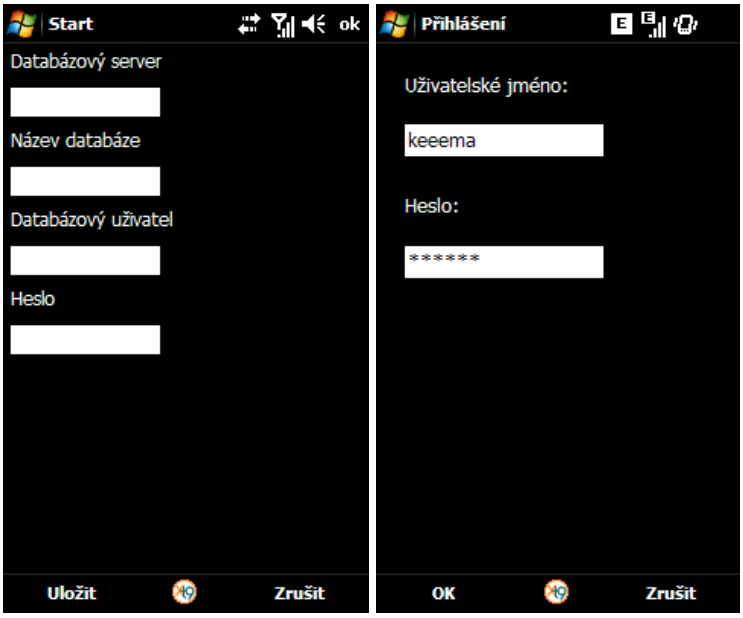
Tato část bude sloužit řidičům pro automatický i manuální záznam cesty pomocí zařízení se systémem Windows Mobile a modulem GPS.

Mobilní aplikace dostala název RAPT Mobile (***R**egistration **A**nd **P**lanning **T**rips on **Mobile***).

4.2.1 Přihlášení do systému

Po prvním spuštění se jako první zobrazí okno pro zadání informací o připojení k databázi. Na tomto okně je nutné zadat adresu databázového serveru, název databáze, databázové přihlašovací jméno a heslo pro přístup k databázi. Údaje jsou povinné. Informace jsou v případě správného zadání uloženy do uživatelské části registrů operačního systému.

Dalším oknem je okno pro zadání přihlašovacího jména a hesla. Obě hodnoty musí být zadány a po jejich potvrzení je provedeno ověření uživatele. V případě špatného zadání může uživatel zadat informace znovu. V opačném případě aplikace postoupí dál do hlavního okna.



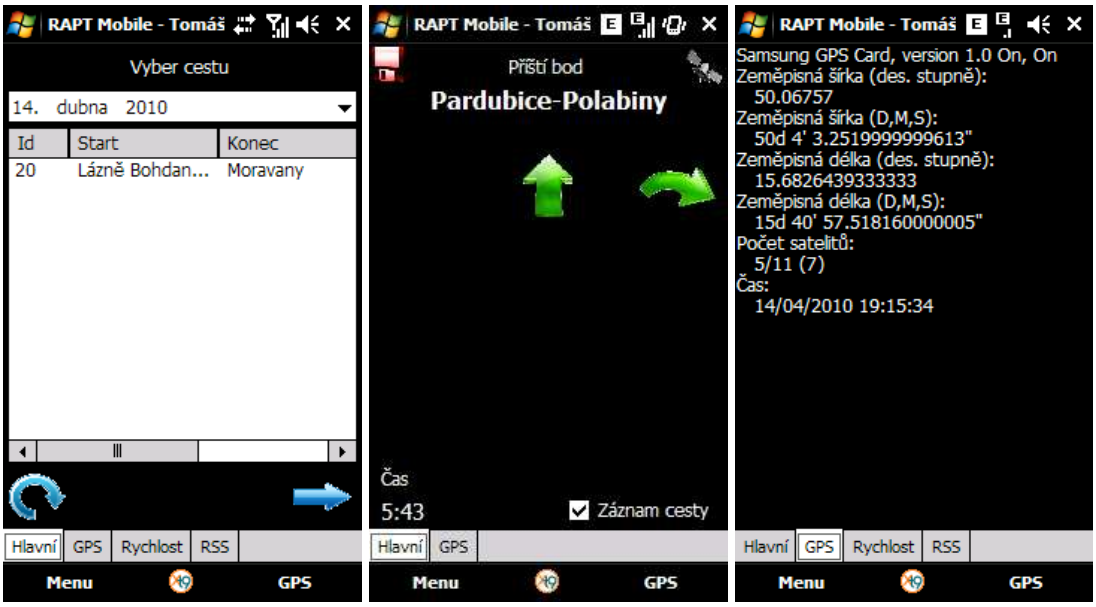
Obrázek 52 Okno pro nastavení databáze a přihlašovací okno

