Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Extrakce dat o aktuálních pozicích vozidel MHD v Plzni a jejich využití v simulaci silniční dopravy

Plzeň, 2014 Petr Šiml

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 14. 5. 2014 .....

Petr Šiml

Poděkování

Rád bych poděkoval Ing. Tomáši Potužákovi, Ph.D. za trpělivost, rady a vedení v celém průběhu této práce.

Abstract

Extraction of Data Describing Current Positions of Public Transport Vehicles in Plzeň and its Utilization in Road Traffic Simulation

This thesis deals with the general simulation and its subset – the simulation of road traffic. It also includes the description of the PMDP's web. PMDP is a company that provides public transport in Plzeň. On the web there are information about schedules and about delay of the vehicles on the road.

The main task of the practical part of the thesis is to analyze and implement the application which collects all needed data from PMDP's web and determines position of the vehicles on the road. The application should also have some mechanism to provide the determined positions for the systems of the road traffic simulation.

Abstrakt

Extrakce dat o aktuálních pozicích vozidel MHD v Plzni a jejich využití v simulaci silniční dopravy

Tato práce se zabývá obecně pojmem simulace a také její podmnožinou, simulací silniční dopravy. Také obsahuje popis webu PMDP – firmy, která v Plzni zajišťuje městskou hromadnou dopravu. Na webu PMDP jsou obsaženy informace o jízdních řádech a také o zpoždění jednotlivých vozidel na trati.

Hlavní úlohou praktické části této práce je analýza a implementování aplikace, která sbírá data z webu PMDP a určuje souřadnice vozidel městské hromadné dopravy na trati. Aplikace má také obsahovat možnost poskytnutí určených souřadnic systémům určeným k simulaci silniční dopravy.

OBSAH

1 ÚVOD

V současné době, kdy se denně setkáváme s narůstající dopravou všude kolem nás, je nutné se pokusit tuto dopravu v některých místech usměrnit, či regulovat. Důvodem k těmto opatřením je zvyšující se počet dopravních kolapsů a nutnost zlepšit ekonomické, ale i ekologické podmínky pro život.

Řešení tohoto problému není právě jednoduché, protože nelze předem určit, kolik dopravních vozidel bude v určitý moment na určitém místě. Dochází tak alespoň k odhadům, které by měly napovědět, jaká opatření by pomohla na některých místech a jaká by situaci možná ještě zhoršila. K těmto odhadům se dají využít mimo jiné dopravní simulace, které při správné funkčnosti mohou pomoci předvídat hustotu dopravy za určité situace. K tomu je zapotřebí velké množství vstupních dat z různých zdrojů.

Hlavním cílem této práce je vytvořit aplikaci, která by sbírala informace o pozicích jednotlivých vozů MHD na území města Plzeň a v jeho okolí. Tyto pozice lze pak dále zpracovat a využít v systémech pro simulování dopravy, např. pro simulaci vozidel MHD. Přesné informace o pozicích vozidel nejsou ale bohužel veřejně k dispozici. Tato práce se zabývá získáním zpoždění vozidel z webu Plzeňských městských dopravních podniků (PMDP) [1] a dále počítáním jejich GPS souřadnic ze známé trasy dané linky. Vypočítané souřadnice by pak aplikace měla být schopna poskytnout v dostupné formě právě pro simulační systémy.

V textu práce bude nejprve podrobněji vysvětlen obecný pojem simulace, poté její odvětví simulace silniční dopravy a následně popsán web PMDP. Dále bude provedena důkladná analýza problému, která napoví nejlepší použitelnou architekturu a technologie pro vývoj aplikace. Samotný vývoj aplikace a některé z použitých algoritmů pak budou popsány v části zaměřené na implementaci.

2 SIMULACE

Jak již bylo naznačeno v úvodu, systémy pro simulaci dopravy obsahují oproti reálnému prostředí větší či menší zjednodušení a jejich výsledky tak nejsou zcela přesné. Podobně jako není zcela přesná předpověď počasí. Děje se tak z důvodu (i v dnešní době) nedostatečného výpočetního výkonu počítačů, což se často řeší využitím distribuovaného či paralelního výpočetního prostředí. Dalším problémem je absence některých potřebných dat pro zahrnutí veškerých parametrů [2].

Simulace lze obecně rozdělit do několika skupin. Za hlavní parametry dělení se považuje účel simulace a způsob toku simulačního času [3].

2.1 Rozdělení podle účelu simulace

Dle účelu můžeme dělit simulace do hlavních dvou skupin a to na *analytické simulace* a *virtuální prostředí*.

2.1.1 Analytické simulace

Slovo *analýza* asi nemusí být více představováno, v obecném pojetí se jedná o rozbor něčeho konkrétního a vyvození důsledků. Analytické simulace zkoumají reálné systémy nebo systémy, které jsou v danou chvíli před fází realizace, a počítá se s jejich výstavbou. Hlavním cílem těchto simulací bývá pro reálné systémy určit chování při nějaké změně, např. chování dopravní sítě při nehodě nebo stavebních pracích.

Pro systémy ve fázi návrhu pak tyto simulace nacházejí uplatnění v rozhodnutí, zda budovaný systém má smysl dále vyvíjet nebo ne. V tu chvíli dovedou simulační programy zabránit zbytečnému utrácení peněz a vrátit neefektivní projekt na začátek [3].

Jelikož účelem této práce je získávat informace z reálného dopravního systému, měly by výstupní data sloužit právě pro analytické simulace.

2.1.2 Virtuální prostředí

Dalším typem simulací při dělení podle jejich účelu jsou virtuální prostředí. V těchto simulacích je typicky předpokládána neustálá přítomnost lidských uživatelů, kdy systém v závislosti na jejich rozhodnutí vyhodnocuje situaci. Narozdíl od analytických situací, které pracují spíše v diskrétním čase (viz dělení simulací podle plynutí času), se

virtuální prostředí soustředí na real-time zpracování a jedním z klíčových problémů může být doba odezvy [3].

Běžnou úlohou, kterou můžeme do této skupiny zařadit, jsou trenažery pro jízdu autem či jiným dopravním prostředkem.

2.2 Rozdělení podle toku času

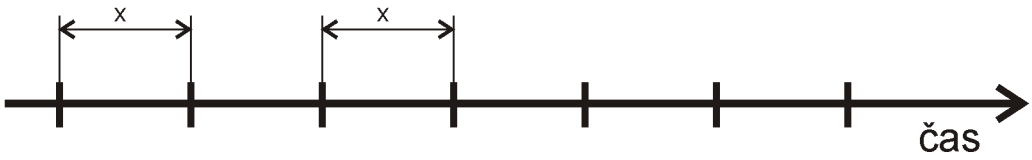
V závislosti na tom, jak plyne simulační čas, můžeme simulace zařadit do skupiny *spojitých simulací* nebo *diskrétních simulací* [3]. Pro vysvětlení rozdílů mezi těmito skupinami budeme sledovat aktuální stav simulace. Ten je dán hodnotami různých parametrů a funkcí v jednu konkrétní chvíli.

Pro spojité simulace se tento aktuální stav mění spojitě. Lze tedy vypočítat nebo jinak určit tento stav pro jakýkoli vybraný moment.

Naopak v diskrétních simulacích se tento stav mění nárazově a to buď v pravidelných intervalech (pak se jedná o tzv. *time-stepped simulace*) nebo ve chvílích, kdy nastala nějaká událost (tzv. *event-driven simulace*) [3].

2.2.1 Time-stepped simulace

V time-stepped (neboli krokovaných) simulacích je celkový čas simulace rozdělen do stejně dlouhých časových intervalů (v závislosti na typu úlohy mohou být velikosti intervalů zvoleny v řádech milisekund, ale i hodin), viz Obrázek 1. Po každém intervalu se vždy vypočítají všechny parametry a hodnoty funkcí, které určují stav simulace, a zpracují se události vyskytlé v jednom kroku [3].

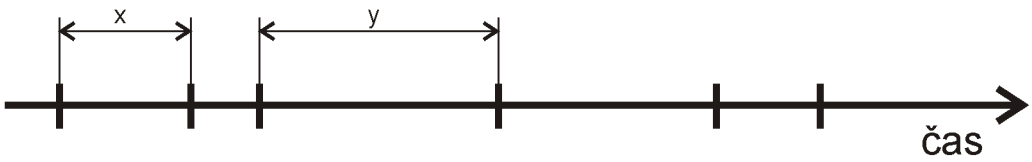


Obrázek 1 – Časová osa time-stepped simulace

2.2.2 Event-driven simulace

Ve skupině event-driven simulací, neboli simulací řízených událostmi, nejsou časové intervaly pro provádění výpočtů pravidelné (viz Obrázek 2). Simulace obsahuje několik událostí (např. výskyt nového prvku nebo jeho přesun na jiné místo), které se plánují na určitý čas a v tomto čase se vykonají. Důležitou součástí simulace je proto tzv. kalendář

obsahující jednotlivé události a časy, kdy se má která událost vykonat. Události jsou řazeny podle času a vždy se vykoná nejprve ta s nejnižším simulačním časem. V jejím průběhu může událost sama vytvořit novou nebo některou z již naplánovaných událostí přesunout na jiný čas [3].



Obrázek 2 – Časová osa event-driven simulace

2.3 Distribuované simulace

Aby se urychlil výpočetní čas simulací, dochází v dnešní době k rozdělení výpočetního výkonu mezi více (jader) procesorů na jednom zařízení, případně k rozdělení na více strojů (obecně uzlů). Obě možnosti mají své výhody i nevýhody [3].

Rozdělení na více uzlů se musí vypořádat s delšími časy vzájemné komunikace. Lze ale provádět výpočet i na stovkách strojů zároveň a výrazně tak zvýšit celkový výkon. Naopak využití vícejádrových procesorů nebo více procesorů v jednom zařízení využívá výhody krátkých komunikačních časů. Také se zde může využít sdílené paměti při menším množství procesorů. V tu chvíli se ale samozřejmě musí zabránit různým speciálním chybám pomocí synchronizace [4].

Ať už se použije jakákoli z těchto dvou metod, musí se s rozdělením na více výpočetních prvků počítat už při návrhu simulace. Otázkou zůstává, jak práci rozdělit, aby nedocházelo k chybám kvůli špatné synchronizaci a zároveň aby nebyla komunikace mezi dvěma procesy delší, než kdyby simulace běžela jen na jednom z nich [4]. Těmito problémy se zabývají následující kapitoly o dekompozici problému a synchronizaci.

2.3.1 Dekompozice

Dekompozice v tomto případě udává, jak by měla být simulace rozdělena mezi jednotlivé procesy. Existuje více možností, jak dekompozici provést, mezi základní způsoby patří:

- *Dekompozice výpočtu* – simulace je rozdělena na moduly vykonávající rozdílné části simulačního výpočtu [5].

- *Prostorová dekompozice* – na všech výpočetních prvcích běží stejný kód, který se ale vykonává pro menší množství dat (data na všech prvcích dohromady dávají kompletní data simulace) [5].
- *Časová dekompozice* – na všech výpočetních prvcích běží stejný kód nad všemi daty, ale pro menší časový interval. Každý výpočetní prvek pak počítá jiný časový interval. Problémem je, že pro výpočet intervalu x je třeba odhadnout výsledek intervalu $x - 1$. Po vykonání všech intervalů se tak zpětně zkoumá, zda byly odhady vstupních dat správné. Pokud se ukáže, že nebyly, musí se následující časový interval spočítat znovu [3].

