VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

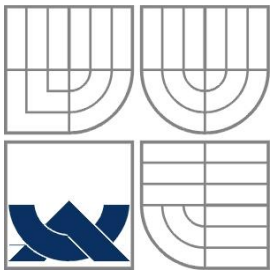
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

# INTELIGENTNÍ KNIHA JÍZD

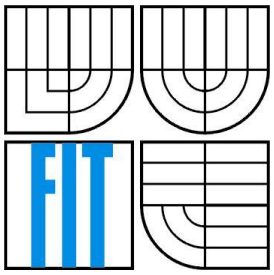
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE VÁCLAV HOLUŠA
AUTHOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

# INTELIGENTNÍ KNIHA JÍZD
INTELLIGENT LOG BOOK

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
VÁCLAV HOLUŠA

VEDOUCÍ PRÁCE
SUPERVISOR
ING. ONDŘEJ MALAČKA

BRNO 2011

# Abstrakt

Tématem této práce je návrh a vytvoření aplikace – Inteligentní knihy jízd – sloužící k uživatelsky přívětivému zobrazení historie jízd služebních vozidel. Současně mohou být data podrobena analýze pro zjištění překračování povolené rychlosti, případně bezdůvodného stání s nastartovaným motorem. Vozidla a jejich trasy jsou zakresleny v mapových podkladech Google Maps. Aplikace je nasazena v prostředí konkrétní firmy, jejíž zaměstnanci poskytli součinnost během jejího vytváření. Je implementována jako webová především s ohledem na požadavek možnosti jejího použití na různých zařízeních (počítač, mobilní telefon) a různých platformách.



# Abstract

The topic of this thesis is the proposal and creation of the Intelligent Log Book application, which is used for user-friendly view of the company vehicle's rides history. Data can be analyzed to check exceeding of the speed limit or unreasonable standing with running engine. Vehicles and their tracks are plotted into Google Maps. Application is deployed in a specific company. Its employees provided cooperation during development. Project is implemented as a web-based application due to request for usage on different machines (PC, mobile phone) and different platforms.



# Klíčová slova

GPS, kniha jízd, služební vozidla, Google mapy, analýza jízd, kontrola zaměstnanců, trasa na mapě, snižování nákladů

# Keywords

GPS, log book, company vehicles, Google Maps, rides analysis, monitoring of employees, track on map, costs reduction


# Citace

Holuša Václav: Inteligentní kniha jízd, bakalářská práce, Brno, FIT VUT v Brně, 2011

# Inteligentní kniha jízd

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Ondřeje Malačky.
Další informace mi poskytl Ing. Jan Horáček.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Václav Holuša
16.5.2011

## Poděkování

Rád bych poděkoval vedoucímu mé práce, Ing. Ondřeji Malačkovi, za poskytnutí cenných informací, pomoci a součinnosti během psaní této práce. Dále bych chtěl poděkovat zaměstnancům firmy MIRABEL lexart a.s. za stanovení požadavků a zpětnou vazbu a připomínky, které podávali během vývoje aplikace.

# Obsah

# 1 Úvod

V podnikatelském prostředí, nejen v době stále přetrvávající hospodářské krize, je jednou z klíčových záležitostí efektivní využívání všech firemních prostředků. Mezery v efektivnosti využívání těchto prostředků způsobují firmám zvýšené náklady a přispívají k oslabení konkurenceschopnosti firmy. Jedním z faktorů, který způsobuje právě tuto neefektivitu, je zneužívání firemních prostředků zaměstnanci. Ti často používají mobilní telefony k soukromým hovorům, firemní počítače k osobním účelům apod. V rámci této bakalářské práce se zaměřím na další z řady případů – zneužívání firemních vozidel.

Existuje mnoho způsobů, jak může nepoctivý zaměstnanec neoprávněně využívat prostředky firemního vozového parku – soukromé jízdy služebními vozidly, porušování interních pravidel firmy či obecně platných zákonů a dopravních předpisů, úmyslně nesprávné vedení evidence služebních jízd či dokonce krádeže pohonných hmot.

Mimo to nelze opomenout i neefektivitu vznikající špatnou koordinací provozu firemních automobilů. Jedná se zde většinou o situace, kdy vozidlo v daný okamžik není na místě, kde je ho nejvíce potřeba, či naopak je na stejné trase uskutečněno několik jízd různými vozidly, přičemž by se daly některé cesty sloučit.

S provozem firemních vozidel je pro firmu dále spojeno mnoho rizik. Jedná se o rizika jejich poškození, krádeže či způsobení škody třetí osobě, rizika spojená se sankcemi v případě nedodržení povinnosti vést správnou evidenci, dodržování zákonných přestávek, nebo rizika sankcí v případě porušení dopravních předpisů.

Všechny zmíněné problémy do určité míry eliminuje kvalitní kontrola provozu firemních vozidel. Cílem této bakalářské práce je vytvoření aplikace – Inteligentní knihy jízd, která by podobnou kontrolu usnadnila. Inteligence této aplikace spočívá v možnostech, kterými klasická papírová kniha jízd nedisponuje. Nelze však říci, že by tuto plně nahrazovala. Jedná se především o uživatelsky přívětivé zobrazení kompletní historie jízd vozidel a možnost jejich analýzy. Ta spočívá ve zjištění a upozornění na překročení povolené rychlosti či bezdůvodné stání s nastartovaným motorem, dodržování zákonných přestávek, hlídání časů odjezdů, příjezdů, apod. Další výhodou je možnost generování chronologicky seřazeného tabulkového výpisu jízd pro usnadnění kontroly správnosti vyplnění knihy jízd.

Systém je vytvářen v prostředí konkrétní společnosti, měl by však být obecně použitelný. Při návrhu a implementaci vycházím z existujícího hardwarového řešení v podobě zařízení, která umožňují z jednotlivých vozidel odesílat informace získané z vlastních GPS antén.

Podstatou vytvořené aplikace je pomocí těchto informací, které jsou samostatně nepoužitelné, umožnit kontrolovat provoz firemních vozidel a strojů a adekvátně reagovat na neočekávané situace, které při jejich provozu vznikají.

Tato bakalářská práce se zabývá částí systému, která je zodpovědná za zpracování uložených dat, zobrazení vybraných informací v aplikaci, práci s vybranými mapovými podklady, uživatelsky přívětivé zobrazení aktuálně provozovaných vozidel na mapě, vedení knihy jízd, zobrazení trasy jednotlivých jízd na mapových podkladech a zpracování potřebných statistických dat. Aplikace je implementována jako webová, a to především s ohledem na požadavek zákazníků týkající se jejího použití na různých platformách a zařízeních.

Kapitola 2 se zabývá motivací k vytvoření konkrétní aplikace, vzniklé analýzou situací, které mohou ve firmách nastat v souvislosti s vozovým parkem a nastíněním současného stavu v konkrétní firmě, ve které bude aplikace nasazena.

Kapitola 3 ve stručnosti popisuje cílový stav, který by měl být dosažen na základě analýzy současného stavu a požadavků zákazníka.

Kapitola 4 je zaměřena na návrh cílové aplikace s ohledem na využití existujícího zařízení. Jsou zde uvedeny požadavky uživatelů v konkrétní firmě a postup při návrhu aplikace na míru těmto požadavkům. Dále je v kapitole popsán postup výběru vhodného poskytovatele mapových podkladů.

V 5. Kapitole je uveden postup implementace obou částí aplikace – serverové a klientské. Jsou zde popsány zvolené platformy, nástroje a knihovny.

Kapitola 6 se zabývá nasazením aplikace v konkrétní vybrané firmě a popisuje příklady jejího využití v praxi.

V kapitole 7 jsou stručně rozepsány možnosti dalšího rozšíření aplikace.

# 2 Motivace

Cílem této práce je snaha o vyřešení problémů, se kterými se potýká drtivá většina firem. První fází je popis výchozích podmínek uvažované firmy a obecných problémů, se kterými se potýká. Na základě jejich analýzy bude možné navrhnout řešení, které by následky zmíněných problémů pomohlo zmírnit.

V práci budu systém navrhovat pro podmínky obecné firmy, přičemž v jednotlivých kapitolách postupně popíši aplikaci popisovaných bodů na podmínky vybrané konkrétní společnosti.

Vycházejme z faktu, že ve většině firem existují firemní vozidla. Má-li firma zaměstnance, pak je přirozené, že jsou jimi tato vozidla používána k pracovním účelům. Většina problémů, se kterými firmy bojují, vycházejí právě z toho, že použití vozidel je často v rozporu s předem dohodnutými pravidly, ať už jsou to neformální dohody, interní předpisy či zákonem vyžadované postupy a normy.

## 2.1 Situace ve firmě

Předpokládejme klasické hierarchické uspořádání zaměstnanců ve firmě. V tomto uspořádání mají vedoucí pracovníci zvýšené pravomoce, ale také zodpovědnost. Vedoucí zaměstnanci tudíž potřebují vykonávat dohled nad prací svých podřízených.

Jak už bylo řečeno, zaměstnanci ke své práci využívají firemní vozidla. V praxi je toto zajišťováno tak, že zaměstnanci je přiděleno či svěřeno služební vozidlo, za jehož provoz a všechny věci s ním související je odpovědný. Pokud v souvislosti s provozem vozidla vyvstane problém, pak se jej tento odpovědný zaměstnanec snaží v rozsahu své odpovědnosti vyřešit. Nedokáže-li sám řešení nalézt, pak ho konzultuje s nadřízeným. Některá vozidla používá dle aktuální potřeby více zaměstnanců, nikoliv pouze odpovědná osoba.

Předpokládejme stav, že firemní prostředky (a tedy i firemní automobily) slouží primárně k firemním účelům. Jinými slovy, vozidla by měla být využívána pouze k jízdám souvisejícím s vykonáváním pracovní činnosti.

Skutečností je, že zaměstnanci, často bez vědomí vedoucího, vozidla využívají i k soukromým účelům, a to obvykle i v pracovní době. Tento jev způsobuje firmě zbytečné dodatečné náklady, které jsou samozřejmě vysoce nežádoucí, neboť staví firmu do nevýhodného postavení vzhledem ke konkurentům. Navíc z těchto nežádoucích jízd pro firmu vyplývají také určitá rizika, neboť firma je do určité míry zodpovědná za zaměstnance během pracovní doby (odpovědnost za pracovní úraz) a zde se nesouvisející náklady mohou řádně prodražit. Dále existuje riziko škody na firemním majetku, případně i škody na majetku třetích osob.

Dalším předpokladem je, že firma využívá vozidla různých typů. Tato vozidla mají často zcela jiné charakteristiky, využití a požadavky na kontrolu nad provozem. Pro ilustraci můžeme uvést např. následující typy (ve skutečnosti lze předpokládat, že se jednotlivé typy vozidel v podmínkách každé společnosti liší daleko více).

- **Osobní automobily a dodávky.**
  Osobní automobily, případně lehké užitkové vozy jsou využívány k různým cestám – některé služební jízdy jsou krátké, některé i velmi dlouhé. Vozidla slouží jak k přepravě osob, např.

dělníků na stavbu či montáž, tak k přepravě materiálu. Riziko vzniku škody na těchto vozidlech v běžném provozu je vysoké. Náklady na opravy také nejsou zanedbatelné.

- **Nákladní automobily.**
  Charakteristiky různých značek a typů nákladních automobilů jsou odlišné. Tyto vozy nejsou určeny pro běžný provoz, slouží převážně k přepravě materiálu. Vyznačují se vysokou spotřebou (což má za následek i větší ztráty v případě neoprávněného používání), a nesnadnou kontrolu spotřebovaného paliva.