Toto okno spočívá v hlavním menu u spodního okraje obrazovky a z panelu záložek. Základní část aplikace obsahuje hlavní záložku a záložku informací z GPS modulu. Další záložky jsou již záložkami zásuvných modulů.

Hlavní záložka má dvě rozvržení. První slouží pro výběr cesty a druhé pro průběh cesty.

První rozvržení obsahuje rozbalovací kalendář pro výběr data, tlačítko pro načtení dat, tabulku s načtenými cestami a tlačítko pro výběr cesty. Cesty jsou načteny pouze pro přihlášeného uživatele a pro vybraný den.

Po vybrání cesty je zobrazeno druhé rozvržení záložky. Na tomto rozvržení je zobrazen následující bod na cestě, šipky doprava a doleva pro posun v seznamu dosud neprojetých bodů na cestě, šipka nahoru pro manuální zadání průjezdu aktuálním bodem, čas a zaškrtávací políčko pro zapnutí/vypnutí automatického záznamu průběhu cesty.



Obrázek 53 První a druhé rozvržení hlavní záložky a záložka GPS informací

Pokud uživatel v menu *GPS* spustí modul GPS, a ten má dostupné informace o aktuální pozici, probíhá zadávání průjezdů naplánovanými body automaticky. Zaznamenání průjezdu je doprovázeno zvukovým upozorněním.

Informace o signálu GPS je na hlavní záložce zobrazována v pravém horním rohu ikonkou satelitu. Pokud není signál dostupný, je možné zadat průjezdy ručně šipkou nahoru.

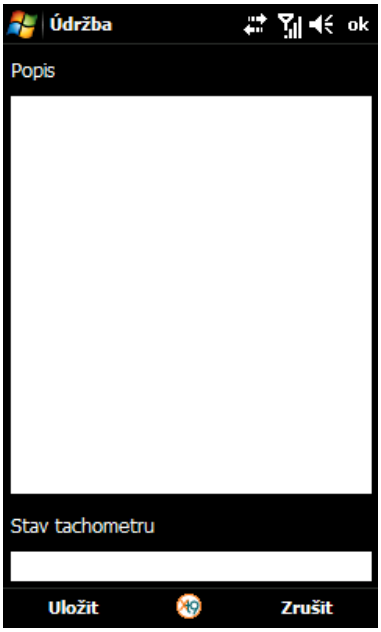
Dále v případě dostatečného signálu GPS a zaškrtnutého políčka pro automatický záznam probíhá automatické zaznamenávání polohy vozidla v intervalu 1 minuta.

V průběhu ukládání informací na vzdálený server se zobrazí v levém horním rohu červená ikonka diskety. Pokud při ukládání nastane chyba, je vydáno výstražné zvukové znamení a ikona diskety je změněna na ikonu vykřičníku.

V případě průjezdu naplánovaným bodem a uložení bodu jako projetého aplikace automaticky přesune ukazatel na obrazovce na další bod. Pokud jsou všechny body projeté, šipky z obrazovky zmizí.