2.3.2 Synchronizace

Všechny distribuované simulace, kdy se jednotlivé procesy podílejí na společném výsledku, by měly obsahovat určité synchronizační mechanismy, aby předešly chybám vzniklým z práce s neaktuálními nebo nesprávnými daty. V žádném případě totiž nelze spoléhat na rychlost procesů a musí se zaručit stejný výsledek, jako by simulace běžela sériově v rámci jednoho procesu [4].

Problém synchronizace procesů se částečně liší v závislosti na tom, jak se simulace staví k toku času. Pro oba typy simulací zmíněné v kapitole 2.2 je jednou z běžných možností využití synchronizačního primitiva *bariéra*. Ta způsobuje, že procesy na bariéře čekají na ostatní, dokud se všechny k bariéře nedostanou (v tu chvíli si procesy mohou data vyměnit, protože data jsou aktuální) [4].

U event-driven simulací je nutné nejprve určit, které události se mohou provést bez nutnosti synchronizace [3]. Po jejich provedení se využije právě zmíněné bariéry, aby si procesy vyměnily informace.

U time-stepped simulací, kde se využívá bariéra, vždy procesy spočítají vše v rámci jednoho kroku. Poté se synchronizují jejich lokální časy a výpočet pokračuje. Nevýhoda spočívá ve velkém množství a frekvenci zasílaných dat. Situaci lze vylepšit pomocí lokálních hodin, kdy k synchronizaci samotných procesů dochází jen jednou za více časových intervalů [4].

V distribuovaném prostředí je pro implementaci synchronizace nutno využít zasílání zpráv [3]. V paralelním prostředí je možné s výhodou využít sdílenou paměť.

3 SIMULACE SILNIČNÍ DOPRAVY

Simulace silniční dopravy je stále se rozšiřující odvětví obecné simulace. S postupem času se i toto odvětví začalo dělit do určitých skupin podle několika aspektů. Jeden ze zásadních lze nalézt v rozdílech reprezentace vozidel.

3.1 Reprezentace vozidel

3.1.1 Makroskopické simulace

V rámci makroskopických simulací nejsou vozidla jako konkrétní objekty vůbec uvažována. Doprava na jednotlivých silnicích je reprezentována toky vozidel popsaných za pomoci určitých parametrů. Parametry mohou být různé, např. počet průjezdů vozidel za jednotku času a průměrná rychlost. Tyto simulace mohou být bez větších obtíží spuštěny i na jednoprocesorových počítačích pro oblast jednoho středně velkého města [6].

3.1.2 Mikroskopické simulace

Mikroskopické simulace obsahují jednotlivá vozidla jako samostatné objekty, jež se pohybují po svých vlastních trasách, mají různou rychlost a další parametry. Díky tomu mnohem více odpovídají realitě, ale samozřejmě jsou také mnohem náročnější na výpočetní výkon. Pro tyto simulace je často využíváno paralelní či distribuované výpočetní prostředí.

3.1.3 Mezoskopické simulace

Mezoskopické simulace jsou poslední skupinou v rámci dělení podle reprezentace vozidel. Sem patří všechny simulace využívající některé prvky zároveň z makroskopických i mikroskopických simulací. Uvažují se zde jednotlivá vozidla, nicméně stále se některé parametry týkají jejich skupin [7].

3.2 Tok času

Pro simulaci silniční dopravy je vhodné používat diskrétní reprezentaci času. Kvůli většímu množství událostí (díky mnoha vozidlům, které se rozjíždějí či brzdí) jsou pak častějším řešením time-stepped simulace probrané v kapitole 2.2.1. Po každém kroku se tak mohou vypočítat pozice vozidel, jejich směr a ostatní důležité parametry. Jedno

vozidlo pak může reagovat na pohyb vozidla, které jede před ním, za ním, či v protějším směru předjíždí atd.

3.3 Základní dopravní modely

Jak vyplývá z předchozího textu, nejběžněji používanými typy silničních simulací jsou mikroskopické simulace s využitím krokovaného času s pevně daným intervalem. Pro tyto simulace existují dva běžně používané modely popsané v následujících kapitolách.

3.3.1 Cellural automaton model

Prvním modelem, který zmíníme, je model využívající buňkový automat (cellural automaton model) [8]. V tomto případě jsou silnice rozděleny na stejně velké buňky. Každá buňka může obsahovat pouze jedno vozidlo nebo být prázdná a zároveň vozidlo je vždy v jedné buňce. Každé z vozidel má pak určitou rychlost, kterou se vždy v prováděném kroku simulace pohybuje dopředu po své trase. V závislosti na rychlosti vozidla se po provedeném kroku přesune o určitý počet buněk na další pozici [8].

Rychlost vozidla se může v průběhu simulace měnit. Každé z vozidel totiž neustále zrychluje na svou maximální rychlost (danou limitem vozu nebo rychlostním omezením silnice), pokud před sebou nemá překážku v podobě jiného vozidla. V tu chvíli naopak rychlost snižuje, aby nedošlo ke srážce [8].

Aby se simulace přiblížila reálné situaci, každému z vozidel se s určitou pravděpodobností náhodně změní rychlost, čímž se simuluje chování řidiče, ovlivnění dopravy počasím atd.

3.3.2 Car-following model

Další možností pro výběr dopravního modelu jsou car-following modely [9]. Oproti modelu využívajícímu buňkový automat zde nejsou žádné buňky, do kterých se musí vozidlo vejít. Místo toho může být na libovolném místě jízdního pruhu. Společnou vlastností obou modelů je reakce vozidla na podněty způsobené ostatními objekty na silnici. Vozidla tak zrychlují nebo zpomalují v závislosti na prvním vozidle před sebou. Také zde nejsou přítomny nedeterministické změny v rychlosti.

3.4 Distribuované simulace silniční dopravy

V této části využijeme poznatky o distribuovaných simulacích a simulacích silniční dopravy.

3.4.1 Dekompozice

Pro simulaci silniční dopravy nejsou zcela vhodné všechny možnosti dekompozice. V následujících odstavcích jsou naznačeny zejména problémy, které se při daném typu dekompozice mohou objevit.

Princip dekompozice výpočtu, kdy se jednotlivé části simulačního programu vykonávají zvlášť, nelze u simulací silniční dopravy snadno použít. Důvodem je velké množství výpočtů, které jsou potřeba pro výpočet souřadnic vozidel a jejich dalších parametrů. V tu chvíli by jeden z procesorů byl vytížen na maximum, přičemž ostatní by místo paralelního běhu větší část času čekaly [5].

Řešením by bylo rozdělit výpočet na podúkoly a ty přiřadit jednotlivým procesům. V tu chvíli ale vyvstane otázka, jak toto rozdělení správně provést. Pohyby vozidel jsou totiž dány celou řadou poměrně jednoduchých pravidel a pro kompletní pohyb vozidla jsou zapotřebí všechny najednou. Po rozdělení na podúkoly by pak muselo docházet k časté meziprocesové komunikaci, což by výrazně zpomalilo vykonávání kódu.

Naopak nejvíce používanou dekompozicí pro simulace silniční dopravy je prostorová dekompozice. Ta umožňuje rozložit dopravní síť do menších celků a každý proces pak podle stejného kódu zpracovává svůj díl práce. Vozidla se samozřejmě mohou pohybovat i mezi jednotlivými úseky sítě, ale v jeden okamžik se o konkrétní vozidlo stará vždy právě jeden proces. Při přechodu do jiné části musí být informace o vozidle (jeho směr, aktuální rychlost atd.) předány mezi procesy [4].

Při použití této metody je nutno řešit, jakým způsobem síť rozdělit, aby každý z procesů měl přibližně stejně velký díl práce (tzn. aby jejich výpočetní časy byly přibližně stejné), a zároveň, aby se co nejvíce omezila meziprocesová komunikace. Snížení komunikace dosáhneme, pokud jednotlivé části sítě rozdělené mezi procesy budou propojeny co nejmenším množstvím silnic, případně méně frekventovanými silnicemi. Díky tomu nebude nutné mezi procesy posílat větší množství dat a výrazně se tak urychlí celkový výpočetní čas [4].

V případě časové dekompozice se může výrazným způsobem snížit výpočetní čas. Mohou ale nastat známé problémy popsané v kapitole 2.3.1, kdy skutečný výstup z jednoho intervalu nemusí odpovídat odhadům z intervalu následujícího. Aby se zabránilo podobným chybám, které by naopak zpomalily výpočet, může se určit úroveň tolerance – tedy výstupní data z jednoho intervalu se nemusejí přesně shodovat se

vstupními daty dalšího [10]. I tak jim ale musejí být dostatečně podobné. Přesto se časová dekompozice výrazněji v distribuovaných dopravních simulacích nepoužívá.

3.5 Synchronizace

V kapitole 2.3.2 bylo vysvětleno, proč je důležitá synchronizace obecně v distribuovaných simulacích. I při simulacích silniční dopravy může špatná (nebo žádná) synchronizace nepříznivě ovlivnit výsledky. Např. při time-stepped simulacích může způsobit, že vozidlo přijede na určité místo dříve nebo později, než mělo. S dřívějšími příjezdy se lze poměrně snadno vypořádat pomocí časových údajů, které s sebou vozidlo může přenášet. Díky nim může být vozidlo dočasně umístěno do bufferu a vloženo zpět na plánované místo v požadovaný čas.

Pozdější příjezdy ale mohou ovlivnit dlouhodobě simulaci a nelze jim snadno zabránit. Musejí se tedy použít synchronizační primitiva, typicky bariéra zmíněná v kapitole 2.3.2.

3.6 Shrnutí

V rámci této práce budou sbírány data o pozicích jednotlivých vozidel MHD v Plzni. Tato data lze v budoucnu využít zejména v simulaci s následujícími vlastnostmi:

- Time-stepped simulace – sběr dat lze provádět periodicky, mohou být tedy využitelná pro simulaci s pravidelnými časovými intervaly.
- Analytická simulace – data budou vycházet přímo z reálného prostředí.
- Mikroskopická simulace – data budou charakterizovat vždy jedno konkrétní vozidlo.

4 PLZEŇSKÉ MĚSTSKÉ DOPRAVNÍ PODNIKY

4.1 O společnosti

Plzeňské městské dopravní podniky, a.s. jsou společností provozující veškerou městskou hromadnou dopravu v Plzni a jejím nejbližším okolí. PMDP sídlí v centru města, konkrétně na Denisovo nábřeží. Mimo MHD se společnost zabývá i provozem autoškoly, mytím vozů nebo jejich lakováním [1].

4.2 Městská hromadná doprava v Plzni

4.2.1 Typy vozidel a značení linek

V rámci městské hromadné dopravy v Plzni se cestující mohou běžně setkat se třemi typy vozidel (mimo historická) [1]. Jsou zde:

- trolejbusy,
- tramvaje,
- autobusy.

V počtu linek přitom jednoznačně převažují autobusy nad trolejbusy. Tramvajové linky jsou pouze tři.

Za specifické lze označit noční autobusy. Ty lze rozeznat podle označení linky *NX* (kde *X* je kladné číslo), zatímco ostatní linky jsou značeny pouze čísly. I tak lze ale rozpoznat typ vozidla pro danou linku podle jejího označení, protože tramvajové linky mají číselné značení $X \leq 4$, trolejbusové linky $10 \leq X < 20$ a autobusové linky $X \geq 20$.