- **Stavební nebo speciální stroje.**
  Další skupinu vozidel tvoří stavební stroje. Tato vozidla se vyznačují velmi vysokou spotřebou pohonných hmot a zcela jinou charakteristikou ujetých tras. Vozidla při práci často stojí na místě, případně popojíždí po velmi krátkých úsecích (běžně v řádech jednotek metrů). Jejich opravy vyžadují specializovaný servis a bývají velmi nákladné. Neoprávněný provoz tak může firmě způsobit nemalé škody v podobě zvýšené spotřeby pohonných hmot, dodatečné náklady při odstávce poškozeného stroje (potřeba zapůjčení jiného) nebo nákladů na jeho opravu.

Kontrola a dohled nad provozem vozidel jsou pro každou firmu nákladné činnosti. Zodpovědní pracovníci jsou často vytížení a nemohou být v jednu chvíli na více místech. Přijetí dalších zaměstnanců pro tento účel by provoz vozidel, potažmo provoz celé firmy, učinilo značně nákladnějším, nehledě na fakt, že vykonávání osobního dohledu nad služebními jízdami je organizačně neproveditelné.

## 2.2 Časté problémy

Pokusme se vytipovat některé z častých problémů, se kterými se firmy využívající služební vozidla potýkají:

- **Zaměstnavatel nemá přehled o uskutečněných jízdách.**
  Zaměstnanci využívají služební vozidla i k soukromým účelům, a odpovědným pracovníkům tyto skutečnosti zamlčují. To vede nejen k rizikům a nákladům zmíněným v předchozích odstavcích, ale zprostředkovaně i ke snížení produktivity práce (zaměstnanci část pracovní doby nevykonávají efektivní práci ve prospěch zaměstnavatele). Vzhledem k tomu, že není možné jednoznačně určit, jak dlouho jednotlivé služební cesty mají trvat, nebo přesně určit počet kilometrů (zácpy, kolony, objížďky) je toto neoprávněné využívání vozidel zaměstnancům těžko dokazatelné.

- **Zaměstnanci překračují povolenou rychlost.**
  Řidiči jsou si vědomi faktu, že vlastníkem a provozovatelem automobilu je firma, tudíž necítí osobní zodpovědnost za přestupky proti pravidlům silničního provozu. Nemají také potřebu šetřit provozní náklady. Dochází tak zbytečně k vysokému opotřebení automobilů a vysoké spotřebě pohonných hmot. Občas se také stává, že vozidlo je zachyceno policejním radarem (např. při překročení povolené rychlosti), a je poté obtížné zjistit, kdo je za spáchaný přestupek odpovědný.

- **Není snadné dohledat, kde se jednotlivá auta nacházejí.**
  Při běžném provozu firmy dochází k situacím, kdy automobil nutně potřebuje jiný zaměstnanec, než ten, který ho aktuálně používá. V takových chvílích je nesnadné dohledat, kde se jednotlivá vozidla nacházejí a vybrat tak např. náhradní vozidlo pro splnění úkolu či zakázky nebo najít pomoc pro řidiče, který se pro neodstranitelnou poruchu svého vozu ocitnul na daném místě v nesnázích. Je často nutné telefonicky obvolat několik zaměstnanců a zjistit, kde právě jednotlivá vozidla jsou, za jak dlouho mohou přijet zpět, jestli jsou právě využívána, případně zda mají dostatek pohonných hmot.

- **Dochází ke krádežím pohonných hmot.**
  Tento problém se týká zejména speciálních (např. stavebních) strojů, které mají velkoobjemové nádrže a malé odchylky při tankování a spotřebě jsou těžko odhalitelné. Nehledě na fakt, že ke krádežím pohonných hmot dochází i u odstavených vozidel, protože jejich palivové nádrže jsou neuzamykatelné a snadno dostupné.

- **Vyskytují se problémy při vedení knihy jízd.**
  Zaměstnavatel je ze zákona povinen vést pečlivě, úplně a správně knihu jízd. Zaměstnanci často i přes příkazy vedení tuto povinnost neplní, nebo si údaje v knihách jízd dodatečně vymýšlejí či tyto zkreslují. Kontrola správnosti takto vyplněných údajů je velmi obtížná až nemožná. Ztráty způsobené vedením chybné evidence se projevují jak v daňové oblasti, tak zvýšeným rizikem postihů ze strany státních kontrolních orgánů.

- **Je obtížné prokazovat jednotlivá provinění zaměstnanců.**
  Neexistuje-li jasný přehled o tom, kdo dané vozidlo v daný okamžik řídil, všechny přestupky, škody či způsobené problémy (ať už se jedná o pokuty, dopravní nehody, poškození způsobená nesprávným používáním či jiné problémy zmiňované v předchozích bodech) jsou obtížně prokazatelné. Vymáhání náhrady škody po zaměstnanci je tak často nemožné. Zaměstnanci přirozeně zamlčují, že vozidlo v daný okamžik řídili a vzhledem k tomu, že u služebních vozidel obvykle dochází ke střídání řidičů, je nesnadné tento negativní jev omezit.

## 2.3 Prostředí konkrétní firmy

Praktická implementace našeho řešení je prováděna v prostředí vybrané firmy. Zde již existuje hardwarový systém pořizující data o poloze vozidel. Vzhledem k tomu, že hardwarová část tohoto řešení není předmětem této bakalářské práce, je existující systém popsán jen ve stručnosti.

Každé vozidlo je vybaveno zařízením, které pomocí GPS antény snímá aktuální polohu v předem daných intervalech (v našich podmínkách každých 30 vteřin). Ihned po zaznamenání polohy je prostřednictvím GSM sítě, pomocí technologie datových přenosů GPRS, odeslána informace o nové poloze v daném čase na server (pozn.: podrobnosti serverové části řešení jsou popsány v kapitole 4).

Zařízení jsou vybavena interní pamětí, která jim umožňuje v případě ztráty GSM signálu zaznamenávat polohu po určitou omezenou dobu (v našem případě přibližně 30 dnů) a po následném připojení je dávkově odeslat na server.

V případě, že dojde ke ztrátě GPS signálu, k žádnému odeslání, ani zaznamenání informace nedochází. Nová poloha je odeslána až v okamžiku, kdy je GPS signál obnoven.

Volba technologie GPRS se v průběhu času projevila jako vhodná, i přes nedostatky, kterými tato technologie trpí (zejména nestabilita a pomalá odezva). Hlavní výhodou je především vysoké pokrytí území České republiky GPRS signálem [5] (viz Tabulka č. 1).

| **Mobilní operátor** | **Pokrytí GPRS signálem** |
|---|---|
| Telefónica O2 | 98% |
| T-Mobile | 98% |
| Vodafone | 94% |

**Tabulka č. 1: Pokrytí území České republiky GPRS signálem**

# 3 Analýza cílového stavu

Cílem práce je vytvořit softwarové řešení, které za pomoci hardwarových zařízení popsaných v předchozí kapitole vyřeší co možná největší část popisovaných problémů. Vycházejme z analýzy cílového stavu popsaného v podmínkách konkrétní vybrané firmy. Snahou je dosáhnout takového návrhu, který by byl použitelný obecně, tj. nejen řešení na míru danému zákazníkovi.

V návaznosti na situaci popsanou v kapitole 2.1 je požadavkem vytvořit aplikaci, která by umožňovala výše uvedené problémy minimalizovat. Uživateli aplikace bude umožněno:

- **V každém okamžiku na obrazovce vidět aktuální polohu všech vozidel.**
  Je důležité, aby měl přehled o všech vozidlech v každém okamžiku, kdy s aplikací pracuje. U uživatele se předpokládá vysoké pracovní vytížení, proto je při návrhu aplikace třeba dbát na vysokou efektivitu používání. Není tedy žádoucí, aby pro zjištění polohy vozidla bylo zapotřebí např. vykonat několik kliknutí myši apod.
- **Mít k dispozici kompletní historii jízd daného vozidla.**
  Historii jízd vozidla uživatel zjišťuje až v případě, že má o tuto konkrétní informaci zájem (tedy např. při zpětném řešení problému). Není tedy zapotřebí, aby historie jízd byla bezprostředně přístupná v každém okamžiku.
- **Mít informaci o případném překročení rychlosti.**
  Vzhledem k tomu, že tento přestupek je možné řešit až, stačí mít tuto informaci k dispozici až na vyžádání.
- **Mít možnost zjistit vybrané statistické informace o jízdách.**
  V rámci vyhodnocování práce jednotlivých zaměstnanců, případně při hodnocení využití jednotlivých automobilů mohou být užitečné vybrané statistické údaje o provedených jízdách. Jedná se např. o doby jízdy, ujeté vzdálenosti, průměrné rychlosti, maximální dosažené rychlosti apod. U těchto informací nepředpokládáme nutnost vidět je v reálném čase (v každém okamžiku).

Vzhledem k tomu, že předpokládáme, že dohled nad vozidlem je jen jednou z mnoha činností, které pověřený zaměstnanec vykonává, není žádoucí, aby ovládání aplikace bylo složité. Z toho důvodu by aplikace měla obsahovat jen nutné minimální množství ovládacích prvků. Mělo by být na první pohled zřejmé, k čemu daný ovládací prvek slouží.

Vzhledem k datům, které má aplikace k dispozici, není implementačně složité, aby program poskytoval i další informace (např. složitější statistické informace o jízdách, grafy rychlosti apod.). V zájmu zachování jednoduchosti užívání však jsme ochotni se této funkcionality vzdát. Primárním požadavkem ze strany vybrané firmy bylo snadné užívání, bez nutnosti procesu zaškolení uživatele.

# 4 Návrh

Jak bylo popsáno v kapitole 2.3, při návrhu řešení počítáme s tím, že určitá část systému je již v provozu nasazená, a nelze ji proto modifikovat. Naše řešení se jí tudíž musí přizpůsobit. Jedná se především o tři základní komponenty:

- Hardwarová část – jednotlivá zařízení montovaná do automobilů.
- Systém přenosu dat mezi zařízeními v automobilech a databází.
- Databáze.

Tato aplikace tedy nemusí vůbec řešit problematiku pořízení vstupních dat.

## 4.1 Hardwarová část

V jednotlivých automobilech jsou instalována zařízení. Tato zařízení považujme za daná, princip jejich činnosti nespadá do tématu této bakalářské práce. Podrobnější popis jejich funkce je uveden v kapitole 2.3.

## 4.2 Systém přenosu dat mezi zařízeními v automobilech a databází

Na serveru, kde běží databáze, je spuštěna služba, která naslouchá na určeném portu a zachytává příchozí datovou komunikaci. Tu poskytují hardwarová zařízení v automobilech prostřednictvím sítě GSM, jak bylo popsáno v kapitole 2.3. Jedná se o krátké textové řetězce, ve kterých jsou v daném formátu uloženy informace:

- Čas, ve kterém byly zasílané hodnoty měřeny.
- Identifikátor zařízení.
- Aktuální poloha vozidla.
- Příznaky jednotlivých událostí.

Systém tak posílá specifickou zprávu při nastartování vozidla, při vypnutí motoru a v periodických intervalech v průběhu jízdy.

Vzhledem k tomu, že zařízení v automobilech jsou hardwarová a nemáme možnost jejich funkčnost ovlivnit, formát zasílaných zpráv mezi zařízeními a touto službou pro nás není podstatný.

## 4.3 Databáze

Máme k dispozici již navrhnutou (existující) databázi. Jednotlivé tabulky databáze jsou naplňovány službou popsanou v předchozí podkapitole.