4.2.2 Okno zaznamenání údržby na cestě

Hlavní menu aplikace dále umožňuje zaznamenat informaci o proběhlé údržbě na cestě. To je možné pomocí okna dostupného pomocí položky *Provést údržbu*. V tomto okně je možné zadat stav tachometru a prováděné úkony.

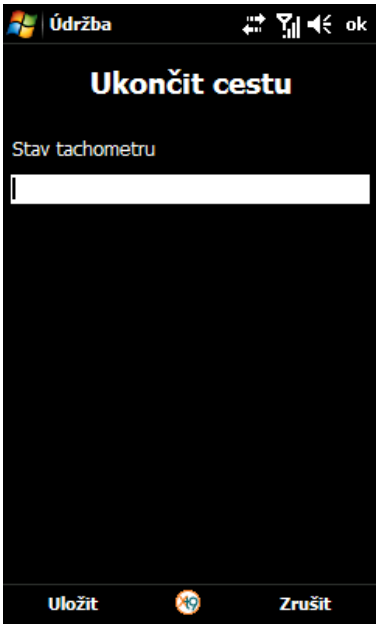


Obrázek 54 Okno pro záznam údržby

4.2.3 Okno ukončení cesty

Pokud je cesta dokončena, je možné ji ukončit i v systému. K tomu je vytvořeno okno přístupné z hlavního menu pomocí tlačítka *Ukončit cestu.*

V tomto okně je nutné zadat koncový stav kilometrů z tachometru vozidla. Po potvrzení ukončení se hlavní záložka přesune zpět do rozvržení pro výběr cesty.

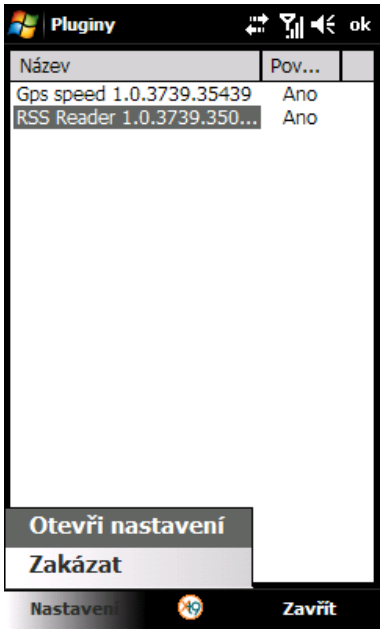


Obrázek 55 Okno pro ukončení cesty

4.2.4 Okno nastavení zásuvných modulů

Stejně jako desktopová část i mobilní umožňuje nastavení zásuvných modulů. Příslušný formulář je dostupný z hlavního menu pomocí tlačítka *Pluginy*.

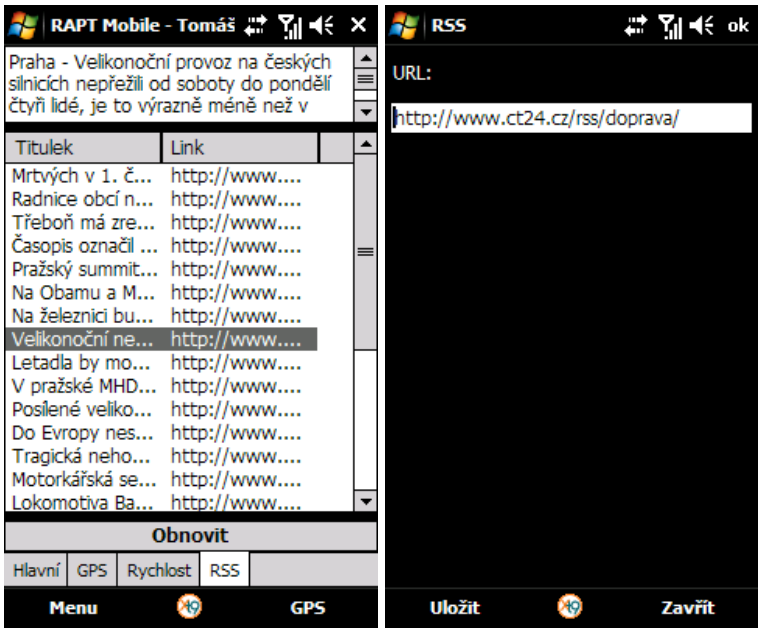
Na tomto formuláři se nachází tabulka zásuvných modulů. Tyto moduly je možné v menu formuláře zapínat nebo vypínat – položka menu se mění dle aktuálního stavu vybraného modulu. Dále je možné zobrazit formulář *Nastavení* jednotlivých modulů.