Značení linek obsahuje také speciální znaky, které dávají cestujícím na vědomí, že dané vozidlo se v průběhu jízdy odkloní od běžného směru. Např. vozidla linky 13 jezdí typicky v jednom směru ze zastávky *Na Dlouhých* do zastávky *NC Černice*. V případě, že je v tomto směru nějaký odjezd vozidla označen písmenem C (např. 7:32C pro odjezd v 7:32), jede toto vozidlo ze zastávky *Na Dlouhých* pouze do zastávky *Černice*.

4.2.2 Časy odjezdů vozidel

Pro všechny linky MHD na území Plzně jsou časy odjezdů rozděleny do tří skupin [1]:

- pracovní dny,
- soboty,

- neděle a svátky.

Dle rozdělení do těchto skupin pak mají vozidla např. ve všechny všední dny stejné odjezdové časy z konkrétních zastávek.

4.2.3 Poloha zastávek

Pro zastávky je samozřejmě typické, že jsou na obou stranách silnice a v závislosti na směru jízdy vozidla cestující vystupují buď na jedné nebo na druhé straně vozovky. Tyto protilehlé zastávky mívají stejná pojmenování, aby nedocházelo k matení cestujících. V některých případech ale může nastat situace, že dvě zastávky se stejným jménem nejsou jen na protilehlých stranách silnice, ale jsou od sebe výrazně vzdáleny. Viz Obrázek 3, kde jsou dvě zastávky s názvem *Náměstí Republiky* zobrazené na mapě.

Obrázek 3 – Zastávky Náměstí Republiky

Některé ze zastávek se stejným pojmenováním jsou využívané různými typy vozidel. Také mohou být stejným jménem označené jak průjezdní, tak konečné zastávky. Proto např. pro zastávku Bory existuje šest různých míst, kde staví tramvaje a autobusy.

4.2.4 Změny jízdních řádů

Časy odjezdů vozidel ze zastávek se každoročně upravují podle využití směrů a linek [1]. I v průběhu roku lze ovšem nalézt (spíše krátkodobé) změny jízdních řádů, většinou z důvodu uzavření některých silnic díky stavebním pracím. V tu chvíli se na webu objevují nové linky, názvy zastávek a s tím související změny odjezdových časů.

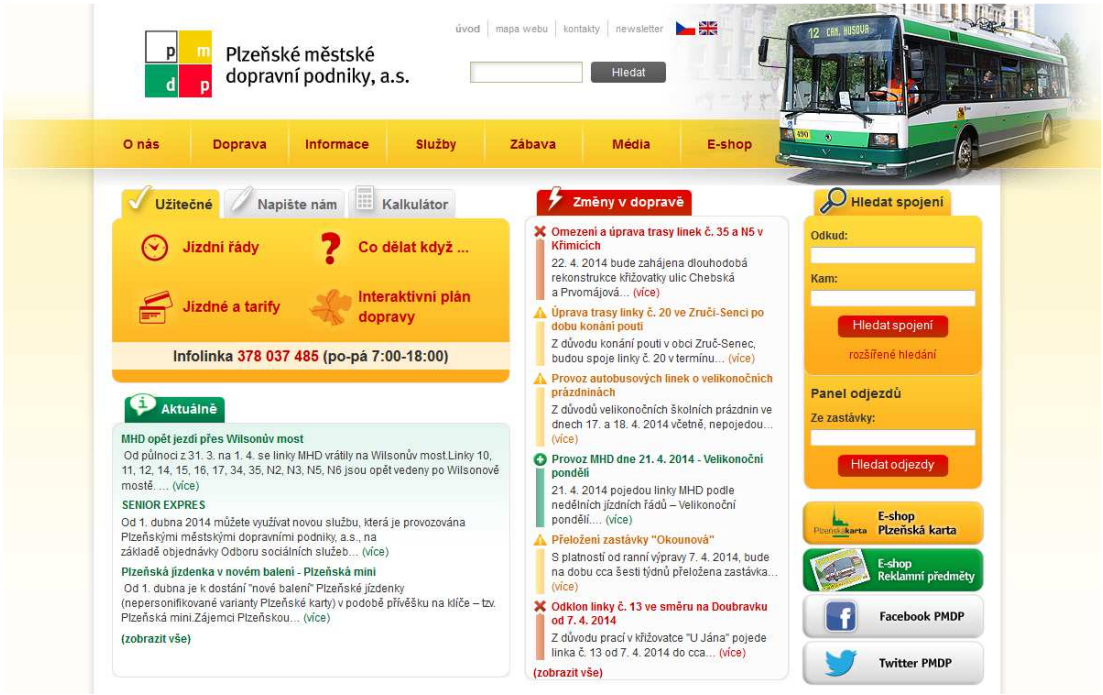
4.2.5 Statistické údaje

V rámci plzeňské MHD je evidováno k 20. 4. 2014:

- 45 linek,
- 368 názvů zastávek,
- 141 246 časů odjezdů.

4.3 Webové stránky PMDP

Webové stránky Plzeňských městských dopravních podniků (viz Obrázek 4) se nacházejí na adrese http://www.pmdp.cz [1]. Obsahují nepřeberné množství informací a multimedií. Uživatelům je tu k dispozici vše od prodeje jízdenek po fotografie jednotlivých typů dopravních vozidel v Plzni. Většina těchto dat a informací není pro tuto práci relevantních, web ovšem nabízí uživateli i možnost prohlížet jízdní řády všech linek MHD a další funkce s tím spojené. Zejména pak informace o zpoždění vozidel na trati.



Obrázek 4 – Web PMDP

Pro jízdní řády je na webu přímo zařízena subdoména http://www.jizdnirady.pmdp.cz, která nabízí uživateli vše přehledně na jednom místě. Zde jsou umístěny sekce *Vyhledat spojení*, *Panel odjezdů*, *Zastávkové JŘ*, *Denní JŘ*, *Zastávky* a *JŘ v mobilu*.

Tyto sekce obsahují např. formuláře pro vyplnění míst, odkud nebo kam se chce návštěvník stránek dostat pomocí MHD. Všechny tyto formuláře jsou vybaveny tzv. našeptávačem, který napovídá uživatelům po zadání části textu plné názvy volených míst.

Pro každou ze záložek je pak na webu obsažena i nápověda obsahující všechny možnosti a způsoby správného používání funkcí.

4.3.1 Záložka Vyhledat spojení

Záložka *Vyhledat spojení* (viz Obrázek 5) umožňuje, jak napovídá název, vyhledávat konkrétní spoje mezi dvěma zastávkami včetně možných přestupů. Uživatel může zadat nejen počáteční a konečnou zastávku, ale i zastávku, kde chce přestupovat na jinou linku, nebo naopak zvolit pouze přímé spoje mezi zastávkami.

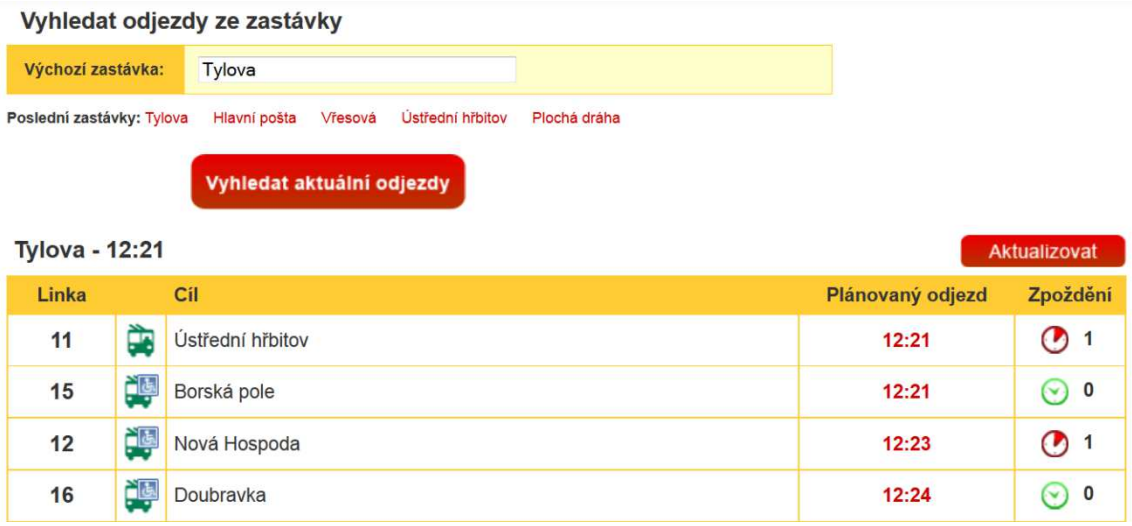
Obrázek 5 – Záložka Vyhledat spojení

Dalšími údaji pro vyhledání spoje je samozřejmě datum a čas, kdy systém automaticky rozpozná, zda se jedná o pracovní den nebo např. svátek, a vybere požadované časy odjezdů nebo příjezdů – podle volby uživatele.

Nepřekvapí ani možnost zvolit maximální počet přestupů nebo nejdelší čekací dobu mezi dvěma přestupy. Dále může uživatel vybrat typy vozidel, kterými se chce dopravovat. K dispozici jsou všechny typy vozidel zmíněné v kapitole 4.2.1.

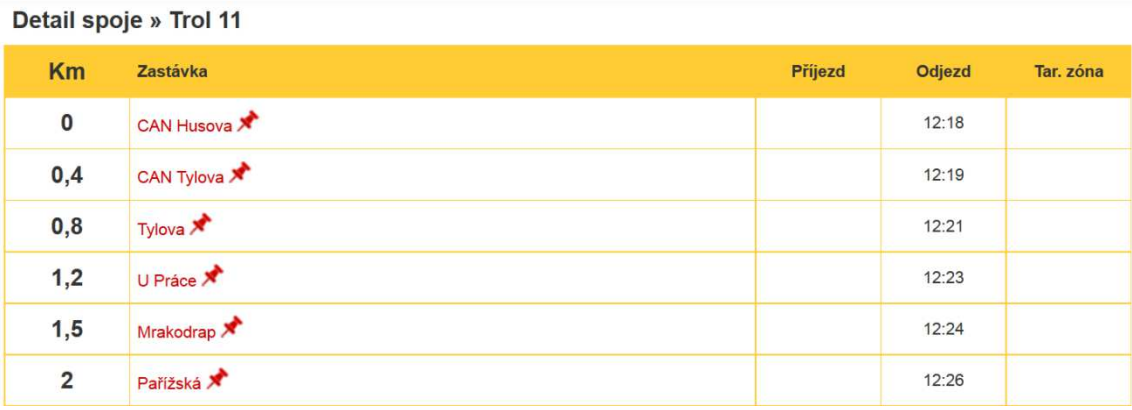
4.3.2 Záložka Panel odjezdů

V sekci *Panel odjezdů* (viz Obrázek 6) může uživatel vyhledávat nejbližší odjezdy z jedné konkrétní zastávky. Výsledky se zobrazí seřazeně podle času odjezdu vozidel a to pro všechny linky procházející zastávkou v jakémkoli směru. Největší výhodou této části webu je zobrazování informací o zpoždění vozidel. To se aktualizuje pravidelně každou minutu a uživatel může na první pohled (podle barevného odlišení) rozpoznat, které vozy budou v zastávce včas a které ne.