### 4.3.1 Databázová platforma

Databáze, implementována na platformě Firebird, běží spolu s výše popsanou službou na dedikovaném serveru s operačním systémem Windows Server.

> *Projekt Firebird je komerčně nezávislý projekt programátorů C a C++, technických poradců a podporujících vývojářů. Nabízí multiplatformní relačně databázový systém založený na zdrojovém kódu vydaným společností Inprise Corp (nyní známa jako Borland Software Corp) v červenci 2000.* [1]

Databázová platforma Firebird se vyznačuje těmito, pro naše účely relevantními, charakteristikami:

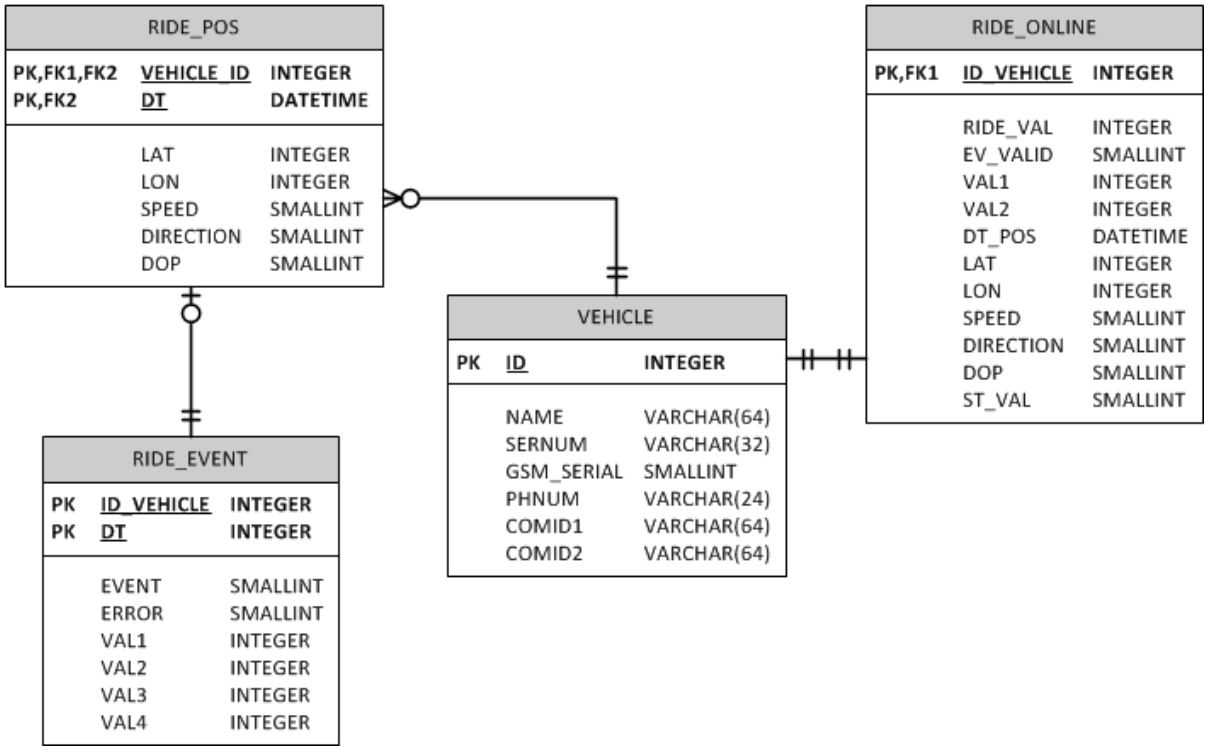
- Je multiplatformní (podporována jsou prostředí MS Windows, Mac OS X, Linux, BSD, Unix, Solaris, HP UX). [3]
- Podporuje uložené procedury a triggery.
- Podporuje práci s transakcemi.
- Poskytuje dostatečný výkon.
- V porovnání s jinými databázovými platformami poskytuje omezené možnosti týkající se indexů [2], což by mohlo být limitujícím faktorem především při budoucích rozšířeních. Vzhledem k tomu, že databáze pracuje s poměrně velkým množstvím dat (tato data neustále přibývají), která musí být archivována po potenciálně neomezenou dobu, by tato omezení mohla v budoucnu způsobit výkonnostní problémy.
- Poskytuje omezené možnosti partitioningu [2]. To by mohlo být také limitujícím faktorem (ze stejných důvodů, jako jsou uvedeny v předchozím bodu).

Z výše uvedených charakteristik můžeme usoudit, že platforma Firebird v současné chvíli požadavkům pro splnění cílů této práce bez problémů vyhovuje. V případě budoucího rozvoje aplikace by se však mohly vyskytnout určité komplikace, zejména výkonnostního charakteru.

### 4.3.2 Struktura databáze

V této kapitole se pokusme ve stručnosti popsat strukturu tabulek v existující databázi. Vybrány jsou pouze ty, které jsou pro účely této bakalářské práce relevantní. Ve skutečné databázi, se kterou pracujeme, jsou i některé další tabulky, které však nemají vztah ke zpracovávané problematice (jedná se např. o tabulky sloužící pro účely autentizace uživatelů, konfigurace a nastavení, logování činnosti, apod.)

Obrázek č. 1 znázorňuje zjednodušené schéma databáze, kde byly vynechány tabulky, které nejsou použity.




**Obrázek č. 1: Schéma databáze**

Funkce znázorněných tabulek jsou popsány v následujících odstavcích.

Tabulka *VEHICLE* slouží pro uložení jednotlivých vozidel zavedených v systému. Sloupec `NAME` obsahuje název vozidla. Jeho SPZ je uložena ve sloupci `SERNUM`. Sloupce `GSM_SERIAL` a `PHNUM` slouží ke spárování záznamu v databázi s příchozími daty, které odeslalo zařízení namontované v automobilu (podrobněji viz kapitola 2.3).

Tabulka *RIDE_ONLINE* slouží jako dočasné úložiště naposledy zaznamenané polohy vozidla. S každou nově příchozí informací o poloze vozidla je tato informace aktualizována. Důležité sloupce jsou `LAT` a `LON`, které uchovávají informaci o zeměpisné šířce resp. délce ve formátu „Decimal Degrees“ převedeném na milisekundy. Sloupec `DT_POS` obsahující časové razítko záznamu a dále sloupec `SPEED`, ve kterém je uložena aktuální rychlost vozidla a `DIRECTION` udávající směr jízdy v úhlových stupních, kde sever je 0°, jih 180°, apod. Ostatní sloupce obsahují nám neznámé hodnoty, některé jsou nulové.

Největší a nejpoužívanější tabulka *RIDE_POS* obsahuje jednotlivé zaznamenané body pro všechna vozidla zavedená v systému. Primárním klíčem tabulky je kombinace `VEHICLE_ID` a časového razítka `DT`. Sloupce `LAT`, `LON`, `SPEED` a `DIRECTION` jsou popsány v předchozím odstavci.

Tabulka *RIDE_EVENT* obsahuje záznamy v případě, že se jedná o začátek nebo konec jízdy. Záznam se uloží pouze pro daná časová razítka. Sloupec `EVENT` nabývá hodnoty 0, což značí začátek jízdy a hodnoty 1 značící konec jízdy. Spolu se značkou konce jízdy se ukládá do sloupce `VAL1` celková ujetá dráha dané jízdy. Ostatní sloupce jsou vždy nulové.

Existující struktura databáze se nejeví jako optimální, a to především z následujících důvodů:

- Záznamy v tabulkách nemají jednoznačné identifikátory. Pro naše aktuální účely to není výrazný problém (data budou z databáze především vybírána a zobrazována, úpravy či mazání záznamů zatím nejsou relevantní), nicméně z hlediska budoucího rozšíření či obecných konvencí se jedná o jednoznačně nesprávný návrh.
- Jako cizí klíč je na mnoha místech použit datový typ `DATETIME`. To se také nejeví jako příliš vhodný návrh, neboť to znesnadňuje navázání dalších tabulek, výběry záznamů apod.
- Pojmenování sloupců a tabulek je v mnoha případech nejasné, nevypovídající o jejich obsahu. Na mnoha místech se setkáme se sloupci pojmenovanými např. „`VAL1`“, „`VAL2`“, … aniž by jejich význam byl bez znalosti vnitřní struktury aplikace patrný. Pro tvůrce aplikace se zřejmě nejedná o závažný problém. Jelikož však implementujeme řešení, u kterého předpokládáme budoucí rozvoj, není to příliš šťastná volba.
- Pojmenování sloupců nezachovává jednotnou konvenci. Cizí klíče v různých tabulkách odkazující se na stejný atribut jsou pojmenovány různými způsoby. Například `ID_VEHICLE` vs. `VEHICLE_ID`.
- Tabulka `RIDE_ONLINE` slouží čistě k zobrazení naposledy uloženého údaje v tabulce `RIDE_POS`. Dochází tedy zbytečně k redundanci dat. Záznam o změně polohy vozidla je zapsán jak do tabulky `RIDE_POS`, tak i do tabulky `RIDE_ONLINE`, kde přepíše poslední obdobný záznam k tomuto vozidlu.
  Databáze používající takovou tabulku je dle [10, kap. 3.3] chybně navržená. Toto řešení zde bylo patrně zvoleno z výkonnostních důvodů, jelikož tabulka `RIDE_POS` obsahuje řádově statisíce až miliony záznamů (závisí na počtu aut a intenzitě jejich využití). Data v této tabulce neustále narůstají a čtení údajů z této tabulky je velmi častým úkonem. To v databázové praxi lze řešit vhodným indexováním. Z tohoto pohledu se jeví tabulka `RIDE_ONLINE` jako přebytečná.

### 4.3.3 Souhrn

V kontextu informací uvedených v předchozích podkapitolách vyvstávají dvě základní možnosti. Můžeme se smířit s existujícím návrhem databáze a vyvíjenou aplikaci jemu přizpůsobit, nebo máme možnost vytvořit další vrstvu aplikace, která by zajistila replikaci dat z existující databáze do nové, lépe navržené. V následujících několika odstavcích se pokusme shrnout pozitiva a negativa obou těchto možností.

Při práci s replikovanou databází bychom stáli před rozhodnutím, zda tuto databázi provozovat na stejné platformě, nebo použít vhodnější. V případě, že bychom se rozhodli pro jinou databázovou platformu, potýkáme se s možnou potřebou dalšího serverového vybavení, což cílové řešení činí nákladnějším. V případě, že využijeme stávající platformu, nezbavíme se některých technických nedostatků uvedených v kapitole 4.3.1.

Implementace replikační vrstvy by dále mohla znamenat určité výkonnostní problémy. V tuto chvíli tento problém není aktuální, nicméně v kontextu možného budoucího rozvoje se nejedná o systémově nejvhodnější řešení.

Vytvoření další vrstvy aplikace zvyšuje náklady na vývoj aplikace. Vzniká také větší prostor pro případný výskyt programátorských či technických chyb. Mimo to by to znamenalo složitější monitoring provozu těchto serverů.

Přínosem popisovaného řešení je zmíněný lepší databázový návrh. To by mohlo znamenat úsporu nákladů při budoucím rozšiřování vyvíjené aplikace. Dalším potenciálním přínosem by byla možnost zvolit jinou platformu pro databázi a učinit tak ostatní části aplikace méně závislé na konkrétní platformě.

Z výše uvedených informací lze soudit, že v současné chvíli se řešení založené na využití existující databáze jeví jako vhodnější. Přínosy vytvoření replikační vrstvy patrně nepřevyšují nad náklady s tím spojenými.