Obrázek 56 Okno nastavení zásuvných modulů

4.2.5 Zásuvný modul čtečky RSS kanálů

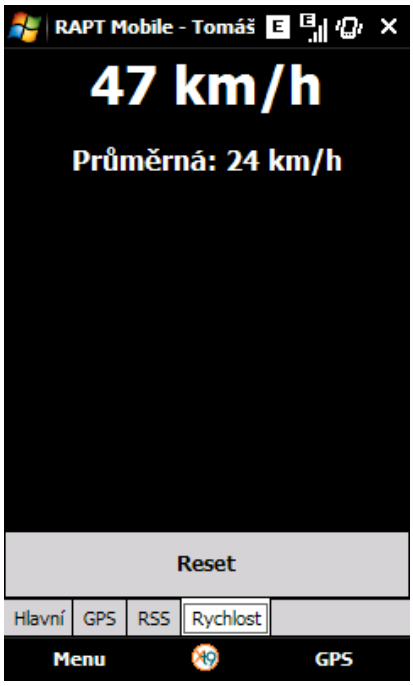
Tento modul poskytuje záložku se stejnou funkcionalitou jako modul v desktopové verzi. Umožňuje zobrazovat zprávy z RSS kanálu nastaveného v konfiguračním okně.



Obrázek 57 Záložka čtečky RSS kanálů a okno jejího nastavení

4.2.6 Zásuvný modul pro zobrazení rychlosti

Tento modul umožňuje v případě dostupných GPS souřadnic na separátní záložce zobrazovat aktuální a průměrnou rychlost. Hodnotu průměrné rychlosti lze během cesty resetovat.



Obrázek 58 Záložka zásuvného modulu pro zobrazení rychlosti

5 Závěr

Všechny cíle, stanovené v zadání a úvodu diplomové práce byly splněny. Byl vytvořen systém umožňující plánování cest řidičů a vozidel včetně online sledování v desktopové aplikaci a umožňující přímé zaznamenávání během cesty pomocí mobilního zařízení vybaveného modulem GPS. Systém je také dále dynamicky rozšiřitelný o zásuvné moduly v podobě *dll* knihoven, čímž je usnadněno další možné vylepšování.

Ovládání aplikace je postaveno tak, aby bylo co nejvíce jednoduché, intuitivní a srozumitelné. Tomu vděčí i díky vícejazyčnosti prostředí.

Jediným omezením se ukázala být rychlost internetového připojení mobilního zařízení, kdy v místech, kde byla k dispozici pouze technologie GPRS, byly pomalejší odezvy při načítání dat. Při testu byl použit telefon Samsung Omnia i900 s připojením Na stálo od firmy Vodafone. I přes toto omezení však mobilní aplikace plnila svůj účel a celý systém pracoval tak, jak bylo požadováno.

Soupis bibliografických citací

[1] WESLEY, Addison. *UML2 a unifikovaný proces vývoje*. Brno: Computer Press, a. s., 2008. 567 s. ISBN 978-80-251-1503-9.

[2] NAGEL CH. et al. *C# 2008. Programujeme profesionálně*. Brno: Computer Press, 2009. ISBN 80-251-1181-4.