Obrázek 6 – Panel odjezdů

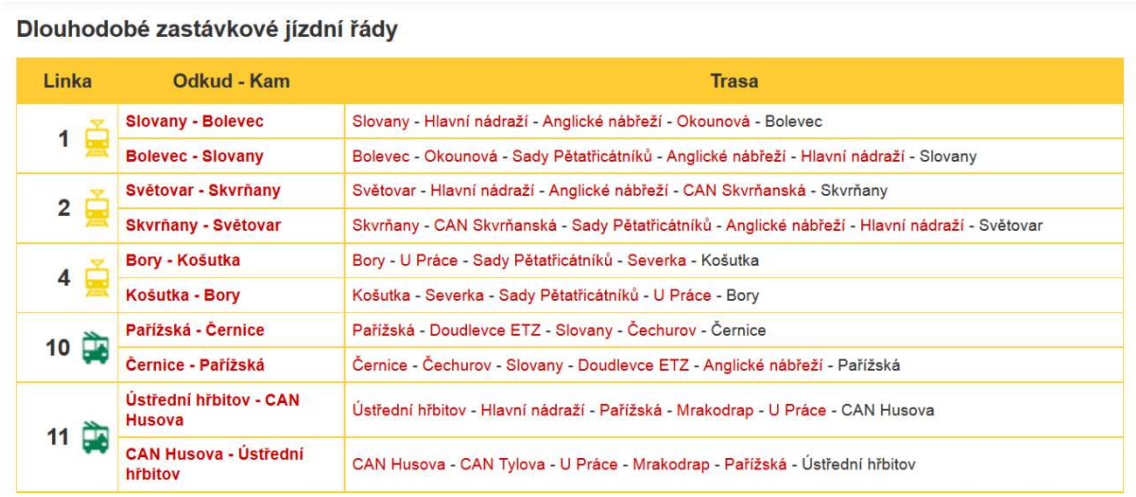
Při zvolení času ze sloupce Plánovaný odjezd se uživateli detail spoje vybraného vozidla s odjezdovými časy ze všech zastávek (viz Obrázek 7).



Obrázek 7 – Detail spoje

4.3.3 Záložka Zastávkové JŘ

V záložce *Zastávkové JŘ* (viz Obrázek 8) lze nalézt přehledný výpis všech linek a jejich hlavních směrů. Linky jsou seřazeny podle svého značení a pro každou z nich jsou tu uvedeny jen základní směry bez speciálních označení zmíněných v kapitole 4.2.1.

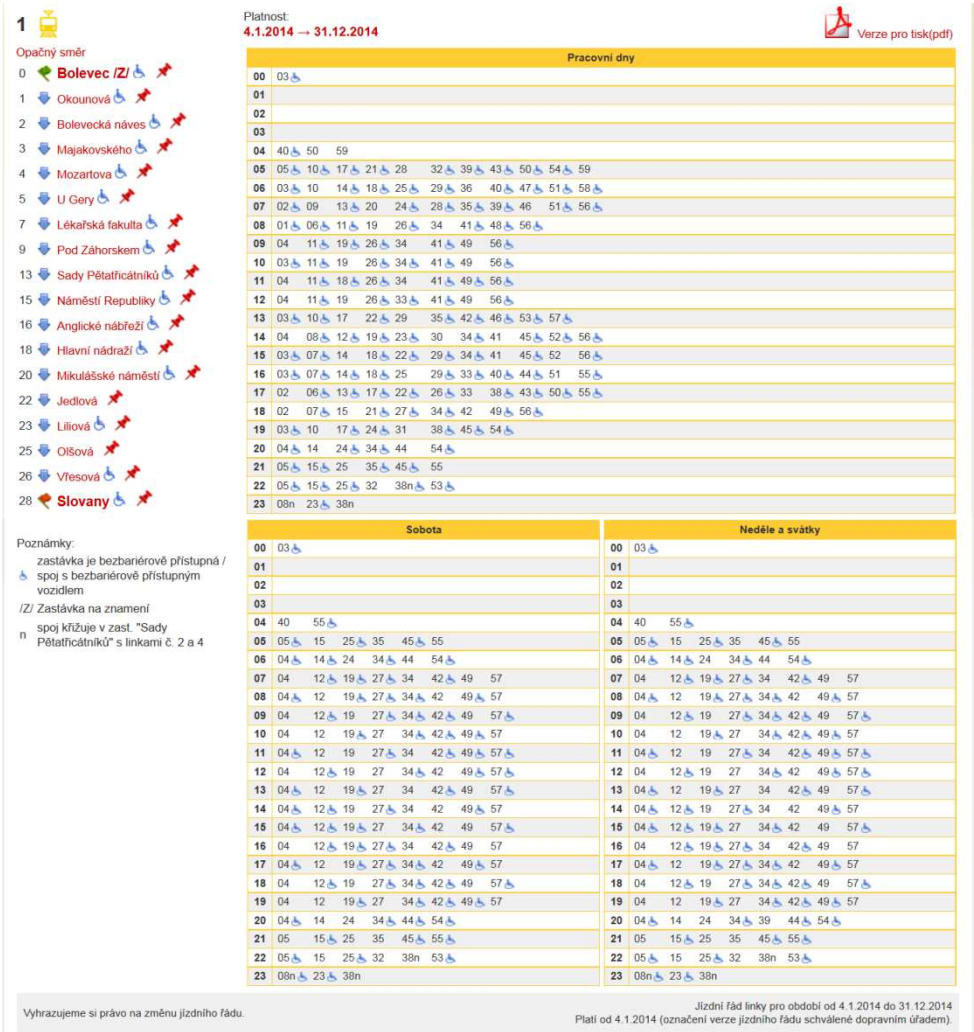


Dlouhodobé zastávkové jízdní řády

Linka	Odkud - Kam	Trasa
1	Slovany - Bolevec	Slovany - Hlavní nádraží - Anglické nábřeží - Okounová - Bolevec
	Bolevec - Slovany	Bolevec - Okounová - Sady Pětatřicátníků - Anglické nábřeží - Hlavní nádraží - Slovany
2	Světovar - Skvrňany	Světovar - Hlavní nádraží - Anglické nábřeží - CAN Skvrňanská - Skvrňany
	Skvrňany - Světovar	Skvrňany - CAN Skvrňanská - Sady Pětatřicátníků - Anglické nábřeží - Hlavní nádraží - Světovar
4	Bory - Košutka	Bory - U Práce - Sady Pětatřicátníků - Severka - Košutka
	Košutka - Bory	Košutka - Severka - Sady Pětatřicátníků - U Práce - Bory
10	Pařížská - Černice	Pařížská - Doudlevce ETZ - Slovany - Čechurov - Černice
	Černice - Pařížská	Černice - Čechurov - Slovany - Doudlevce ETZ - Anglické nábřeží - Pařížská
11	Ústřední hřbitov - CAN Husova	Ústřední hřbitov - Hlavní nádraží - Pařížská - Mrakodrap - U Práce - CAN Husova
	CAN Husova - Ústřední hřbitov	CAN Husova - CAN Tylova - U Práce - Mrakodrap - Pařížská - Ústřední hřbitov

Obrázek 8 – Zastávkové JŘ

Z této sekce se lze zvolením požadovaného směru dostat na stránku s kompletními časy odjezdů ze zastávek daného směru (viz Obrázek 9).



1

Opačný směr

0 Bolevec /Z/
1 Okounová
2 Bolevecká náves
3 Majakovského
4 Mozartova
5 U Gery
7 Lékařská fakulta
9 Pod Záhorskem
13 Sady Pětatřicátníků
15 Náměstí Republiky
16 Anglické nábřeží
18 Hlavní nádraží
20 Mikulášské náměstí
22 Jedlová
23 Liliová
25 Olšová
26 Vřesová
28 Slovany

Poznámky:
zastávka je bezbariérově přístupná / spoj s bezbariérově přístupným vozidlem
/Z/ Zastávka na znamení
n spoj křižuje v zast. "Sady Pětatřicátníků" s linkami č. 2 a 4

Platnost:
4.1.2014 → 31.12.2014

Verze pro tisk(pdf)

Pracovní dny	
00	03
01	
02	
03	
04	40 50 59
05	05 10 17 21 28 32 39 43 50 54 59
06	03 10 14 18 25 29 36 40 47 51 58
07	02 09 13 20 24 28 35 39 46 51 56
08	01 06 11 19 26 34 41 48 56
09	04 11 19 26 34 41 49 56
10	03 11 19 26 34 41 49 56
11	04 11 18 26 34 41 49 56
12	04 11 19 26 33 41 49 56
13	03 10 17 22 29 35 42 46 53 57
14	04 08 12 19 23 30 34 41 45 52 56
15	03 07 14 18 22 29 34 41 45 52 56
16	03 07 14 18 25 29 33 40 44 51 55
17	02 06 13 17 22 26 33 38 43 50 55
18	02 07 15 21 27 34 42 49 56
19	03 10 17 24 31 38 45 54
20	04 14 24 34 44 54
21	05 15 25 35 45 55
22	05 15 25 32 38n 53
23	08n 23 38n

Sobota	
00	03
01	
02	
03	
04	40 55
05	05 15 25 35 45 55
06	04 14 24 34 44 54
07	04 12 19 27 34 42 49 57
08	04 12 19 27 34 42 49 57
09	04 12 19 27 34 42 49 57
10	04 12 19 27 34 42 49 57
11	04 12 19 27 34 42 49 57
12	04 12 19 27 34 42 49 57
13	04 12 19 27 34 42 49 57
14	04 12 19 27 34 42 49 57
15	04 12 19 27 34 42 49 57
16	04 12 19 27 34 42 49 57
17	04 12 19 27 34 42 49 57
18	04 12 19 27 34 42 49 57
19	04 12 19 27 34 42 49 57
20	04 14 24 34 44 54
21	05 15 25 35 45 55
22	05 15 25 32 38n 53
23	08n 23 38n

Neděle a svátky	
00	03
01	
02	
03	
04	40 55
05	05 15 25 35 45 55
06	04 14 24 34 44 54
07	04 12 19 27 34 42 49 57
08	04 12 19 27 34 42 49 57
09	04 12 19 27 34 42 49 57
10	04 12 19 27 34 42 49 57
11	04 12 19 27 34 42 49 57
12	04 12 19 27 34 42 49 57
13	04 12 19 27 34 42 49 57
14	04 12 19 27 34 42 49 57
15	04 12 19 27 34 42 49 57
16	04 12 19 27 34 42 49 57
17	04 12 19 27 34 42 49 57
18	04 12 19 27 34 42 49 57
19	04 12 19 27 34 42 49 57
20	04 14 24 34 39 44 54
21	05 15 25 35 45 55
22	05 15 25 32 38n 53
23	08n 23 38n

Vyhrazujeme si právo na změnu jízdního řádu.

Jízdní řád linky pro období od 4.1.2014 do 31.12.2014
Platí od 4.1.2014 (označení verze jízdního řádu schválené dopravním úřadem).

Obrázek 9 – Časy odjezdů vozidel

Zde jsou v levém sloupci uvedeny všechny zastávky zvoleného směru a při výběru konkrétní zastávky se zobrazí časy odjezdů vozidel. Časy jsou rozděleny do tabulek zvlášť pro jednotlivé typy dnů zmíněné v kapitole 4.2.2. Jsou zde uvedeny i vozidla zvolené linky jezdící v jiném než běžném směru. Jejich časy odjezdů ze zastávky jsou označeny příslušným písmenem, ke kterému je na stránce přidáno vysvětlení, např. pro již zmíněnou linku 13 a označení C: *C Končí v zastávce Černice.*

U výpisu zastávek je za jejich názvy zobrazena ikona připomínající špendlík. Kliknutí na ikonu se zobrazí Interaktivní plán dopravy města Plzně (více v kapitole 4.3.8) s umístěním zastávky na mapě.