## 4.4 Implementovaná funkčnost

Jak již bylo popsáno podrobně v předchozích kapitolách, při volbě implementované funkcionality jsme limitováni existujícími hardwarovými zařízeními, existující databází a v neposlední řadě také rozsahem daném bakalářskou prací.

V návaznosti na analýzu požadovaného cílového stavu a rozhovory s vybranou konkrétní firmou jsme se rozhodli do implementace zahrnout následující funkcionalitu:

- Zobrazování aktuální polohy všech vozidel na mapě.
- Prohlížení historie jízd vybraného vozidla.
- Zobrazení trasy vybrané jízdy na mapě.
- Analýza historie jízd, statistické informace.
- Zobrazení informací o tom, která vozidla jsou v danou chvíli v provozu.

## 4.5 Návrh uživatelského rozhraní

### 4.5.1 Požadavky zákazníka

Při návrhu uživatelského rozhraní vyvíjené aplikace vycházíme z požadavků zákazníka (zvolené konkrétní společnosti). Předpokládejme, že tyto požadavky do značné míry odpovídají obecnému účelu aplikace. Výchozí informace jsme získali kontaktem s budoucími uživateli. Jejich stručný výčet je uveden v následujících odstavcích.

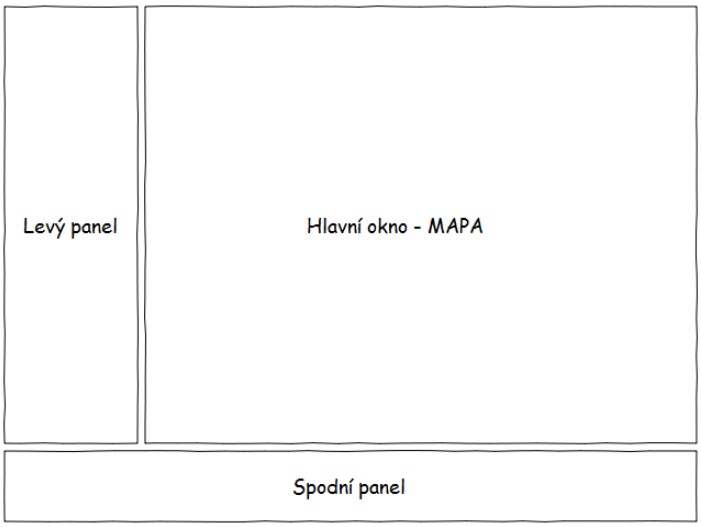
Důležitým rozhodnutím je, pro jaká zařízení bude uživatelské rozhraní navrženo. Dle vyjádřeného přání uživatele bude aplikace primárně používána na monitorech s úhlopříčkou více než 24 palců. Existuje však opodstatněný předpoklad, že aplikaci bude občas potřeba použít i na mobilním telefonu. Předpokládá se využití tzv. smartphone, v tuto chvíli jako příklad můžeme jmenovat telefony s operačním systémem Windows Mobile, Windows Phone, Apple iOS, Android či Symbian. Tyto mobilní telefony jsou zpravidla vybaveny dotykovým displejem s rozlišením 480x800 obrazových bodů a internetovým prohlížečem schopným zobrazit HTML webové stránky s plnou podporou JavaScriptu.

Dalším požadavkem uživatele bylo, aby v každý okamžik bylo na první pohled patrné, kde se aktuálně jednotlivá auta nacházejí. Je tedy žádoucí zobrazit všechna auta na mapě najednou.

Dále bylo požadováno, aby bylo zřetelně rozlišeno, které auto je v provozu, a které má vypnutý motor, a to nejlépe tak, aby v případě nastartování auta uživatel aplikace o této události ihned věděl.

### 4.5.2 Rozvržení uživatelského rozhraní

Na základě výše uvedených požadavků navrhujeme následující rozvržení uživatelského rozhraní aplikace:



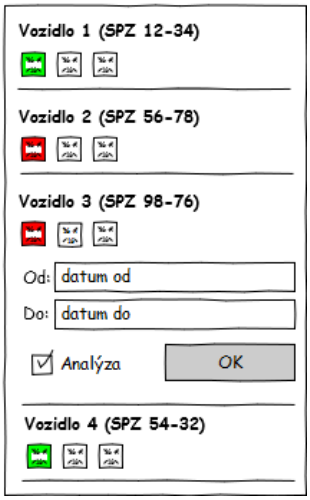

**Obrázek č. 2: Rozvržení uživatelského rozhraní**

#### Hlavní okno – MAPA

Pro potřeby této aplikace, kde se jedná především o uživatelsky přívětivé zobrazení dat, je důležité, aby co největší plochu aplikace obsadila mapa. Veškeré ovládací prvky týkající se práce s mapou jako přiblížení a oddálení, posun, případně přepnutí zobrazení jsou součástí mapových podkladů a budou tedy zobrazeny také v tomto okně.

#### Levý panel

V levém panelu budou zobrazena všechna vozidla zavedená v systému a ovládací prvky týkající se daného vozidla. Podrobnější rozložení ovládacích prvků je znázorněno na obrázku č. 3.




**Obrázek č. 3: Detail ovládacích prvků v levém panelu**

V levém panelu vidíme seznam jednotlivých vozidel oddělených horizontální čarou. U každého vozidla vidíme jeho název a SPZ, na jejichž základě lze tato vozidla od sebe jednoznačně odlišit.

První ikona mění svou barvu na základě stavu vozidla. Je-li vozidlo nastartováno, její barva je zelená a uživatel tedy na první pohled vidí, která vozidla jsou nastartována, či v pohybu. Červená ikonka značí vozidlo s vypnutým motorem. Kliknutím na tuto ikonu dojde k přiblížení mapy na polohu vybraného vozidla a zobrazení jeho poslední trasy v případě, že stojí, nebo aktuální trasy v případě, že je v pohybu.

Ikona druhá slouží k načtení historie jízd daného vozidla. Po kliknutí jsou zpřístupněny další volby znázorněné na obrázku č. 3 u třetího vozidla. Uživatel pomocí interaktivního kalendáře zvolí časová rozmezí, která ho zajímají. Pokud zaškrtne volbu „Analyzovat“, budou načtené jízdy podrobeny analýze překračování rychlosti. Po odeslání formuláře dojde k načtení jednotlivých jízd a jejich zobrazení ve spodním panelu – rozhraní pro vedoucího.

Poslední ikona slouží pro generování tabulkového výpisu jízd vybraného vozidla. Pro výpočet stavu tachometru je potřeba zadat jeho aktuální stav, zbylé údaje budou zpětně dopočteny automaticky podle naměřených a uložených dat.

## Spodní panel

Spodní panel bude sloužit pro zobrazení chronologické tabulky se seznamem jízd v požadovaném časovém rozmezí. Zobrazeny jsou údaje jako datum a čas začátku a konce jízdy, doba jízdy, průměrná rychlost, maximální dosažená rychlost a v případě, že uživatel zvolí možnost analýzy v levém panelu, je u jízd zobrazeno případné varování v situaci, kdy v dané jízdě došlo k překročení rychlosti.

## 4.6 Návrh technického řešení

Nabízí se několik alternativ. Aplikaci můžeme realizovat jako desktopovou a implementovat ji v některém z mnoha možných programovacích jazyků. Na výběr se nabízí C#, Java, C++ a další. Další možností je implementovat aplikaci jako webovou stránku zobrazitelnou v prohlížeči.

Vzhledem k požadavkům na využití na různých zařízeních postavených na různých platformách se jako vhodná jeví druhá možnost – webová aplikace. Tomu nahrává i fakt, že zobrazení mapových podkladů ve webových stránkách je nativně podporováno většinou poskytovatelů map.
Jak již bylo řečeno v předchozích kapitolách, aplikace bude využívat stávající databázi.

### 4.6.1 Výběr mapových podkladů

Nedílnou součástí aplikace jsou vhodné mapové podklady pro uživatelsky přívětivé zobrazení vozidel, ujetých tras, případně vyznačení překročení rychlosti. Na výběr máme množství variant, z nichž uvádíme ty neznámější[1]:

- **OpenStreetMap** je volně dostupný poskytovatel map, které tvoří a upravují uživatelé, kteří se do projektu dobrovolně přihlásí. To s sebou přináší výhodu v podobě široké aktivní komunity. Jistou nevýhodou těchto map je, že nikdo neručí za správnost a úplnost zobrazených map, takže se můžeme setkat s místy, která nejsou na mapě vůbec vyznačena.
- **Google Maps** – mapové podklady od světového giganta Google ve verzi 3 nabízí vylepšenou podporu zobrazení na mobilních zařízeních a dosahují vyššího výkonu oproti verzím předešlým. Nově také už není třeba se registrovat pro získání přístupového klíče.
- **Bing Maps –** mapy z rodiny Bing vytvořené společností Microsoft se vyznačují inovativní funkčností zejména pro dotykové ovládání. Pro vývoj aplikace s použitím Bing Maps je potřeba se zaregistrovat a získat *Maps Developer Account*.
- **Seznam Mapy** – mapové podklady od „Seznamu“ nabízí nově ve verzi 4 zcela přepracovaný koncept původních map. Kladou více důrazu na jednoduchost a intuitivnost ovládání.
- **Atlas Mapy** – další mapy českých tvůrců. Pro vývoj aplikací využívající tyto mapové podklady je nutné se zaregistrovat a získat přístupový klíč.

Pro výběr nejvhodnějších mapových podkladů je potřeba se zamyslet, jakou funkcionalitu od daných map očekáváme.

Vzhledem k charakteru aplikace je základním předpokladem vhodných map poskytnutí API pro využití mapových podkladů v aplikacích třetích stran. Důležitou, nikoli nutnou vlastností map by také měla být možnost přepnutí zobrazení na satelitní snímky, popřípadě zobrazení tzv. hybridní mapy, což je kombinace satelitních snímků se základní mapou. Nutnou funkcí je možnost zobrazení vlastních bodů a křivek v mapě. Pro zobrazení vlastních bodů je také vhodné, aby šlo jednoduše jako značku vložit libovolný obrázek vzhledem k požadavku na jednoznačné rozeznání jednotlivých vozidel zobrazených na mapě. Pro zjištění řádu silnice, na které se zaznamenaný průjezdní bod nachází, lze využít funkce reverzního geokódování. V neposlední řadě jsou důležitým hodnotícím kritériem také licenční podmínky jednotlivých poskytovatelů mapových podkladů.

V následujících podkapitolách si jednotlivé požadavky rozebereme podrobněji. Výsledný souhrn je uveden v tabulce č. 2.

[1] Bereme v úvahu Českou republiku.

## API pro využití mapových podkladů 3. stranou

Pro potřeby vlastní webové aplikace je potřeba API, které nám umožní použití mapových podkladů a práci s nimi i mimo jejich domovskou stránku.

- OpenStreetMap – *„Přestože OpenStreetMap tvoří svobodná data, nemůžeme zdarma poskytovat třetím stranám mapové API.“* [6]. Z uvedené citace je patrné, že OpenStreetMap žádné API pro aplikace třetích stran neposkytují.
- Google Maps – Google nabízí plně využitelné API.
- Bing Maps – Bing nabízí po registraci plně využitelné API.
- Seznam Mapy – Seznam nabízí plně využitelné API.
- Atlas Mapy – Atlas nabízí po registraci plně využitelné API.

## Možnost satelitních snímků

Možnost přepnutí mapových podkladů na satelitní snímky je důležitá v případě, že systém využívá např. stavební firma (jako např. v našem případě) a její pracovní stroje a automobily jezdí mj. v místech, kde teprve vzniká silnice a není tedy zatím na mapě.