[3] TROELSEN, A. *C# a .NET 2.0 profesionálně*. Brno: Zoner Press, 2007. ISBN 80-86815-38-2.

[4] Global Positioning System na *Wikipedie : otevřená encyklopedie* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 30. 8. 2005, 19. 3. 2010 [cit. 2010-03-21]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Global_Positioning_System>

[5] Windows Mobile na *Wikipedie : otevřená encyklopedie* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 20. 8. 2006, 23. 2. 2010 [cit. 2010-03-21]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Windows_Mobile>

[6] .NET na *Wikipedie : otevřená encyklopedie* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2. 8. 2005, 22. 9. 2009 [cit. 2010-03-21]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/.NET>

[7] *Wikipedie : Message-Digest algorithm* [online]. 2008 [cit. 2008-04-27]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/MD5>.

[8] *Návrhové vzory*. Brno: Computer Press, a. s., 2007. 527 s. ISBN 978-80-251-1582-4

[9] Model-view-controller na *Wikipedie: otevřená encyklopedie* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 11. 6. 2006, 23. 3. 2009 [cit. 2010-03-28]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Model-view-controller>.

[10] Observer na *Wikipedie: otevřená encyklopedie* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 5. 1. 2010, 8. 3. 2010 [cit. 2010-03-28]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Observer>.

[11] Globální družicový polohovací systém na *Wikipedie : otevřená encyklopedie* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 10. 4. 2008, 9. 4. 2010 [cit. 2010-04-10]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/ Globální_družicový_polohový_systém>.

[12] NMEA 0183 na *Wikipedie : otevřená encyklopedie* [online].St. Petersburg (Florida) : Wikipedia Foundation, 29. 3. 2007, 5. 4. 2010 [cit. 2010-04-10]. Dostupné z WWW: <http://en.wikipedia.org/wiki/NMEA_0183>.

[13] *Www.manualy.net* [online]. 207 [cit. 2010-04-10]. Teorie relačních databází: Normalizace. Dostupné z WWW: <http://www.manualy.net/article.php?articleID=13>.

[14] RSS na *Wikipedie : otevřená encyklopedie* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 7. 10. 05, 15. 3. 2009 [cit. 2010-04-13]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/RSS>.

[15] Extensible na *Wikipedie : otevřená encyklopedie* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 6. 6. 2006, 21. 10. 2007 [cit. 2010-04-13]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Extensible_Markup_Language>.

[16] *MSDN* [online]. 28. 8. 2008 [cit. 2010-04-13]. GPS Application . Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/bb158699.aspx>.

[17] *Zipcodeworls* [online]. 2009 [cit. 2010-04-14]. Zipcodeworld. Dostupné z WWW: <http://www.zipcodeworld.com/samples/distance.cs.html>.

Seznam příloh

Záznam o údržbě

Údržbář:	Růt Tomáš
Vozidlo:	Ford Scorpio 3E3 3333
Datum:	28.2.2010
Stav tachometru:	195000

Provedené úkony:

kontrola brzd

.....
Podpis

Příloha C

Výpis jízd

Řidič	Vozidlo	Začátek	Cíl	Plánovaný odjezd	Plánovaný příjezd do cíle	Odjezd	Příjezd do cíle	Konečný stav km
Růt Tomáš	Hyundai Atos 3E3 5548	Lázně Bohdaneč	Černá u Bohdanče	1.3.2010 20:00:07	1.3.2010 22:00:07	1.3.2010 22:02:37	1.3.2010 22:12:08	199920
Růt Tomáš	Ford Scorpio 3E3 3333	Pardubice	Lázně Bohdaneč	5.3.2010 13:00:44	5.3.2010 14:00:44	5.3.2010 13:40:20	5.3.2010 13:54:04	199930
Růt Tomáš	Ford Scorpio 3E3 3333	Lázně Bohdaneč	Kynice	6.3.2010 10:00:00	6.3.2010 12:00:00	6.3.2010 10:14:03	7.3.2010 14:29:31	200055
Růt Tomáš	Ford Scorpio 3E3 3333	Kynice	Lázně Bohdaneč	7.3.2010 13:00:00	7.3.2010 15:00:00	7.3.2010 13:39:10	7.3.2010 14:35:07	200115
Růt Tomáš	Ford Scorpio 3E3 3333	Lázně Bohdaneč	Pardubice-Pardubičky	8.3.2010 5:30:37	8.3.2010 6:00:37	8.3.2010 5:35:41	8.3.2010 5:52:57	200130
Růt Tomáš	Hyundai Atos 3E3 5548	Lázně Bohdaneč	Pardubice-Pardubičky	9.3.2010 5:30:50	9.3.2010 6:00:50	9.3.2010 5:32:45	9.3.2010 5:57:31	1966
Růt Tomáš	Ford Scorpio 3E3 3333	Lázně Bohdaneč	Rybitví	10.3.2010 20:07:57	10.3.2010 20:07:57	10.3.2010 21:19:32	10.3.2010 21:25:38	200153
Růt Tomáš	Hyundai Atos 3E3 5548	Pardubice-Pardubičky	Lázně Bohdaneč	11.3.2010 17:00:22	11.3.2010 18:00:22	11.3.2010 17:10:24	11.3.2010 17:30:27	2515
Růt Tomáš	Ford Scorpio 3E3 3333	Lázně Bohdaneč	Moravany	14.4.2010 20:00:49	14.4.2010 22:00:49			

Příloha D

Záznam o cestě

Řidič:	Růt Tomáš	Plánovaný začátek:	6.3.2010 10:00
Vozidlo:	Ford Scorpio	Plánovaný konec:	6.3.2010 12:00
RZ:	3E3 3333		
Schválil:	Růt Tomáš	Ukončení cesty:	7.3.2010 15:02
Potvrdil:	Růt Tomáš	Konečný stav km:	200055

Průběh cesty:

Začatek:	6.3.2010 10:14
Konec:	7.3.2010 14:29

Lázně Bohdaneč - Rybitví - Pardubice - Heřmanův Městec - Čepí - Podhořany - Dolní Bučice - Čáslav - Golčův Jeníkov - Leština u Světlé - Kynice

…….

podpis řidiče

Záznam o cestě

Řidič:	Růt Tomáš	Plánovaný začátek:	6.3.2010 10:00
Vozidlo:	Ford Scorpio	Plánovaný konec:	6.3.2010 12:00
RZ:	3E3 3333		
Schválil:	Růt Tomáš	Ukončení cesty:	7.3.2010 15:02
Potvrdil:	Růt Tomáš	Konečný stav km:	200055

Průběh cesty:

Začatek:	6.3.2010 10:14
Konec:	7.3.2010 14:29

Lázně Bohdaneč - Rybitví - Pardubice - Heřmanův Městec - Čepí - Podhořany - Dolní Bučice - Čáslav - Golčův Jeníkov - Leština u Světlé - Kynice

…….

podpis řidiče

Lázně Bohdaneč	6.3.2010 10:14:03
Rybitví	6.3.2010 10:20:25
Pardubice	6.3.2010 10:40:16
Heřmanův Městec	6.3.2010 11:27:36
Čepí	6.3.2010 11:23:01
Podhořany	6.3.2010 11:35:47
Dolní Bučice	6.3.2010 11:41:29
Čáslav	6.3.2010 11:45:22
Golčův Jeníkov	6.3.2010 11:53:09
Leština u Světlé	6.3.2010 14:15:10
Kynice	7.3.2010 14:29:31

Příloha F

TripPoints

«column»
*PK id: bigint
FK tripID: bigint
FK pointID: bigint
SeqNo: int
ThroughfareDateTime: datetime
FK createby: bigint
createdate: datetime
FK lasteditby: bigint
lasteditdate: datetime
valid: bit