4.3.4 Záložka Sdružené JŘ

Sdružené jízdní řády je část webu, na které lze zadávat více míst, odkud kam se chtějí cestující přesunout v rámci Plzně a jejího okolí. Zadat lze samozřejmě název zastávky, dále pak jména ulic nebo významné záchytné body v Plzni jako je např. Komorní divadlo. Už z popisu možností je zřejmé, že tato funkce je zaměřena spíše pro turisty v Plzni, kteří nevědí, kudy vedou konkrétní linky MHD.

4.3.5 Záložka Denní JŘ

Funkce pro vyhledávání odjezdů vozidel jedné linky z konkrétní zastávky pro určitý směr je obsažena v sekci Denní jízdní řády. Uživatelé zde mohou navolit, zda chtějí vyhledané časy odjezdů pro všechny dny v týdnu nebo jen pro určitý konkrétní den.

4.3.6 Záložka Zastávky

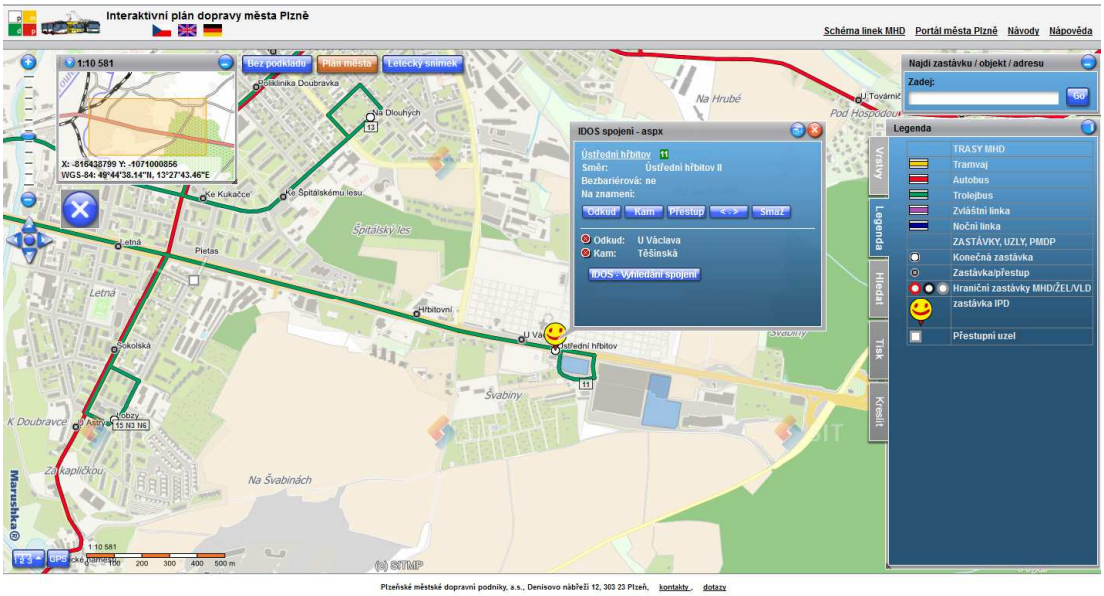
V této sekci lze nalézt seznam všech abecedně seřazených zastávek včetně linek, které jimi procházejí. Při zvolení jedné ze zastávek se uživateli zobrazí záložka *Panel odjezdů* s nejbližšími odjezdy vozidel. Při zvolení jedné z linek se pak uživatel dostane k časům odjezdů – viz Obrázek 9.

4.3.7 Záložka JŘ v mobilu

PMDP umožňují cestujícím stáhnout si aplikace pro jízdní řády do mobilních telefonů. Telefony s operačním systémem Android mohou využít aplikaci, která obsahuje funkce pro vyhledávání odjezdů včetně aktuálních informací o zpoždění. Další aplikací pro Android, ale i telefony s podporou mobilní Javy, je offline verze jízdních řádů. V sekci *JŘ v mobilu* jsou odkazy na tyto aplikace a jejich detailnější popis.

4.3.8 Interaktivní plán dopravy města Plzně

Velkou výhodou webu PMDP je propojení s tzv. *Interaktivním plánem dopravy města Plzně*. Jedná se o mapu obsahující všechny linky MHD v Plzni včetně jejich zastávek spolu s GPS souřadnicemi (viz Obrázek 10). Hlavní funkcí této aplikace je provázání právě s webem PMDP, kde mohou uživatelé nalézt informace o zastávkách a linkách, a známým vyhledávačem IDOS [11]. Ten umožňuje uživatelům vyhledávat ve známém prostředí všechny odjezdy vozidel MHD v Plzni.



Obrázek 10 – Interaktivní plán dopravy města Plzně

5 ANALÝZA ŘEŠENÍ

V této kapitole se budeme zabývat analýzou výsledného programu.

5.1 Specifikace požadavků

5.1.1 Účel práce

Výsledná aplikace má sloužit k určení co nejpřesnějších GPS souřadnic vozidel MHD v Plzni. Souřadnice budou dále poskytovány pro účely simulací silniční dopravy, proto je musí být možno získat ve snadno přenositelné a platformě nezávislé formě.

5.1.2 Získání GPS souřadnic vozidel MHD

GPS souřadnice vozidel nejsou veřejně dostupnými informacemi poskytovanými PMDP ani jinou organizací. Pro získání těchto údajů se využije zpoždění vozidel uvedené na webu PMDP. Požadavkem na výpočet je co možná největší přesnost.

Aby se minimalizovala nutnost interakce s uživatelem, jsou dalšími požadavky opakovatelnost měření, kdy aplikace bude moci běžet na pozadí systému a sbírat data. A dále výběr jednoho směru linky, pro který se budou data pravidelně získávat (tzn. uživatel nebude muset vždy vybrat konkrétní vozidlo, pro které by se měly souřadnice počítat).

5.1.3 Poskytnutí souřadnic ostatním aplikacím

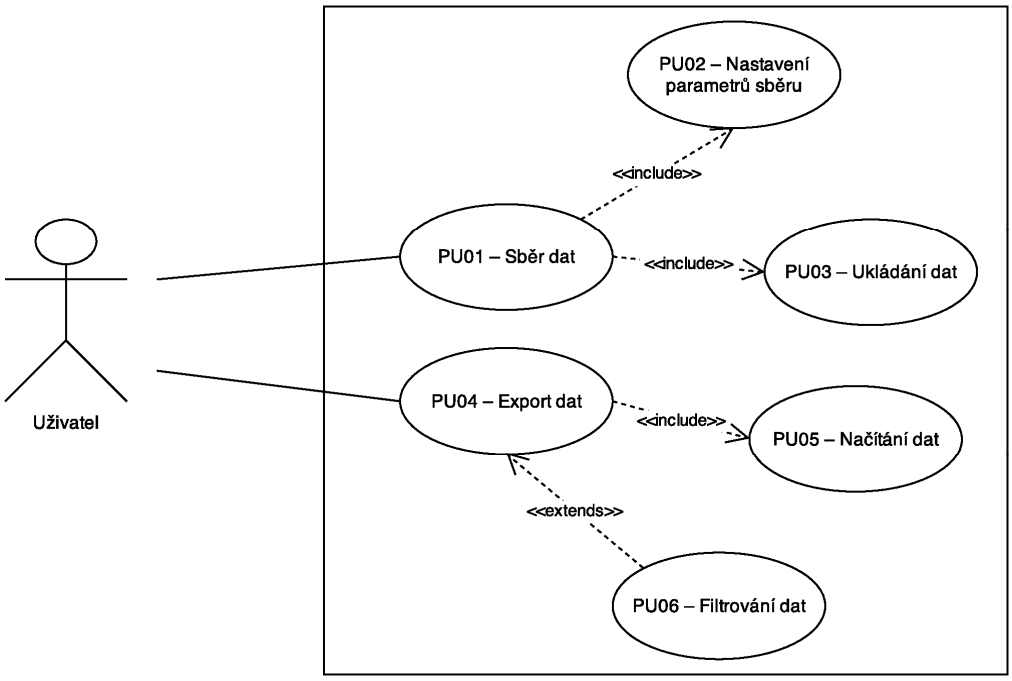
Aplikace musí mít možnost poskytovat data pro programy zabývající se simulací silniční dopravy. Způsob poskytování dat je ponechán na důkladnější analýze, ale doporučuje se využít přenositelný formát XML.

5.1.4 Požadavky na uživatelské rozhraní

Uživatelské rozhraní musí být přehledné a mělo by dodržovat tradiční přístup, obsahovat menu a toolbar pro nejčastěji používané akce. V případě problémů např. s připojením k webu PMDP musí být uživateli zřetelně dáno na vědomí, že se vyskytl problém.

5.2 Případy užití

Systém bude využíván v jednu chvíli právě jedním uživatelem. Případy užití z diagramu (viz Obrázek 11) jsou popsány v následujících kapitolách.



Obrázek 11 – Diagram případů užití

5.2.1 PU01 – Sběr dat

Uživatel spouští sběr dat, kde daty jsou myšleny zejména pozice vozidel MHD na trati. Ty se vypočítají ze zpoždění vozidel na webu PMDP.

5.2.2 PU02 – Nastavení parametrů sběru

Pozice vozidel se budou určovat jen pro určitý směr linky. Tento směr musí být před samotným sběrem vždy vybrán. Dalšími z parametrů bude počet opakování sběre dat a také časový interval mezi jednotlivými měřeními.

5.2.3 PU03 – Ukládání dat

Po sběru dat budou výsledky ukládány pro možnost pozdějšího exportu.

5.2.4 PU04 – Export dat

Uživatel prostřednictvím navrhovaného systému může získat nasbíraná data pro potřeby externích aplikací.

5.2.5 PU05 – Načítání dat

Při exportu se data musí načítat z jejich úložiště, kam byla nahrána po sběru dat.

5.2.6 PU06 – Filtrování dat

Export bude moci využít filtr pro zúžení výběru vrácených dat. Filtrem může být např. výběr konkrétního data a času.

5.3 Celková analýza problému

V této kapitole bude důkladněji rozebrán problém získávání souřadnic vozidel a následně možnosti jejich poskytnutí externím aplikacím.

5.3.1 Získání informací o zpoždění

Informace o zpoždění vozidel jsou dostupné pouze na webu PMDP v sekci *Panel odjezdů* popsané v kapitole 4.3.2. Na jedné HTML stránce jsou vždy zpoždění pro 15 vozů, které mají nejbližší čas odjezdu z vybrané zastávky. Mohou zde být tedy i časy odjezdů s dřívějším časem, než je současný v době hledání, pokud má vozidlo zpoždění.

Pro získání těchto údajů je nutné parsovat HTML a to buď pomocí vlastního parseru nebo s využitím hotové knihovny zvolené podle konkrétního programovacího jazyka. Ten bude vybrán v kapitole 5.4 věnující se výběrem technologií.

Struktura HTML stránek s *Panelem odjezdů* je pro část s hledanými daty popsána v příloze A. Aby bylo možné získat správné HTML pro hledanou zastávku, je nutné podívat se i na tvorbu URI *Panelu odjezdů*. Např. pro aktuální informace o zpožení vozidel ze zastávky Tylova je URI následující:

```
http://jizdnirady.pmdp.cz StationMarker.aspx?ObjectId=Tylova
```

Parametr `ObjectId` udává jedinečnou identifikaci zastávky v rámci webu. V případě jednoslovných názvů zastávek je jeho hodnota samotným názvem zastávky. Pokud je ale název zastávky složen z více slov, nahrazuje se mezera znakem '+'. Pro zastávku Ústřední hřbitov tedy bude `ObjectId` rovno `Ústřední+hřbitov`.