- OpenStreetMap – mapy jsou tvořeny participujícími uživateli a nenabízí možnost přepnutí na satelitní snímky.
- Google Maps – plně poskytuje satelitní snímky pořízené vlastní družicí GeoEye-1.
- Bing Maps – možnost přepnutí satelitních snímků není podporováno.
- Seznam mapy – plně poskytuje.
- Atlas mapy – plně poskytuje.

## Vlastní značky a křivky

Umístění vlastních značek a křivek do mapového podkladu je pro tento projekt stěžejní. Je to způsob, jak zobrazit ujetou trasu či aktuální polohu vozidel uživateli. O možnostech zakreslení vlastních křivek do mapy pojednává kapitola 5.2.1.

- OpenStreetMap – umožňuje vložení jedné značky bez možnosti změny její ikony. Vložení křivky umožňuje pouze registrovaným uživatelům jako tzv. GPS stopu importem souboru ve formátu GPX.
- Google Maps – umožňuje vložit neomezené množství značek včetně možnosti nastavení vlastní ikonky. Vkládání křivek je umožněno buď importem souboru ve formátu KML, nebo také jako vlastní křivku dle zadaných bodů.
- Bing Maps – umožňuje vložit vlastní značky včetně možnosti vlastní ikonky. Vložení křivky je možné pomocí souboru ve formátu GPX nebo vlastní křivkou.
- Seznam mapy – umožňuje vložení značek včetně možnosti definovat vlastní ikonku. Vkládání křivek je umožněno buď importem souboru ve formátu GPX, nebo KML, ale také jako vlastní křivku dle zadaných bodů.
- Atlas mapy – umožňují vložit neomezené množství značek včetně možnosti definovat vlastní ikonku. Trasu je možno zakreslit pouze jako vlastní vektorovou křivku.

## Reverzní geokódování

Geokódování je proces, při kterém se na základě katastrálního umístění (adresa, město, PSČ, apod.) vyhledají zeměpisné souřadnice. Pro účely aplikace je nutné využít reverzního geokódování, kde pomocí zeměpisných souřadnic zjistíme přesnou katastrální polohu místa až po úroveň názvů jednotlivých ulic. Díky tomuto systému jsme schopni zjistit, na jaké silnici se dané vozidlo nacházelo a kontrolovat tak překračování rychlosti s rozlišením úrovně silnice město/mimo město. Tato metoda má jistá omezení, jako například nemožnost rozeznání snížení povolené rychlosti mimo město pod 90 km/h nebo naopak povolení vyšší rychlosti na některých silnicích ve městě. V současné době však neexistuje žádný volně dostupný internetový zdroj rychlostních limitů pro všechny silnice v České republice.

- OpenStreetMap – podporují pouze zobrazení informací na základě výběru objektu, nikoliv podle zadaných souřadnic.
- Google Maps – podporuje po celém území České republiky do úrovně ulic.
- Bing Maps – podporuje pouze jako SOAP službu, mapové API nativně nepodporuje.
- Seznam mapy – podporuje po celém území České republiky do úrovně ulic.
- Atlas mapy – nepodporuje.

## Licenční podmínky umožňující použití map pro tento účel

V práci rozhodně nesmíme opomenout licenční podmínky jednotlivých variant mapových podkladů.

- OpenStreetMap – jsou vydávány pod licencí „Creative Commons Uveďte autora - Zachovejte licenci“ (CC-BY-SA 2.0)[2], která umožňuje libovolné použití těchto mapových podkladů (včetně úpravy) pod podmínkou uvedení jejich zdroje.
- Google Maps – umožňuje libovolné využití mapových podkladů pro nekomerční účely. Komerční využití je umožněno pouze držitelům *Google enterprise licence*.
- Bing – umožňuje libovolné využití pro nekomerční použití. Komerční použití je umožněno nákupem *Bing Maps for Enterprise* licence.
- Seznam – Licenční podmínky uvádí: „*Software nesmí být použit pro zobrazení, hledání, navigaci a zobrazení tras vozidel, vlaků, lodí, letadel a jiných dopravních a přepravních prostředků*“ [7]. Bohužel tato citace přesně vyjadřuje účel použití naší aplikace, což činí tyto mapy licenčně nevyhovující.
- Atlas mapy – umožňuje libovolné využití mapových podkladů pro nekomerční účely. Komerční využití možno pouze se souhlasem Centrum Holdings s.r.o.

[2] Podrobnosti o licenci CC BY-SA 2.0 jsou uvedeny na adrese http://creativecommons.org/licenses/by-sa/2.0/deed.cs

### 4.6.2 Vyhovující mapové podklady

Rozbor požadavků na mapové podklady z předchozí podkapitoly je shrnut v následující tabulce.

| | API pro využití mapových podkladů 3. stranou | Možnost satelitních snímků | Vlastní značky a křivky | Reverzní geokódování | Licenční podmínky umožňující použití map pro tento účel |
|---|---|---|---|---|---|
| **OpenStreetMap** | NE | NE | NE | NE | ANO |
| **Google Maps** | ANO | ANO | ANO | ANO | ANO |
| **Bing Maps** | ANO | NE | ANO | NE | ANO |
| **Seznam mapy** | ANO | ANO | ANO | ANO | NE |
| **Atlas mapy** | ANO | ANO | ANO | NE | ANO |

**Tabulka č. 2: Souhrn požadovaných funkcí mapových podkladů**

Ze zobrazených výsledků vyplývá, že z vybraných poskytovatelů mapových podkladů existuje pouze jediný splňující veškeré požadavky, které na mapy klademe. Aplikaci tedy postavíme na mapových podkladech *Google Maps API*.

# 5 Implementace

Aplikace bude rozdělena na serverovou a klientskou část. Poznamenejme, že část systému zodpovědná za pořízení dat a jejich uložení do databáze je mimo režii naší aplikace.

Serverová část je zodpovědná za výběr požadovaných dat z databáze, jejich sestavením do logické struktury požadované klientskou částí. Jak bylo řečeno v kapitole 4.3, serverová část nebude obsahovat replikační vrstvu. Systém tedy pracuje s jedinou databází o dané struktuře.

Serverová část se dále stará o poskytování dat v daném formátu skrze zvolený protokol. Byl použit protokol HTTP. Podrobnější informace o technické implementaci, použitých platformách a také o struktuře dat při přenosu mezi klientskou a serverovou částí se zabývá kapitola 5.3.

Klientská část má na starosti primárně zobrazení informací uživateli. Data poskytnuta serverovou částí sestaví do objektové struktury, která je vhodná pro další zpracování. Za pomoci použitých knihoven a technologií (jedná se zejména o Google Maps API a jQuery) tato data poté zobrazuje v okně prohlížeče v požadované podobě.

Další důležitou funkcí klientské části je zprostředkovat interakci uživatele s aplikací. Aplikace tímto poskytuje rozhraní pro uživatele, které mu umožní snadno zvolit, která data z databáze je třeba vybrat a zobrazit.

## 5.1 Serverová část

### 5.1.1 Zvolená platforma

Pro implementaci serverové části aplikace se nabízí různé platformy. Pro srovnání vybíráme několik z nich:

- **Microsoft ASP.NET** – platforma závislá na komerčním licencovaném operačním systému *Microsoft Windows Server*. Pro programování skriptů a aplikací se zde používají jazyky z rodiny .NET. Jedná se např. o C#, VB.NET, J# a další. Všechny sdílejí společné knihovny platformy .NET Framework. Vzhledem k faktu, že mezi vývojáři webových stránek se jedná o jednu z méně rozšířených platforem, jeví se z hlediska budoucí rozšiřitelnosti aplikace jako méně vhodná. Platformní omezení se také jeví jako limitující faktor. Vysoká cena potřebných komerčních licencí by vývoj, nasazení a provoz aplikace neúměrně zdražila.
- **Java Servlet** – třída napsaná v jazyce Java sloužící k vyhodnocení požadavku a zaslání odpovědi přes protokol HTTP. Pro jeho běh na serveru je zapotřebí mít nainstalován *Apache Tomcat Server* a samozřejmě také *Java SDK*. Java patří k nejrozšířenějším programovacím jazykům, takže z hlediska budoucího rozšíření je v tomto ohledu vhodná.
- **PHP** – skriptovací jazyk, oblíbený především jako nástroj pro tvorbu dynamických webových stránek. Z tohoto důvodu je také značně rozšířen mezi webovými vývojáři. Mnohé hostingové společnosti nabízejí PHP jako primární platformu na svých serverech. Často se setkáme také s hostingem zdarma, opět nabízející využití PHP.

Z výše uvedených platforem bylo zvoleno PHP. Je vhodné pro malé až střední projekty a mám s touto platformou již mnohaleté zkušenosti.

Skripty programované v jazyce PHP je možné spustit na více platformách. V tomto případě je zvolen operační systém *GNU Linux*, distribuce *CentOS 5*. Jako webový server poskytující služby spojené s provozem HTTP je použit *Apache 2.0*. Pro zpracování skriptů je nainstalováno PHP ve verzi 5.3.3.

### 5.1.2 Přístup k databázi

Při přístupu k databázi využívá serverová část databázovou vrstvu Dibi[3]. Jedná se o knihovnu funkcí, které zastřešují funkcionalitu

- Připojení k databázi.
- Použití tzv. Lazy loadingu – nahrávání dat z databáze až ve chvíli, kdy jsou opravdu zapotřebí.
- Zasílání SQL dotazů databázi.
- Ošetření vstupů.
- Získání výsledků a jejich zpracování do podoby, ve které s nimi může aplikace snadno dále pracovat.

Použití databázové vrstvy Dibi usnadnilo práci a učinilo aplikaci nezávislou na použité databázové platformě, jelikož umožňuje jednotný zápis SQL dotazů pro různé databázové platformy. Považujme to za správné rozhodnutí i z hlediska bezpečnosti a snahy minimalizovat výskyt programátorských chyb. Zatímco při výběru dat klasickou metodou je potřeba vstupy ošetřit, Dibi vše udělá za nás.

Rozdíl v zápisu výběru dat z databáze a jedním z možných zápisů s použitím knihovny Dibi je patrný z ukázky zdrojového kódu č. 1. Jak už bylo řečeno, díky databázové vrstvě Dibi jsme se stali nezávislými na databázové platformě, jelikož syntaxe zápisu je jednotná. O přeformátování zápisu a použití potřebných metod se stará samotná knihovna.

Pokud by došlo ke změně databázové platformy, tak při použití klasické metody musíme přepsat všechny volané funkce na jejich ekvivalenty pro jinou databázovou platformu a upravit styl zápisu SQL dotazu, aby korespondoval s cíleným databázovým nástrojem. Při použití Dibi se změna týká pouze přihlašovacích údajů a zvoleného databázového driveru. Dříve použité metody pro výběr dat zůstanou zachovány.

Identifikátor připojení k databázi je v Dibi statický, proto stačí použít připojení k databázi kdekoliv v kódu a nemusíme se jím už dále zabývat. Zatímco klasická metoda jej vyžaduje u každého databázového dotazu uvést v parametru.