«FK»
+ FK_TripPoints_Users_createby(bigint)
+ FK_TripPoints_Users_lasteditby(bigint)
+ FK_TripPoints_Point(bigint)
+ FK_TripPoints_Trip(bigint)

«unique»
+ UQ_TripPoints_id(int)

«PK»
+ PK_TripPoints(bigint)

PathLog

«column»
*PK id: bigint
FK tripID: bigint
Latitude: real
Longitude: real
ThroughfareDateTime: datetime
FK createby: bigint
createdate: datetime
FK lasteditby: bigint
lasteditdate: datetime
valid: bit

«FK»
+ FK_PathLog_Users_createby(bigint)
+ FK_PathLog_Users_lasteditby(bigint)
+ FK_PathLog_Trip(bigint)

«unique»
+ UQ_PathLog_id(int)

«PK»
+ PK_PathLog(bigint)

Trip

«column»
*PK id: bigint
vehicleID: bigint
FK userID: bigint
Purpose: varchar(200)
ApproximateDistance: int
Approved: bit = 0
FK ApprovedById: bigint
ApprovedDateTime: datetime
PlannedDateTimeBegin: datetime
PlannedDateTimeEnd: datetime
Finished: bit
FinishDateTime: datetime
FinalKilometerReading: bigint
Validated: bit
ValidateDateTime: datetime
FK ValidatedById: bigint
FK createby: bigint
createdate: datetime
FK lasteditby: bigint
lasteditdate: datetime
valid: bit

«FK»
+ FK_Trip_Users_aproovedbyid(bigint)
+ FK_Trip_Users_createby(bigint)
+ FK_Trip_Users_lasteditby(bigint)
+ FK_Trip_Users_userid(bigint)
+ FK_Trip_Users_validatedby(bigint)

«unique»
+ UQ_Trip_id(bigint)

«PK»
+ PK_Trip(bigint)

Vehicle

«column»
*PK id: bigint
Mark: varchar(50)
Type: varchar(50)
FK vehicleCategoryID: bigint
RegistrationMark: varchar(50)
KilometerReading: int
LastTechnicalControlDate: datetime
Insurance: varchar(50)
Year: int
FK createby: bigint
createdate: datetime
FK lasteditby: bigint
lasteditdate: datetime
valid: bit

«FK»
+ FK_Vehicle_Users_createby(bigint)
+ FK_Vehicle_Users_lasteditby(bigint)
+ FK_Vehicle_VehicleCategory(bigint)

«unique»
+ UQ_Vehicle_id(int)

«PK»
+ PK_Vehicle(bigint)

Point

«column»
*PK id: bigint
Name: varchar(30)
Circle: float
Latitude: real
Longitude: real
FK createby: bigint
createdate: datetime
FK lasteditby: bigint
lasteditdate: datetime
valid: bit

«FK»
+ FK_Point_Users_createby(bigint)
+ FK_Point_Users_lasteditby(bigint)

«unique»
+ UQ_Point_id(int)

«PK»
+ PK_Point(bigint)

Users

«column»
Login: varchar(20)
*PK id: bigint
Surname: varchar(30)
Name: varchar(30)
Title: varchar(20)
Street: varchar(30)
HouseNo: int
City: varchar(30)
PostCode: int
PhoneNo: varchar(20)
Email: varchar(30)
BirthNo: varchar(11)
Administrator: bit
Dispatcher: bit
Driver: bit
Password: varchar(50)
Picture: Image
FK createby: bigint
createdate: datetime
FK lasteditby: bigint
lasteditdate: datetime
valid: bit

«FK»
+ FK_Users_Users_createby(bigint)
+ FK_Users_Users_lasteditby(bigint)

«unique»
+ UQ_Users_Login(varchar)
+ UQ_User_id(int)

«PK»
+ PK_User(bigint)