Na webu má také každá ze zastávek jedinečnou identifikaci v podobě číselného ID. Pro zmíněnou zastávku Tylova je to např. `60000040`. Toto číslo může nahradit slovní podobu u `ObjectId` a kdykoli se místo něho dosadit do URI, vrácený HTML soubor bude vždy stejný.

Pro zastávky, kde se kříží několik linek, může nastat situace, že hledané vozidlo nebude mezi prvními patnácti. V tuto chvíli lze upravit URI přidáním parametru `DateTime`.

Jeho hodnotou je spojení data a času ve formátu:

`den.měsíc.rok+hodiny:minuty:vteřiny`

Například pro 22.3.2014 ve 13:20 bude `DateTime` rovno `22.3.2014+13:20:00`.

5.3.2 Výpočet GPS souřadnic vozidel

Samotný údaj o zpoždění vozidla pro výpočet jeho GPS souřadnic samozřejmě nestačí. Je nutné znát souřadnice bodů, mezi kterými se vozidlo v danou chvíli pohybuje. Jinými slovy potřebujeme získat GPS souřadnice všech průjezdních bodů vozidla, zejména pak zastávek jeho trasy.

Jelikož pozice zastávek (a dalších průjezdních bodů trasy vozidel) také nejsou veřejně dostupnými údaji, do aplikace je bude muset zadat sám uživatel. Aby se souřadnice nemusely zadávat při každém spuštění aplikace znovu, budou tyto souřadnice ukládány pro pozdější využití.

Pro hledání vozidel jedné linky nebo jednoho směru linky (jak je uvedeno ve specifikaci požadavků) je potřeba nejprve získat její trasu, tedy přesné pořadí zastávek od počáteční do cílové. Trasy všech linek jsou uloženy také na webu PMDP v sekci *Zastávkové JŘ* (viz kapitola 4.3.3). Jednou z možností by bylo při hledání vozidel vždy nejprve získat trasu z této stránky a následně hledat vozidlo mezi zastávkami. To by vyžadovalo velké množství přístupů na web, což je nežádoucí z důvodu pomalého přístupu k webu a také z možného zahlcení webového serveru. I trasy vozidel by se tedy měly jednou uložit a následně využívat již sesbíraná data.

Možností, jakým způsobem nakládat s ukládanými údaji, je více. Lze využít ukládání do textových souborů, do XML nebo do databáze (běžící na serveru nebo tzv. vnořené – embedded).

5.3.3 Možnosti poskytnutí dat ostatním aplikacím

Jedním z požadavků na aplikaci je možnost předávat vypočítaná data externím programům (viz kapitola 5.1.3). Stejně jako pro ukládání souřadnic průjezdních bodů zmíněné v předchozí podkapitole je možné předat vypočítaná data pomocí exportu do textových souborů (zřejmě s nějakým známým uspořádáním, např. pomocí CSV souborů), XML souborů nebo data uložit do databáze, kde by mezi nimi mohl jiný program kdykoli vyhledávat.

Další možností je využít posílání dat přes počítačovou síť a zvolit nějaký univerzální protokol, nebo využít další možnosti zvoleného programovacího jazyka.

5.4 Výběr technologií

5.4.1 Zvolená architektura

Aplikace bude dodržovat klasickou třívrstvou architekturu. Architektura je zvolena na základě požadovaného uživatelského rozhraní a nutnosti ukládat a načítat velké množství dat.

5.4.2 Volba jazyka

Pro aplikaci je vhodné použít objektový programovací jazyk, který obsahuje knihovny na parsování. Vzhledem ke zkušenostem s programováním v jazyce Java je tento jazyk vybrán pro vývoj aplikace – konkrétně jeho verze 1.7.

5.4.3 Výběr úložiště

Jak již bylo zmíněno v kapitolách 5.3.2 a 5.3.3, některá data musí být ukládána pro pozdější využití. Pro tento účel je vybrána embedded databáze HyperSQL DataBase (HSQLDB) [12]. Důvodem je lepší způsob vyhledávání (formou SQL dotazů) a hlavně možnost snadno data měnit. Oproti klasické databázi je výhoda v nezávislosti na serveru. Přenos embedded databáze probíhá prostým kopírováním souborů a lze tak jednoduše zálohovat data.

5.4.4 Export dat

Za účelem komunikace s programy pro simulaci silniční dopravy je zvolen export do XML. Tato volba je dána zejména přenositelností XML souborů a jejich snadnou čitelností jak strojově, tak lidským okem. Pro práci s XML existuje v jazyce Java celá řada knihoven zabývajících se načítáním a ukládáním z/do souborů.

Návrh XML pro export dat je v příloze B.

Pro práci s XML je k dispozici několik parserů. Nejznámějšími jsou *SAX*, *StAX*, *DOM* a *JAXB*, které budou stručně popsány v následujících odstavcích [13].

Parser *SAX* využívá proudové zpracování XML. Narozdíl od parserů *DOM* a *JAXB* proto potřebuje výrazně menší množství paměti při zpracování velkého množství dat. Jeho nevýhoda pak spočívá právě v proudovém přístupu. Není totiž možné

„přeskakovat“ v rámci XML elementů a číst může jen z jedné pozice. Používá se prakticky jen pro čtení, takže pro export dat není vhodný.

Parser *StAX* využívá také proudový přístup, proto výhody (i nevýhody) jsou stejné jako u *SAX*. Pomocí tohoto parseru lze ale i zapisovat do souborů a vytvářet nové dokumenty.

Největší přednosti a nedostatky parseru *DOM* jsou přesně opačné. *DOM* umožňuje zapisovat i číst všechny elementy v kterýkoli okamžik. Aby toho ale dosáhl, musí být v paměti načtena celá struktura XML stromu. Tím pádem při větším objemu dat vznikají výrazné paměťové nároky.

JAXB je parserem s podobným přístupem jako má *DOM*. Struktura XML stromu je ale načítána do speciálních objektů vygenerovaných z XSD souboru, který se využívá pro popis XML.

Předpokládaný objem dat pro export do XML je v řádech stovek řádků. Načítání celého stromu do paměti tedy nehraje důležitou roli při volbě parseru. Jelikož *JAXB* ulehčuje práci s čtením souboru, je vybrán právě tento parser. Navíc při importu objektů vytvořených ze souboru XSD může externí program pro simulaci silniční dopravy jednoduše načítat exportovaný XML dokument.

5.4.5 Prezentační vrstva

Pro tvorbu grafického uživatelského rozhraní je v jazyce Java možno použít několik knihoven. Mezi nejznámější patří *AWT* [14], *JavaFX* [15] a *Swing* [16].

Knihovna *AWT* je součástí Java Core API od verze 1.0, ale v dnešní době je považována za zastaralou. Její použití proto není vhodné.

Naopak *JavaFX* je z trojice knihoven nejmodernější a v budoucnu bude nejspíše stále více rozšířená. Tato knihovna nicméně není součástí jazyka Java ve verzi 1.7, ve které bude aplikace vyvíjena.

Pro vytváření uživatelského rozhraní bude využita knihovna *Swing*, která je přímo součástí jazyka Java v použité verzi 1.7.

5.4.6 Další potřebné knihovny

Při vytváření aplikace je vhodné použít i další knihovny výrazně usnadňující práci s parsováním HTML a práci s časem.

Pro parsování HTML byla zvolena knihovna *JSoup* [17]. Ta obsahuje všechny potřebné funkce jako je vyhledávání elementů podle zadaného ID nebo třídy a vyhledávání podle specifického zanoření elementů. Navíc se při získávání HTML z webové adresy nemusí používat HTTP příkazy, ale knihovna si ze zvolené adresy stáhne HTML sama.

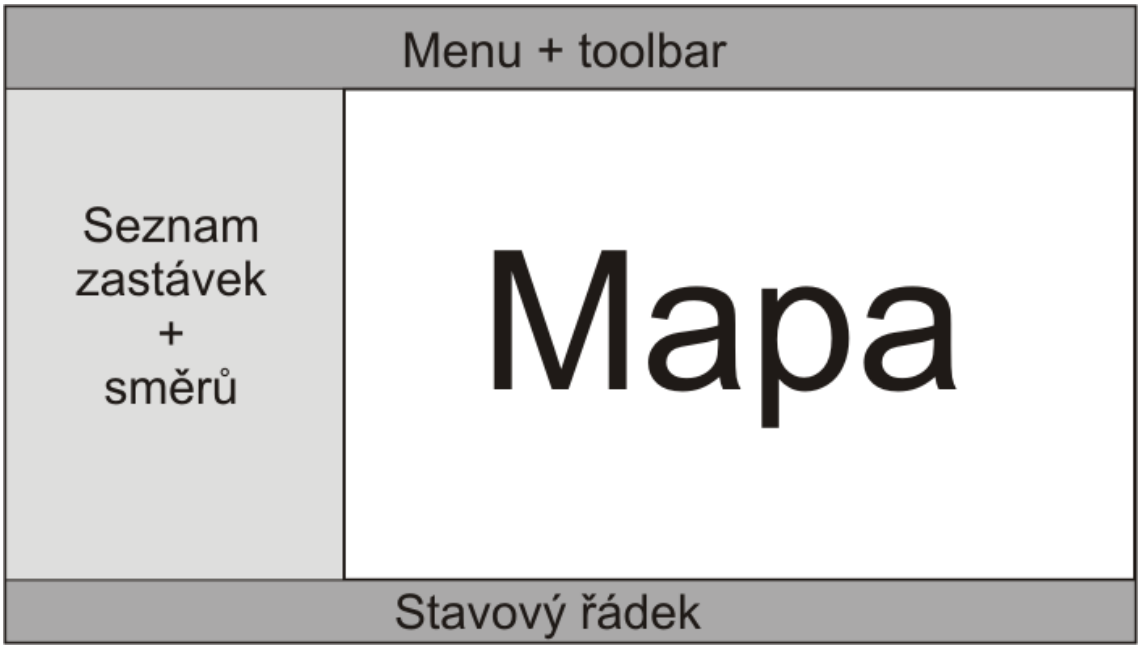
Pro práci s časem byla zvolena knihovna *Joda-Time* [18]. Ta usnadňuje práci s daty a časy. Pomocí knihovny lze totiž časy porovnávat, provádět operace přičítání i odečítání časů a kombinovat data a časy dohromady do jednoho objektu.

5.5 Návrh uživatelského rozhraní

Prostřednictvím grafického rozhraní bude uživatel nastavovat GPS souřadnice zastávkám. Musí se mu tedy zobrazovat seznam zastávek, aby z nich mohl vybírat. Současně musí mít možnost volit ze směrů linek, protože při sbírání informací o vozidlech není vhodné, aby se prohledávaly všechny linky MHD najednou.

Pro zpětnou vazbu od aplikace k uživateli by mělo rozhraní obsahovat tzv. stavový řádek. Ten by měl uživateli dávat na vědomí informace o vykonané akci (např. ho informovat o úspěšně uložených změnách).

Aby mohl uživatel pohodlně zadávat souřadnice, měla by mu být poskytnuta mapa Plzně, do které by zastávky dosazoval. Jednoduchý návrh rozhraní viz Obrázek 12.



Obrázek 12 – Návrh uživatelského rozhraní

Pro uživatelské rozhraní je tedy potřeba mapa Plzně a jejího nejbližšího okolí. Možností, odkud získat tuto mapu, je více. Mezi nejznámější webové stránky s mapami,

které obsahují podrobně zpracované zobrazení Plzně, patří *Google Maps* [19], *Mapy.cz* [20] a *Open Street Map* [21].