[3] Dibi – Database Abstraction Library for PHP 5. Nette Foundation, http://dibiphp.com/cs/

```php
<?php
// výběr dat standardní cestou
$database = 'localhost:/path/to/your.gdb';
$username = 'username';
$password = 'password';
$db = ibase_connect($database, $username, $password);
$query = ibase_query($db, 'SELECT * FROM table WHERE row1 = 5 ORDER BY
row2');
$result = ibase_fetch_object($query);

// výběr dat pomocí Dibi
dibi::connect(
    array(
      'driver'    => 'firebird',
      'database'  => 'localhost:/path/to/your.gdb',
      'host'      => 'localhost',
      'username'  => 'username',
      'password'  => 'password'
    )
);
$result = dibi::select("*")->from("table")->where("row1 = %d",5)
->orderBy("row2")->fetch();
```

**Zdrojový kód č. 1: Rozdíl v připojení k databázi a výběru dat mezi Dibi a klasickou metodou**

Databázovou vrstvu Dibi je možné používat v souladu s New BSD licencí nebo GNU General Public License (GPL) ve verzi 2 nebo 3. To umožňuje prakticky neomezené využití jak pro komerční aplikace, tak i v rámci akademické práce.

Jak již bylo zmíněno v úvodu této kapitoly, aplikace využívá existující databázi a jako nejvhodnější řešení se jevilo použít tuto databázi bez změn její struktury. Podrobnější informace o struktuře databáze, jednotlivých tabulkách a o platformě, na které je databáze založena, jsou uvedeny v kapitole 4.3.2.

### 5.1.3 Třídy a funkce

Vzhledem k funkcionalitě, kterou serverová část naší aplikace zajišťuje (ta je popsána v předchozích podkapitolách), vystačíme si s několika málo třídami.

Třída `Vehicles` se stará o načtení seznamu vozidel z databáze pomocí metody `loadVehicles()`. Metoda `getOnlineData()` třídy `Online` slouží ke zjištění aktuální polohy všech vozidel. Toto zjišťování probíhá v našem případě pravidelně v intervalu 30 sekund.

Třída `Trace` zjišťuje poslední jízdu konkrétního vozidla podle zadaného `ID`. Ve třídě Ride se setkáme s metodou `loadRide()`, která na základě zadaných parametrů `$fromDate` (datum od) a `$toDate` (datum do) vybere z databáze jízdy konkrétního vozidla v daném časovém rozmezí. Třída `Conversions` je využívána všemi ostatními třídami a obsahuje pouze pomocné funkce jako převod formátu data nebo zaokrouhlení čísla na nejbližší násobek zadaného čísla.

Jak je tedy z informací uvedených v předchozích odstavcích patrné, serverová část je implementačně jednoduchá. Zajišťuje jen výběr požadovaných dat z databáze a jejich uzpůsobení formě vyžadované klientskou částí.

## 5.2 Klientská část

Klientská část slouží především k zobrazování vybraných dat uživateli. Implementace grafického uživatelského rozhraní vychází z návrhu popsaného v kapitole 4.5. Z přirozené podstaty vyvíjené aplikace vyplývá potřeba interakce s uživatelem. Je tedy patrné, že se jedná o dynamickou webovou stránku. Tomu bylo potřeba přizpůsobit i výběr platformy, na které je klientská část založena.

Pro naprogramování klientské části aplikace se výběr možných platforem zužuje. Bylo zvoleno použití mapových podkladů Google, které jsou dodávány spolu s Google Maps API (podrobnější informace o výběru vhodného poskytovatele mapových služeb jsou uvedeny v kapitole 4.6.1). API poskytuje funkcionalitu v jazyce JavaScript, proto použití jiného jazyka by bylo jednoznačně nevýhodné – v některých případech by to znamenalo výkonnostní problémy či řádově náročnější implementaci.

### 5.2.1 Možnosti zakreslení trasy do mapy

Pro zobrazení ujeté trasy v mapě se nabízí několik variant.

#### Import formátu GPX

GPX[4] je univerzální formát zápisu GPS dat (body zájmu, trasy, aj.) pomocí XML struktury. Tento formát je podporován množstvím zařízení, aplikaci či webových služeb.

```
<gpx version="1.0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://www.topografix.com/GPX/1/0"
 xsi:schemaLocation="http://www.topografix.com/GPX/1/0
http://www.topografix.com/GPX/1/0/gpx.xsd">
      <time>2002-02-27T17:18:33Z</time>
      <bounds minlat="42.401051" minlon="-71.126602"
              maxlat="42.468655" maxlon="-71.102973"/>
      <wpt lat="42.438878" lon="-71.119277">
            <ele>44.586548</ele>
            <time>2001-11-28T21:05:28Z</time>
            <name>5066</name>
            <desc><![CDATA[5066]]></desc>
            <sym>Crossing</sym>
            <type><![CDATA[Crossing]]></type>
      </wpt>
      <wpt lat="42.439227" lon="-71.119689">
      ...
```

**Zdrojový kód č. 2: Ukázka formátu GPX**

[4] Oficiální XSD schéma formátu GPX: http://www.topografix.com/GPX/1/1/gpx.xsd

Přímá podpora formátu GPX u vybraného Google Maps API chybí. Existují sice knihovny pro převod do formátu KML (podrobnosti viz dále), ale toto řešení by bylo zbytečně neefektivní a dávalo by větší prostor možnému výskytu programátorských chyb.

## Import formátu KML

KML[5] je formát zápisu GPS dat, rovněž pomocí XML struktury. Narozdíl od formátu GPX umožňuje definovat v mapě prostorové 3D objekty, umožňuje stanovit orientaci mapy, popřípadě pohled kamery při využití Stret-view. Formát KML je navržen společností Google a je použitelný v aplikaci Google Earth nebo Google Maps. [8]

```
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Document>
    <name>KML Samples</name>
    <Folder>
      <name>Placemarks</name>
      <description>Placemarks example</description>
      <Placemark>
        <name>Floating placemark</name>
        <visibility>0</visibility>
        <description>Floats above the ground.</description>
        <LookAt>
          <longitude>-122.0839597145766</longitude>
          <latitude>37.42222904525232</latitude>
          <altitude>0</altitude>
          <heading>-148.4122922628044</heading>
          <tilt>40.5575073395506</tilt>
          <range>500.6566641072245</range>
        </LookAt>
        <Point>
          <altitudeMode>relativeToGround</altitudeMode>
          <coordinates>-122.084075,37.4220033612141,50</coordinates>
        </Point>
      </Placemark>
      <Placemark>
      ...
```

**Zdrojový kód č. 3: Ukázka formátu KML**

Pro použití v naší aplikaci by se tento formát sice hodil, ale po importu do google map pomocí příkazu `new google.maps.KmlLayer('path/to/kml.kml')` se plně integruje do mapy a další práce s křivkami a body je velice obtížná.

- **Vlastní křivka (polyline) pomocí zadaných bodů**
  Vložení křivky je jednou z funkcí nabízených Google Maps API a díky pohodlné další práci se všemi „ručně“ definovanými objekty (křivky, body) byly vybrány jako vyhovující způsob

[5] Oficiální XSD schéma formátu KML: http://code.google.com/intl/cs/apis/kml/schema/kml21.xsd

zakreslení trasy do mapy. Podrobnější informace o zakreslení bodů nebo křivek do mapy jsou uvedeny v následující podkapitole.

### 5.2.2 Google maps API

Hlavní částí uživatelské aplikace je zobrazení mapy. K tomu využíváme již zmíněné Google Maps API.

V aplikaci s mapovým podkladem provádíme několik základních činností:

- Zobrazení mapy.
- Pohyby mapy, přibližování.
- Zobrazení vlastní značky.
- Zobrazení křivky v mapě.
- Zobrazení informační bubliny.
- Zjišťování typu silnice v jednotlivých bodech.

#### Zobrazení mapy

Pro zobrazení mapy je zapotřebí definovat v HTML blokový element, nastavit mu výšku a přiřadit mu identifikátor atributem `id`. Dále definuji JavaScriptové pole s volbami, které jsou pro inicializaci map potřebné. Referenci na element a pole s nastavením poté předám konstruktoru třídy `Google.maps.Map()`, jak je uvedeno v příkladu zdrojového kódu č. 4.

```
<script type="text/javascript">
// střed mapy
var myLatlng = new google.maps.LatLng(49.6780739, 18.358027);
// předvolby pro mapu
var myOptions = {
      zoom: 10,   //stupeň přiblížení
      center: myLatlng,
      streetViewControl: false,
      mapTypeId: google.maps.MapTypeId.ROADMAP
}
// vytvoření mapy
var map = new google.maps.Map(document.getElementById('map'), myOptions);
</script>
<div id="map" style="height: 100%;"></div>
```

**Zdrojový kód č. 4: Zobrazení mapy**

V předvolbách se nastaví požadované přiblížení mapy, souřadnice jejího středu jako objekt třídy `google.maps.LatLng()`, která očekává v parametru zeměpisnou šířku a délku ve formátu Decimal Degrees. Pro projekt je bezpředmětné využití StreetView[6], proto jej zakážeme. Jako základní zobrazení mapy použijeme klasickou mapu, označenou jako `ROADMAP`. Pomocí funkce `document.getElementById()` vytvoříme referenci na náš HTML element, ve kterém chceme

[6] Více o Google StreeView na http://maps.google.com/help/maps/streetview/index.html

mapu vykreslit a spolu s nastavením jej předáme konstruktoru `Google.maps.Map()`. Tím bylo dosaženo zobrazení mapy v naší stránce.

## Zobrazení vlastní značky

Vlastní ikonu můžeme ponechat buď v implicitní podobě, nebo použít jakýkoliv bitmapový obrázek. Důležité je specifikovat polohu, kde se má značka zobrazit a samozřejmě předat instanci naší zobrazené mapy.

```
// pozice značky
var myLatlng = new google.maps.LatLng(49.6780739, 18.358027);

// vložení nové značky
var marker = new google.maps.Marker({
        position: myLatlng,
        map: map,
        icon: new google.maps.MarkerImage('path/to/image.png'); //volitelně
        title: "Hello World!"
});
```

**Zdrojový kód č. 5: Vložení vlastní značky do mapy**

Volba `icon` je nepovinná, slouží k vložení vlastního obrázku namísto standardního. Titulek `title` se zobrazí při najetí myší na značku.

## Zobrazení křivky v mapě

Křivka je definovaná jako soubor bodů spojených čárou. Kromě bodů jí můžeme nastavit barvu, šířku a průhlednost. Jednotlivé body, kterými prochází, jsou uloženy ve struktuře `MVCArray`, která rozšiřuje běžné pole o sadu definovaných funkcí.

```
// pole souřadnic pro křivku
var coords = new google.maps.MVCArray([
        new google.maps.LatLng(37.772323, -122.214897),
        new google.maps.LatLng(21.291982, -157.821856),
        new google.maps.LatLng(-18.142599, 178.431),
        new google.maps.LatLng(-27.46758, 153.027892)
]);
var line = new google.maps.Polyline({
        path: coords,
        map: map,
        strokeColor: "#FF0000", //barva
        strokeOpacity: 1.0,     //průhlednost
        strokeWeight: 2         //šířka čáry
});
```

**Zdrojový kód č. 6: Vložení křivky do mapy**

Výhodou použití `MVCArray` je, že lze do něj přidávat nové prvky za běhu aplikace pomocí příkazu `coords.push(LatLng Object)` a nový bod se automaticky přidá k již zobrazené křivce a dojde k jejímu překreslení.