Map

«column»
*PK id: bigint
Name: varchar(30)
MaxLatitude: real
MaxLongitude: real
MinLatitude: real
MinLongitude: real
Picture: Image
FK createby: bigint
createdate: datetime
FK lasteditby: bigint
lasteditdate: datetime
valid: bit

«FK»
+ FK_Map_Users_createdby(bigint)
+ FK_Map_Users_lasteditby(bigint)

«unique»
+ UQ_Map_id(int)

«PK»
+ PK_Map(bigint)

VehicleCategory

«column»
*PK id: bigint
Category: varchar(10)
Description: varchar(100)
TechnicalControlFirst: int
TechnicalConrolPeriod: int
FK createby: bigint
createdate: datetime
FK lasteditby: bigint
lasteditdate: datetime
valid: bit

«FK»
+ FK_VehicleCategory_Users_createby(bigint)
+ FK_VehicleCategory_Users_lasteditby(bigint)

«unique»
+ UQ_VehicleCategory_id(int)

«PK»
+ PK_VehicleCategory(bigint)

VehicleMaintainance

«column»
*PK id: bigint
FK vehicleID: bigint
FK MaintainanceManId: bigint
MaintainanceDate: datetime
KilometerReading: int
Notes: varchar(50)
FK createby: bigint
createdate: datetime
FK lasteditby: bigint
lasteditdate: datetime
valid: bit

«FK»
+ FK_VehicleMaintainance_Users_createby(bigint)
+ FK_VehicleMaintainance_Users_lasteditby(bigint)
+ FK_VehicleMaintainance_Users_MaintatinanceManId(bigint)
+ FK_VehicleMaintainance_Vehicle(bigint)

«unique»
+ UQ_VehicleMaintainance_id(int)

«PK»
+ PK_VehicleMaintainance(bigint)

UsersDrivingLicenses

«column»
*PK id: bigint
FK userid: bigint
FK drivinglicenseid: bigint
FK createby: bigint
createdate: datetime
FK lasteditby: bigint
lasateditdate: datetime
valid: bit

«FK»
+ FK_UsersDrivingLicenses_DrivingLicenses(bigint)
+ FK_UsersDrivingLicenses_Users_createby(bigint)
+ FK_UsersDrivingLicenses_Users_lasteditby(bigint)
+ FK_UsersDrivingLicenses_Users_user(bigint)

«PK»
+ PK_UsersDrivingLicenses(bigint)

«unique»
+ UQ_UsersDrivingLicenses_id(bigint)

DrivingLicenses

«column»
*PK Id: bigint
* License: varchar(2)
FK createby: bigint
createdate: datetime
FK lasteditby: bigint
lasteditdate: datetime
valid: bit

«FK»
+ FK_DrivingLicenses_createby(bigint)
+ FK_DrivingLicenses_Users_lasteditby(bigint)

«PK»
+ PK_DrivingLicenses(bigint)

«unique»
+ UQ_DrivingLicenses_Id(bigint)
+ UQ_DrivingLicenses_License(varchar)

VehicleDrivingLicenses

«column»
*PK id: bigint
FK vehicleid: bigint
FK drivinglicenseid: bigint
FK createby: bigint
createdate: datetime
FK lasteditby: bigint
lasteditdate: datetime
valid: bit

«FK»
+ FK_VehicleDrivingLicenses_DrivingLicenses(bigint)
+ FK_VehicleDrivingLicenses_Users(bigint)
+ FK_VehicleDrivingLicenses_Users_createby(bigint)
+ FK_VehicleDrivingLicenses_Vehicle(bigint)

«PK»
+ PK_VehicleDrivingLicenses(bigint)

«unique»
+ UQ_VehicleDrivingLicenses_id(bigint)

UpdateMarks

«column»
*PK tablename: varchar(30)
* updatemark: bigint = 1

«PK»
+ PK_UpdateMarks(varchar)

«unique»
+ UQ_UpdateMarks_tablename(varchar)