5.5.1 Google Maps

Asi nejznámějšími webovými mapami na světě jsou Google Maps. Plzeň i její okolí jsou zde podrobně zpracované, zobrazují se dokonce i zastávky MHD. Jasnou výhodou je i použitý typ projekce pro zobrazení zemského povrchu. Ve zkratce lze říci, že pokud uvažujeme mapu obdélníkového tvaru, tak horní (dolní) rohy mapy budou mít stejné souřadnice pro zeměpisnou šířku a levé (pravé) rohy zase stejné souřadnice pro zeměpisnou délku.

5.5.2 Mapy.cz

Asi nejpodrobnější mapy pro Českou republiku jsou Mapy.cz. Také obsahují vyznačené zastávky MHD, nicméně nevýhodou oproti Google Maps je použitý typ projekce. Neplatí zde pravidlo o stejných souřadnicích pro rohové body, proto by výpočet souřadnic kdekoli na mapě musel být prováděn složitějším algoritmem.

5.5.3 Open Street Map

Open Street Map je významným projektem zabývajícím se mapami pod otevřenou licencí. Jeho zásadní nevýhoda ale spočívá v tom, že údaje o mapách do systému vkládají sami uživatelé a proto by mohlo docházet (zejména v okrajových místech potřebné mapy) k nepřesnostem ve struktuře ulic a bloků. Také vzhled map nepatří mezi pozitiva tohoto projektu.

5.5.4 Vyhodnocení výběru mapy

Z důvodu možného použití rychlejšího a jednoduššího algoritmu pro zanášení bodů do mapy jsou vybrány Google Maps. Oproti Open Street Map pak mají výhodu především ve vyšší přesnosti a „lepším“ vzhledu. Jejich použití v rámci diplomové práce je možné při uvedení správných atributů s autorskými právy do pravého dolního rohu mapy [22].

5.6 Databázová struktura

5.6.1 Jízdní řády

Návrh základní databázové struktury vychází z principu fungující městské hromadné dopravy. Musí zde být tabulky charakterizující:

- zastávky,
- linky,
- směry linek (těch může být několik v rámci jedné linky),
- pořadí zastávek pro jeden konkrétní směr,
- časové odjezdy ze zastávek.

5.6.2 Typy dnů a vozidel

V kapitolách 4.2.1 a 4.2.2 jsou popsány typy vozidel a typy dnů, které se v rámci MHD v Plzni uvažují. Na jednotlivé typy dnů se váží časy odjezdů ze zastávek, protože např. ve všední dny bývají spoje vytíženější a vozidla proto jezdí častěji. Typy vozidel pak mohou rozšířit filtr exportu pozic vozidel týkající se jen tramvajových linek atd. Obě tyto vlastnosti rozšiřující výše popsané tabulky by měly být do návrhu databáze také zapracovány.

Ve spojitosti s typy dnů je nutné zmínit státní svátky, pro něž jsou spolu s nedělí vystaveny speciální jízdní řády. Seznam státních svátků tedy musí být také uložen, aby se mohlo při běhu aplikace nejprve testovat, zda den vyhledávání není jedním z nich. Ve chvíli, kdy je pro ukládání všech údajů vybrána databáze, není důvod neuložit svátky do speciální tabulky. Výjimku mezi svátky pak tvoří velikonoční pondělí. To totiž nemá pevně daný datum a musí se vždy pro konkrétní rok počítat. Bylo by zbytečné před každým vyhledávání vozů pomocí speciálního algoritmu počítat, zda daný den není právě velikonočním pondělím. Proto při vytváření modelu databáze budou spočítána data těchto svátků na několik let dopředu a uložena pro pozdější databázové dotazy.

5.6.3 Sběr souřadnic vozidel

Dalšími údaji uchovávanými v databázi budou získané pozice vozidel. Pro jedno měření v určitý čas bude typicky nalezeno více než jedno vozidlo. Z toho bude provázání těchto tabulek dále vycházet. Možnost exportu do XML s filtrováním údajů přímo vyžaduje, aby bylo možné tvořit různé dotazy nad získanými souřadnicemi vozidel. Z důvodu (ne)pravidelné změny jízdních řádů popsané v kapitole 4.2.4 je nutné promyslet provázání údajů o nalezených vozidlech s uloženými jízdními řády.

Při aktualizaci jízdních řádů totiž může nastat situace, že byly některé z linek nebo zastávek zrušeny. Pokud by měly tabulky pro linku a vozidlo mezi sebou vazbu pomocí

cizího klíče, nastala by při smazání linky nekonzistence dat, případně by došlo ke smazání údajů o nalezeném vozidle. Řešením je vždy při ukládání pozice vozidla duplikovat název linky do jeho tabulky.

Podobné řešení bohužel není vhodné aplikovat na případy rušení zastávek. Důvodem je, že by se kromě názvu zastávek musela duplikovat data i o jejich pozicích. Takové množství duplikovaných dat pak není žádoucí. V tu chvíli pak nezbývá než údaje o zastávkách v databázi nechávat i při jejich rušení nebo přejmenování. K takovým změnám nedochází příliš často, proto jedinou nevýhodou bude zobrazení zastávky nepatřící k žádnému směru v seznamu všech zastávek.

6 IMPLEMENTACE

V této kapitole se budeme zabývat implementací aplikace a to všech tří vrstev její architektury. Nejprve bude popsána obecná struktura aplikace a poté detailněji některé její části včetně použitých algoritmů.

6.1 Struktura aplikace

V rámci vývoje aplikace byla použita následující struktura balíků pro přehledné uspořádání tříd:

- *dao* – obsahuje DAO třídy (viz kapitola 6.2.2) a také třídu `DaoManager` (viz kapitola 6.2.3) pro přístup k datům uloženým v databázi.
- *dataCollecting* – jsou zde třídy pro sběr dat z webu PMDP, tzn. pro aktualizaci databáze a pro měření pozic vozidel.
- *dataObjects* – obsahuje objekty získané pomocí DAO tříd z databáze a přenášené až do tříd implementujících uživatelské rozhraní.
- *dbOperations* – zde jsou uloženy třídy pro vytvoření a přístup k databázi.
- *exceptions* – obsahuje všechny výjimky vytvořené pro potřeby aplikace.
- *export* – obsahuje třídy pro exportování vozidel a linek do XML.
- *gui* – zde jsou třídy určené k realizaci grafického uživatelského rozhraní.
- *main* – obsahuje hlavní třídu aplikace, která vytváří hlavní okno GUI.

Pro detailnější informace o struktuře aplikace byl vytvořen UML diagram tříd (viz příloha D), v němž jsou zakomponovány hlavní třídy aplikace.

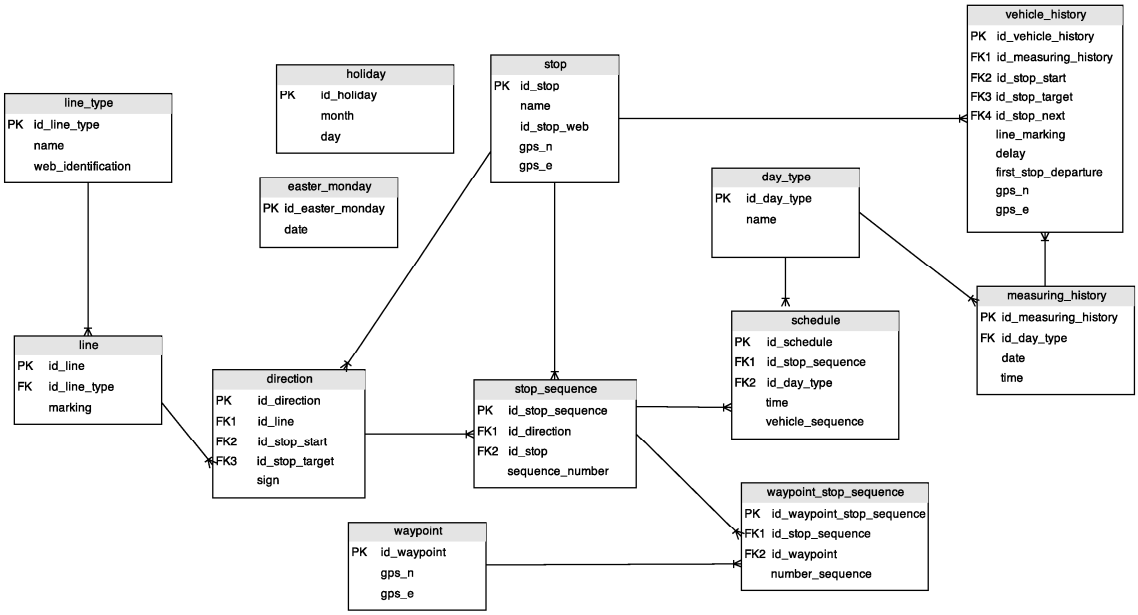
6.2 Vnořená databáze

Jak již bylo zmíněno v kapitole 5.4.3, pro ukládání dat byla zvolena vnořená databáze *HyperSQL DataBase* [12]. Narozdíl od běžně používané databáze běžící na serveru má vnořená databáze rozdílný princip přístupu. Aplikace se totiž běžně při svém spuštění k databázi připojí a odpojují se až při svém ukončení. Pokud by se totiž aplikace připojila a odpojila vždy pro jeden konkrétní databázový dotaz, došlo by ke znatelnému zpomalení. Protože se předpokládá, že k databázi v jednu chvíli přistupuje pouze implementovaná aplikace, tento způsob připojování (kdy je aplikace připojena k databázi celou dobu) je vhodnější.

Při vytváření databáze byly změněny některé její základní parametry. Bylo zakázáno logování vykonávaných příkazů kvůli vyšší rychlosti přístupu k databázi. Také bylo vynuceno dodržování maximální velikosti řetězců pro sloupce tabulek uchovávajících text. Toto nastavení je použito z důvodu úspory paměti při ukládání dat.

6.2.1 Databázový model

Návrh databáze vychází z analýzy dat popsané v kapitole 5.6. Jsou zde tabulky pro uchování jízdních řádů, pro ukládání jednotlivých měření pozic vozidel a také pro určování, zda je konkrétní den svátkem či nikoliv. Celý model databáze viz Obrázek 13.



Obrázek 13 – ERA model databáze

Pochopení významu všech těchto tabulek – a tím i objektů (viz kapitola 6.3) – je nutné pro porozumění celé aplikace. Proto budou v následujících odstavcích důkladněji rozebrány.

Každá zastávka v tabulce `stop` má uložený svůj název i ID z webu PMDP. Přesto tabulka obsahuje, stejně jako všechny ostatní tabulky, speciální primární klíč. Tyto klíče jsou pro všechny tabulky pojmenovány `id_NAZEV_TABULKY`, pro tabulku se zastávkami je tedy název klíče `id_stop`. Dalšími parametry jsou GPS souřadnice zastávky, které musí zadávat uživatel prostřednictvím GUI.

Typy MHD linek zmíněné v kapitole 4.2.1 jsou uloženy v tabulce `line_type`. Každý z typů má své pojmenování (např. *Tramvaj*) a jednoznačný webový identifikátor, díky kterému se odlišuje styl jeho zobrazení ve výpisu linek na webu PMDP (např.

linesList_TramStyle). Všechny linky (tabulka `line`) mají přiřazen jeden z těchto typů a obsahují navíc parametr pro značení dané linky.