## Zobrazení informační bubliny

Informační bublina je vhodný prvek ke zobrazení bližších podrobností o nějaké vlastní značce v mapě. Zobrazení této bubliny je nutné vždy navázat na nějakou událost, například kliknutí na vlastní značku.

```
// objekt infobubliny
var infowindow = new google.maps.InfoWindow({
        content: '<p>Any HTML goes here</p>'
});
// pomocná značka, na kterou navážeme infobublinu
var marker = new google.maps.Marker({
        position: myLatLng,
        map: map
});
// po kliknutí na značku se zobrazí infobublina
google.maps.event.addListener(marker, 'click', function() {
        infowindow.open(map, marker);
});
```

**Zdrojový kód č. 7: Vložení informační bubliny do mapy**

Informační bublina může obsahovat jakýkoliv HTML kód, včetně různých typů nadpisů, tabulek případně odkazů či obrázků. Vytvořili jsme si v mapě vlastní značku a při události `click` na tuto značku dojde k zavolání metody `infowindow.open()`, která na zadané mapě, se středem v zadaném bodě zobrazí naší informační bublinu.

## Zjištění typu silnice v zadaném bodě

Jak bylo zmíněno v kapitole 4.6.1, pro zjištění typu silnice nám poslouží služba reverzní geokódování. To je zakomponováno přímo v Google Maps API.

```
var geocoder = new google.maps.Geocoder();
var latlng = new google.maps.LatLng(49.6780739, 18.358027);

geocoder.geocode({'latLng': latlng}, function(results, status) {
      if (status == google.maps.GeocoderStatus.OK) {
            // USE results[0].types
            ...
      } else {
        alert("Geocoder failed due to: " + status);
      }
});
```

**Zdrojový kód č. 8: Použití reverzního geokódování**

Metoda `geocoder.geocode()` očekává v parametru objekt `LatLng`, který určuje bod, pro který chceme zjišťovat typ silnice. Druhým parametrem je callback kde nám `Geocoder` vráti vícerozměrné pole s výsledky. Čím nižší index pole `results`, tím podrobnější popis silnice dostaneme. Očekávané výsledky v poli `results[0].types` jsou buď `street_address` pro ulici ve městě, nebo `route` identifikující silnici mimo město. Bohužel už nelze touto metodou rozeznat jednotlivé typy komunikací mimo město (silnice 1. třídy, silnice pro motorová vozidla, dálnice, …). Pro tento projekt je však rozeznání na úrovni ve městě/mimo město dostačující.

### 5.2.3 Použité knihovny

Klientská část aplikace je napsána z velké části v jazyce JavaScript. Pro zjednodušení některých operací byla použita JavaScriptová knihovna jQuery.

> *„jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is designed to change the way that you write JavaScript.“* [9]

Tato knihovna je použíta především pro zjednodušení práce s AJAXovými požadavky a práce s HTML prvky stránky. Pro sestavení layoutu stránky je využito pluginu `jQuery UI.Layout`[7].

```
// vložení věty "Hello world!" do poslední buňky každého řádku tabulky
var table = document.getElementById("table");
var rows = [ table ];
for (var i = 0; i < table.getElementsByTagName("tr").length; i++)
      rows[rows.length] = table.getElementsByTagName("tr")[i];
for (var i = 0; i < rows.length; i++){
      var item = rows[i].lastChild;
      while (!item.tagName || item.tagName.toLowerCase() != "td"){
            item = item.previousSibling;
            item.innerHtml = "Hello World!";
      }
}

// stejná operace s použitím jQuery
$("#table td:last-child").html("Hello World!");
```

**Zdrojový kód č. 9: Rozdíl v zápisu kódu bez použití a s použitím jQuery**

Z výše uvedeného zdrojového kódu je patrný rozdíl ve složitosti zápisu kódu s využitím knihovny jQuery a bez ní. Knihovnu jQuery lze použít v souladu s licencí MIT nebo GNU GPL verze 2. Pro tento projekt je tedy její použití vhodné i výhodné.

### 5.2.4 Výsledná aplikace

Výsledná implementace klientské části, která je podrobně popsána v předchozích kapitolách, je vyobrazena v příloze 1. Znázorňuje vzhled uživatelského rozhraní aplikace v situaci, kdy uživatel zvolil historii jízd v zadaném rozmezí a nechal si tuto historii analyzovat. Na první pohled je patrné, ve kterých jízdách došlo k překročení rychlosti a při zvolení konkrétní jízdy je tato vykreslena do mapy s barevným odlišením míst, ve kterých došlo k překročení rychlosti. Dále je také vidět, které vozidla jsou v danou chvíli nastartována.

[7] Podrobnější informace dostupné na http://layout.jquery-dev.net/index.cfm

## 5.3 Přenos dat mezi serverovou a klientskou částí

Vzhledem k množství dat, které databáze uchovává je nereálné, aby se všechna načetla při prvním spuštění aplikace. Řádově statisíce až miliony záznamů by jednak znamenaly neúměrně vysoký datový přenos, ale hlavně by zbytečně zahltily klientskou aplikaci. Proto se jako vhodná metoda jeví načítáni informací asynchronně až ve chvíli, kdy o ně uživatel požádá. K načítání těchto dat využijeme technologii AJAX, která je dnes velice populární a málokterá webová aplikace se bez ní obejde.

### 5.3.1 Způsoby přenosu dat

Serverová část aplikace využívá jinou platformu než klientská. Z tohoto důvodu je nutné použití nějakého univerzálního formátu dat, který půjde jednoduše na obou platformách zpracovat. Na výběr máme formát XML nebo formát JSON.

#### Formát XML

XML je velmi rozšířený značkovací jazyk určen pro uložení datových struktur pomocí textu. Popisuje pouze strukturu dat, nezabývá se jejich vzhledem. Pro vytvoření struktury dat se využívá množina speciálních symbolů, které se nazývají *značky* nebo také *tagy*.

```
<widget>
    <window title="Sample Widget">
        <name>main_window</name>
        <width>500</width>
        <height>500</height>
    </window>
    <image src="images/Sun.png" name="sun1">
        <alignment>center</alignment>
    </image>
    <text data="Click Here" size="36" style="bold">
        <name>text1</name>
        <alignment>center</alignment>
        <onMouseUp>
            sun1.opacity = (sun1.opacity / 100) * 90;
        </onMouseUp>
    </text>
</widget>
```

**Zdrojový kód č. 10: Ukázka formátu XML**

#### Formát JSON

Formát JSON umožňuje zápis libovolné datové struktury, kde se pro její vyznačení nepoužívají značky jako v případě XML, ale pouze různé typy závorek. Ve skutečnosti je JSON serializovaný JavaScriptový objekt. Jeho využití v JavaScriptových aplikacích je proto vhodné.

```
{"widget": {
    "window": {
        "title": "Sample Widget",
        "name": "main_window",
        "width": 500,
        "height": 500
    },
    "image": {
        "src": "images/Sun.png",
        "name": "sun1",
        "alignment": "center"
    },
    "text": {
        "data": "Click Here",
        "size": 36,
        "style": "bold",
        "name": "text1",
        "alignment": "center",
        "onMouseUp": "sun1.opacity = (sun1.opacity / 100) * 90;"
    }
}}
```

**Zdrojový kód č. 11: Ukázka formátu JSON**

V příkladech zdrojových kódů č. 10 a 11 je použit stejný zápis pro demonstraci rozdílu mezi XML a JSON. Z výše uvedených formátů byl zvolen formát JSON vzhledem k jednoduchosti jeho použití na obou použitých platformách jak na serverové části, tak i na klientské.

## 5.4 Problémy při implementaci

Největším problémem, který se při implementaci klientské části vyskytl, byla kompatibilita se staršími verzemi prohlížečů. Vzhledem k tomu, že aplikace obsahuje relativně složitou práci s DOM stromem, narážejí některé starší verze prohlížečů na problémy s výkonem a rychlostí JavaScriptového interpretu.

Další problém vyvstal při testování analýzy historie jízd, kdy bylo zjištěno, že Google API na své serverové straně limituje počet požadavků za jednotku času. Přesné limity nejsou pevně stanoveny, experimentováním jsme zjistili, že lze provést řádově maximálně desítky požadavků za minutu, přičemž je potřeba, aby mezi jednotlivými požadavky bylo rozmezí v řádech sekund. Tento problém byl vyřešen tím, že mezi jednotlivými požadavky aplikace vždy přibližně dvě vteřiny nečinně čeká. To způsobilo znatelné zpomalení analýzy historie jízd. Vzhledem k tomu, že poskytovatel dat pro reverzní geokódování neumožňuje tyto limity překročit, nebylo možné se tohoto nedostatku zbavit.

V serverové části bylo největší komplikací zjišťování funkcí jednotlivých sloupců tabulek existující databáze (podrobněji popsáno v kapitole 4.3).

# 6 Nasazení

Vycházejme z faktu, že aplikace je vyvíjena přímo v prostředí firmy, jejíž zaměstnanci tak mohou i v průběhu vývoje poskytovat cennou zpětnou vazbu. Při vývoji byl použit iterativní model, kdy každá spustitelná verze aplikace byla nasazena do provozu takřka okamžitě, a zpětná vazba od uživatelů tak byla důležitou součástí procesu.

Charakteristikou iterativního modelu je rozdělení procesu vývoje do iterací [4, kap. 3.3.2].

> *„Výhodou iterativního modelu je možnost upřesňovat požadavky uživatelem, aniž to naruší životní cyklus softwaru.“ [4, str. 24]*

Nevýhodou zvolené metodologie je, že bylo občas nutné přepracovat i již dříve hotové celky, případně upravit původně zamýšlenou strukturu aplikace. Pokud by byly všechny požadavky specifikovány předem, pravděpodobně bychom byli schopni dosáhnout lepšího návrhu.

Na začátku práce byla sestavena analýza požadovaného cílového stavu. Ta je podrobně popsána v kapitole 3. Obsahuje mimo jiné především požadavky na funkcionalitu a vlastnosti systému v okamžiku plného nasazení. Aby bylo možné zjistit, do jaké míry se podařilo tyto požadavky naplnit, byl sestaven dotazník, který byl poté rozdán jednotlivým zaměstnancům, kteří v budoucnu přijdou s naší aplikací do styku. Kompletní znění dotazníku je uvedeno v příloze 2.

Vyhodnocením dotazníků jsme došli k následujícím závěrům:

- Aplikaci využívají lidé ve vedení firem popřípadě pověření správci autoparku.
- Aplikace je využívána na různých zařízeních – mobilních telefonech, počítačích.
- Primární využití aplikace je pro zjišťování aktuální polohy vozidel.
- Ovládání aplikace je intuitivní a všichni dotázaní byli schopni ji používat bez asistence další osoby.
- Analýza jízd probíhala výrazně pomalu. Tato skutečnost je vysvětlena v kapitole 5.4.
- V některých případech zůstala na mapě zobrazena jízda, která nesouvisela s právě prováděnou operací.

Aby se daly lépe ilustrovat kladné stránky, či naopak slabiny předkládané aplikace, budou v následujících několika odstavcích popsány příklady vybraných situací, ke kterým došlo v průběhu prvních dnů či týdnů provozu:

- Při testování aplikace v pozdních večerních hodinách uživatel zpozoroval, že jedno z vozidel se dalo do pohybu, aniž by k tomu byl opodstatněný důvod. Zpočátku se uživatel domníval, že se jedná o chybu systému. Další den vyšlo najevo, že vozidlu (jednalo se o velký stavební bagr) byla odcizena nafta z nádrže. Díky faktu, že vozidlo bylo monitorováno, bylo možné dokázat, že odcizení má na svědomí zaměstnanec, který měl přístup ke startovacím klíčům. Při dalším vyšetřování se ukázalo, že stroj stál víčkem nádrže u zdi, bylo tedy nutné s ním několik metrů popojet, aby se zloděj k nádrži dostal.
- Všechna vozidla byla pracovně v terénu. Jeden ze zaměstnanců ve stejnou chvíli nutně potřeboval vozidlo. Díky tomu, že měl přístup k aplikaci, mohl rychle zjistit, které ze služebních vozidel je poblíž a koho je potřeba kontaktovat.

- Stavební dělník si půjčil služební stavební bagr s domluvou, že může na svém vlastním pozemku provést drobné terénní úpravy. Systém vyhodnotil, že stroj byl v provozu několik dní v kuse a najezdil při tom přes 150km. Bylo tak možné dokázat, že se jednalo o nepovolené využití firemních prostředků.
- Ve chvíli, kdy zaměstnanec, který je s monitorováním vozidla obeznámen, odpojí monitorovací zařízení, není možné dokázat, zda tak opravdu učinil, nebo zda došlo ke ztrátě signálu či poruše zařízení. Po tuto dobu může vozidlo užívat libovolně, aniž by vedoucí měli jakoukoliv informaci o jeho poloze či provozu.

# 7 Možnosti rozšíření

Aplikaci nelze považovat za konečný produkt. Vzhledem k použité databázové platformě a nevhodnému návrhu struktury databáze (viz kapitola 4.3.2) je pro další rozšiřování aplikace nutné tyto oblasti vylepšit. To by s sebou neslo také nutnost přeprogramovat službu, která zachytává příchozí data ze zařízení v automobilech, aby ukládala data do nově navržené databáze.

Aplikaci lze poté rozšířit v oblastech zmíněných v kapitole 3, tj. o rozšířené statistické informace o jízdách, případně grafy rychlostí apod. Dalším vhodným rozšíření je elektronická kniha jízd, která by plně nahrazovala klasickou papírovou. Zde je potřeba zapracovat na možnosti sloučení více krátkých jízd do jedné, protože mnohdy není nutné vést jednotlivé malé pojížďky po městě, ale lze je zapsat jako jednu jízdu. Samozřejmostí by byla možnost vytisknout knihu jízd v zákonem stanoveném formátu za vybrané období.

Dalším místem pro vylepšení jsou hardwarová zařízení namontována v automobilech. Užitečná může být možnost přepínání jízd služební/soukromá případně identifikace jednotlivých řidičů, například pomocí osobního čipu. Pro měření spotřeby vozidel je nutné do vozidel nainstalovat čidla průtoku paliva, případně senzor hladiny paliva v nádrži.

Současná zařízení v automobilech také nemají ošetřeny situace, kdy uživatel vozidla toto zařízení odpojí. Řešením by byla záložní baterie umístěná přímo v zařízení, která by ihned po jeho odpojení poskytla dostatek energie pro odeslání informace o odpojení napájení.

Pro minimalizaci problému popsaného v kapitole 5.4 by se mělo provádět zjišťování typu silnice již při ukládání dat do databáze. Časové rozestupy mezi příchody dat jsou dostatečně velké, aby zde nedocházelo k limitování požadavků ze strany poskytovatele mapových podkladů.

V neposlední řadě by také měla být optimalizována aplikace v závislosti na použitém zobrazovacím zařízení. Na mobilním telefonu se lze vzdát některých prvků vzhledem ke snížení datového přenosu.

# 8 Závěr

Jednou z činností, které charakterizují vedení většiny současných firem, je snaha snižovat náklady a zvyšovat produktivitu práce. V současné době světové hospodářské krize se snižování nákladů, obzvláště pak provozních nákladů, stává ještě důležitější. Nižší provozní náklady činí firmu efektivnější a konkurenceschopnější. Investice do systémů, které tyto náklady snižují, se tedy firmám obvykle vyplácí.

Jednou z oblastí, kde firmám vznikají relativně velké provozní a režijní náklady, je správa a provoz vozového parku.

V rámci této bakalářské práce bylo úkolem zpracovat systém, který pomůže efektivněji koordinovat provoz vozidel a provádět kontrolní činnost, která by zabránila neoprávněnému užívání služebních vozidel, porušování interních pravidel či krádežím pohonných hmot.

Cílem bylo vytvořit aplikaci – Inteligentní knihu jízd, která by umožnila sledovat aktuální polohu služebních vozidel, analyzovat jejich provoz a rychlost, kterou se pohybují a evidovat jednotlivé jízdy pro účely jejich zpětné kontroly.

Aplikace vznikala v prostředí konkrétní firmy, měla by však být do budoucna obecně použitelná i v jiných firmách, jelikož úkoly spojené s provozem vozidel jsou ve všech společnostech podobné.

Pro sběr dat byl použit existující systém obsahující hardwarová zařízení a řešení pro odesílání informací o poloze a provozu vozidel z automobilu na firemní server a zápis těchto informací do databáze. Tato část systému nebyla předmětem této práce, a proto jsou v textu jen stručně zmíněny principy, na kterých pracuje.

Hlavní částí práce bylo vytvoření aplikace, která umožní uživatelsky jednoduchým způsobem mít přehled nad pohybem všech firemních vozidel a to jak v reálném čase, tak s libovolnou časovou periodou v minulosti. Všechna vozidla mají uchovánu kompletní historii jízd zobrazitelnou v uživatelsky přívětivé podobě na mapovém podkladu s možností analýzy jednotlivých jízd, pro případné zjištění spáchaných dopravních přestupků (překračování povolené rychlosti) zaměstnanci užívajícími tato vozidla.

Důraz byl kladen na snadné užívání, použitelnost na zařízeních různého druhu – stolních počítačích, noteboocích s malými displeji, mobilních telefonech. Z toho důvodu byla pro implementaci zvolena forma webové aplikace. Výsledné řešení lze snadno používat na osobních počítačích různých platforem, mobilních telefonech s operačními systémy Windows Mobile či Android.

Při návrhu a programování jsme se potýkali s problémy především při výběru vhodných technologií. Další komplikace přinášela častá rozhodnutí ohledně toho, v jakých oblastech použít existující části systémů, a v jakých bodech tyto části upravit, či je vůbec do systému nezapojovat. Fakt, že jsme v projektu použili existující databázi bez možnosti jejich úprav, byl pro výslednou implementaci značně limitující.

Pro zobrazení dat na mapě jsme brali v úvahu několik poskytovatelů mapových podkladů, které byly podrobeny průzkumu, zda splňují požadavky, které na ně klademe. Všem požadavkům vyhověl pouze jediný poskytovatel map – Google Maps.

Vytvořený systém byl nasazen v prostředí konkrétní společnosti. Při nasazení byli uživatelé dotázáni na zkušenosti a postřehy z práce se systémem. Výsledky tohoto malého průzkumu pomohly při vytváření plánu dalšího vývoje aplikace. Výslednou aplikaci tedy rozhodně nelze považovat za konečné řešení. Předpokládají se její další úpravy v průběhu provozu a rozšiřování funkcionality dle požadavků uživatelů. Dalším předpokladem jsou také nezbytné úpravy vedoucí k tomu, aby Inteligentní kniha jízd mohla plně nahradit klasickou, papírovou knihu jízd.

Je vhodné podotknout, že v této oblasti existují i jiné komerční systémy s podobnou funkčností. Jsou však obvykle velmi nákladné a jejich integrace s jinými existujícími systémy je nesnadná. Řešení vytvořené v rámci této bakalářské práce by mělo poskytovat veškerou požadovanou funkcionalitu jako nízkonákladová aplikace, která je nenáročná na systémové prostředky a především snadná pro uživatele.

# Literatura


[1] Firebird Foundation Incorporated. *Firebird - The RDBMS that's going where you're going* [online]. 2001 [cit. 2011-05-08]. About Firebird. Dostupné z WWW: <http://www.firebirdsql.org/index.php?id=about-firebird&nosb=1>.

[2] Comparison of relational database management systems. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 4.3.2005, last modified on 7.5.2011 [cit. 2011-05-08]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Comparison_of_relational_database_management_systems>.

[3] Firebird Foundation Incorporated. *Firebird - The RDBMS that's going where you're going* [online]. 2001 [cit. 2011-05-08]. Supported platforms. Dostupné z WWW: <http://www.firebirdsql.org/index.php?op=files&id=engine>.

[4] Křena, B., Kočí, R.: *Úvod do softwarového inženýrství IUS : Studijní opora*. Brno : vl.n., 2010. 103 s.

[5] Kysela, J.: *Internet pro všechny* [online]. 1. Březen 2010 [cit. 2011-04-18]. Mobilní Internet v České republice – kompletní přehled. Dostupné z WWW: <http://www.internetprovsechny.cz/mobilni-internet-v-ceske-republice-kompletni-prehled/>.

[6] *OpenStreetMap : Otevřená wiki-mapa světa* [online]. 2005 [cit. 2011-05-08]. Autorská práva a licence. Dostupné z WWW: <http://www.openstreetmap.org/copyright>.

[7] *API Mapy.cz* [online]. c2006 [cit. 2011-05-08]. Licenční ujednání a vytvoření klíče. Dostupné z WWW: <http://api.mapy.cz/keygen>.

[8] *Google Code : KML* [online]. c2011 [cit. 2011-05-08]. KML Tutorial. Dostupné z WWW: <http://code.google.com/intl/cs/apis/kml/documentation/kml_tut.html>.

[9] *JQuery : Write Less, Do More* [online]. c2010 [cit. 2011-05-04]. Dostupné z WWW: <http://jquery.com/>.

[10] Zendulka, J., Rudolfová, I.: *Databázové systémy IDS : Studijní opora*. Brno : vl.n., 2006. 217 s.

# Seznam zkratek

AJAX – Asynchronous JavaScript and XML
API – Application Programming Interface
DOM – Document Object Model
GPRS – General packet radio service
GPS – Global Positioning System
GPX – GPS Exchange Format
GSM – Global System for Mobile communication
HTTP – Hypertext Transfer Protocol
JSON – JavaScript Object Notation
KML – Keyhole Markup Language
PHP – Hypertext Preprocessor
SPZ – Státní Poznávací Značka
SQL – Structured Query Language
XML – Extensible Markup Language

# Seznam příloh

1. Screenshot výsledné aplikace (na samostatném listu)
2. Dotazník pro uživatele aplikace
3. CD obsahující zdrojové kódy klientské aplikace

# Příloha 2

## Dotazník uživatelům aplikace
## Inteligentní kniha jízd

- Uživatelské informace
  - Jaké je Vaše postavení, jaké jsou Vaše úkoly (v kontextu předkládané aplikace)?
  - K jakému účelu aplikaci primárně používáte?
  - Jaké využíváte zařízení?
  - Jaký využíváte prohlížeč?
  - Kolik hodin jste prací s aplikací strávili?

- Návrh uživatelského rozhraní
  - Porozuměli jste všem ovládacím prvkům během prvních 10 minut používání aplikace?
  - Jak dlouho bylo nutné, aby Vám použití aplikace někdo zkušenější vysvětloval?
  - Podařilo se Vám během prvních 30 minut používání nalézt, bez nutnosti asistence další osoby, v aplikaci všechny funkce, které jste potřebovali?
  - Pokud ne – které?
  - Co byste na aplikaci (z pohledu uživatele) změnili, upravili?

- Technická implementace
  - Došlo během Vašeho používání aplikace k nějaké chybě?
  - Pokud ano – k jaké, za jakých okolností, apod.
  - Funguje některá část aplikace viditelně pomalu?
  - Pokud ano – která, za jakých okolností?

- Obecné otázky:
  - Do jaké míry hodnotíte snadnost používání vytvořené aplikace?
  - Splňuje aplikace Vaše očekávání?
  - Máte k aplikaci další připomínky či návrhy k vylepšení?