Každá z linek má také několik směrů (tabulka `direction`), které mají nastaveny zastávky, odkud vozidla vyjíždějí a jaká je jejich cílová stanice. Také obsahují parametr pro značení směru (viz kapitola 4.2.1). Pro každý směr je nutno uchovávat i kompletní seznam zastávek a jejich pořadí ve směru. Tyto informace jsou uloženy v tabulce `stop_sequence`. Mezi jednotlivými zastávkami směru navíc musejí být průjezdní body, protože trasa z jedné zastávky do druhé typicky není přímá cesta, ale vozidlo v průběhu zatáčí. Tyto body jsou uloženy v tabulce `waypoint` včetně svých GPS souřadnic. Databázový model umožňuje přiřazení jednoho průjezdního bodu pro více směrů a to pomocí tabulky `waypoint_stop_sequence`. Ta obsahuje i pořadí průjezdního bodu z jedné zastávky do druhé.

Pro každou zastávku v určitém směru existují časy odjezdů (tabulka `schedule`). Ty mají parametr `vehicle_sequence` určující pořadí vozidla odjíždějícího v daný typ dne z jedné zastávky směru. Lze tak sledovat vozidlo od jeho výjezdu až do cílové stanice. Typy dnů jsou uloženy v tabulce `day_type` spolu s jejich názvem (např. *Sobota*).

V tabulce `measuring_history` jsou uchovány časy a data jednotlivých měření pozic vozidel. K této tabulce se pak váží informace o samotných vozidlech uložené v tabulce `vehicle_history`. Zde jsou zapsány zastávky, odkud vozidlo vyjíždělo, jaká je jeho cílová stanice a před kterou zastávkou bylo v době měření. Navíc je zde obsaženo značení linky vozidla, zpoždění v době měření, čas odjezdu z první zastávky a samozřejmě jeho vypočítané GPS souřadnice.

Posledními tabulkami jsou `easter_monday` uchovávající data velikonočních pondělí (pro jejich výpočet byl použit upravený algoritmus výpočtu velikonoční neděle [23]) a tabulka `holiday`, kde jsou uloženy dny a měsíce státních svátků. Mezi těmito tabulkami a ostatními v databázi neexistuje žádný vztah. Slouží jen pro uchování informací, které využívá aplikace k určení, zda je některý den státním svátkem či nikoli.

6.2.2 Přístup k databázi

V balíku `dao` je pro každou tabulku tzv. DAO třída. Ta obsahuje známé CRUD (*Create*, *Read*, *Update*, *Delete*) operace, tedy umožňuje vkládat do tabulek nové řádky nebo je číst, měnit či mazat.

Všechny DAO třídy mají společného předka, abstraktní třídu `AbstractDao`. Ta svým potomkům poskytuje základní objekty pro přístup k databázi a vykonávání příkazů. Jedním z těchto objektů je instance třídy `DatabaseAccess`, pomocí níž mohou DAO třídy získat připojení k databázi.

Pro samotné vykonávání SQL dotazů v DAO třídách slouží objekt typu `Prepared-Statement`. Ten umožňuje pro řetězce s dotazy nastavovat parametry různých datových typů. Parametry jsou v řetězcích nahrazeny znakem otazníku. Např. pro dotaz `"SELECT * FROM line WHERE id_line=?"` se nastaví hledané identifikační číslo linky jako:

```
preparedStatement.setInt(1, 5);
```

kde v tomto případě `1` je pořadí parametru (v dotazu uveden pouze jeden) a `5` je ID hledané linky.

6.2.3 Třída DaoManager

Třída `DaoManager` slouží v programu pro přístup ke všem DAO třídám, aby ostatní objekty, které chtějí přistupovat k databázi, nemusely uchovávat instance všech DAO tříd. Kromě toho se v jejích metodách může zkombinovat přístup k více tabulkám a lze tak např. získat jedním voláním všechny linky s nastavenými směry.

6.3 Datové objekty

Všemi vrstvami architektury aplikace procházejí objekty z balíku `dataObjects`. Ty korespondují s tabulkami databáze, proto pro většinu tabulek (kromě tabulek pro určení, zda je některý den státním svátkem) je v aplikaci právě jeden typ objektu, který dokáže uchovat stejné množství informací. Objekty jsou pojmenovány podle názvů tabulek.

Cizí klíče tabulek jsou v objektech nahrazeny seznamy. Tedy např. pro jednu linku, která má několik směrů, existuje v objektu `Line` seznam objektů `Direction`. Díky tomu lze přenášet kompletní jízdní řády pro jednu linku (včetně časů odjezdů z jednotlivých zastávek) celou aplikací.

6.4 Výjimky

Při běhu aplikace může dojít k několika očekávaným chybám. Základní výjimky, které provází celou aplikaci jsou `SQLException` a `IOException`.

První ze zmíněných – `SQLException` – může nastat při jakémkoli přístupu k databázi. Ať už při připojování, vykonávání příkazu nebo při jejím odpojování. K výjimce `IOException` pak v aplikaci může dojít zejména při připojování na web PMDP.

Při těchto výjimkách aplikace přestává správně fungovat, proto jejich výskyt ihned ukončuje vykonávání činností jako je aktualizace databáze, sběr dat, ale třeba i ukládání změn do databáze.

Další výjimkou vyskytující se při sběru dat a při aktualizaci databáze je `TaskCancelledException`. Ta indikuje, že uživatel během vykonávání těchto činností stiskl tlačítko pro jejich přerušení.

6.5 Získání jízdních řádů

Pro správnou funkčnost aplikace musejí být jízdní řády v databázi stejné jako jejich aktuální verze na webu PMDP. Proto třída `DatabaseUpdater` obsahuje metody, které mohou databázi naplnit, pokud je prázdná, nebo ji aktualizovat. Tato třída dědí od třídy `SwingWorker`, protože aktualizace databáze trvá delší dobu a musí být o jejím průběhu uživatel informován. Během aktualizace se uživateli dostává informací o jejím stavu prostřednictvím objektu typu `ProgressMonitor`. Zároveň v průběhu aktualizace je uživateli zabráněno ve vykonávání jakýchkoli činností, díky implementaci `Progress-Monitor` může pouze aktualizaci zrušit. V tu chvíli ale nejsou uložená data konzistentní a při nejbližší příležitosti je nutné spustit aktualizaci znovu.

Aby uživatel věděl o nekonzistenci dat, aplikace po úspěšné aktualizaci databáze zapíše příznak o správném průběhu do souboru. Vždy, když se aplikace spouští, pak tento příznak testuje a pokud není v souboru uložen, upozorní uživatele, že by měl databázi znovu zaktualizovat.

Tento problém může nastat i pokud aktualizace skončí během své činnosti, protože nastane nějaká chyba. Chybou může být v tomto případě např. výskyt jedné z výjimek popsaných v kapitole 6.4.

6.5.1 Postup aktualizace

Při aktualizaci jízdních řádů je nutné zachovat v databázi všechny zastávky – i ty, které již v žádném směru nejsou zahrnuty (viz kapitola 5.6.3). Linky a jednotlivé směry jsou ale z databáze smazány vždy, když nejsou aktuální.

Před aktualizací se tedy z databáze musejí získat všechny linky a postupně se pak porovnávají s nalezenými linkami na webu PMDP. Jednou z možností by bylo všechny linky před začátkem aktualizace smazat, nicméně tím by se ztratilo propojení jednotlivých zastávek pomocí průjezdních bodů a uživatel by po aktualizaci musel znovu každou trasu linky zadat do mapy.

Samotná aktualizace je prováděna následujícím způsobem. Jak již bylo řečeno, nejprve se získají všechny linky z databáze. Poté se pro každou z nalezených linek na webu PMDP provede porovnání (pomocí jedinečného názvu linky) s těmito linkami. Pokud v databázi linka z webu není, pak se do ní přidá. Pokud se v databázi vyskytuje, jsou z databáze získány všechny její směry s nastaveným pořadí zastávek.

V tu chvíli se procházejí všechny směry linky nalezené na webu a porovnávají se s těmi v databázi. V databázi se nemohou vyskytovat pro stejnou linku dva směry, které by měly shodné označení, první a poslední zastávku, ale rozdílnou trasu mezi nimi. Při nalezení takového směru je tedy tento z databáze smazán a uložen nový.

Pro každý z nalezených směrů jsou do databáze přidávány odjezdy jeho vozidel z každé zastávky. Nalezení těchto odjezdových časů je popsáno v kapitole 6.5.2.

6.5.2 Přidávání odjezdových časů vozidel

Časy odjezdů ze zastávek jednoho směru jsou získávány ze stránky detailu směru (viz Obrázek 9). Ukládání těchto časů je zpracováno zvlášť pro první zastávku a všechny ostatní. Hlavním důvodem je, že z první zastávky se sbírají i časy odjezdů pro směry se speciálním označením. Tyto směry totiž nejsou nikde přímo uvedeny, proto se při výskytu času odjezdu s nějakým označením vyhledává konkrétní vozidlo v záložce *Panel odjezdů* (viz kapitola 4.3.2). Po nalezení tohoto vozidla lze prozkoumat jeho detail spoje (viz Obrázek 7). Zde jsou uvedeny všechny zastávky trasy vozidla včetně jeho odjezdů a mohou se tedy uložit do databáze.

Druhým důvodem pro rozdělení ukládání z první zastávky a všech ostatních je, že návrh aplikace předpokládá možnost sledovat vozidlo po celou jeho trasu (v databázi je

uloženo pro každý odjezdový čas pořadí vozidla na dané trase – parametr `vehicle_sequence`). Na stránkách s detailem směru se ale pro každou ze zastávek může vyskytovat rozdílné množství odjezdových časů. Je to tím, že vozidla se mohou do trasy připojit až v jejím průběhu, aniž by jejich časy měly speciální označení. Proto je nutné uložit všechny odjezdové časy z první zastávky do proměnné a v dalších zastávkách pak nebrat v úvahu přebytečné časy. Proměnná `stopSchedules` uchovávající tyto časy je typu `HashMap`, přičemž jejími klíči jsou ID typu dne a hodnotami seznamy časů odjezdů pro každé z těchto ID.

6.6 Určení souřadnic vozidel

Hlavní náplní implementované aplikace je sbírání GPS souřadnic vozidel. Tuto funkčnost zajišťuje třída `VehicleSearcher`, která je stejně jako `DatabaseUpdater` potomkem třídy `SwingWorker`. Může se proto opět jednoduše využít instance třídy `ProgressMonitor` a uživateli dávat na vědomí informace o postupu práce.

6.6.1 Nalezení vozidel na trati

Jednou z možností, jak by se dala nalézt vozidla určitého směru na trati, by mohlo být prohledávání všech *Panelů odjezdů* zastávek zadaného směru. V případě, že by se vozidlo nenašlo na první stránce *Panelu odjezdů*, muselo by se hledat na další a znovu se připojit k webovému serveru.

Protože ale není žádoucí, aby aplikace server zbytečně vytěžovala, nejprve se lokálně určí, která vozidla jsou pravděpodobně na trati, a ty se až poté hledají na webu. Pro toto určení stačí jediný přístup na web PMDP a to na *Panel odjezdů* z předposlední zastávky daného směru. Odtud lze získat čas prvního vozidla (jakékoli linky), které má do zastávky přijet.

Získaný čas prvního vozu se využije pro hledání vozidel daného směru v databázi. Vozidla se hledají z uložených časů odjezdů a získají se z databáze všechna mezi prvním na trati (tj. tím, které má nejbližší odjezd z předposlední zastávky) a posledním (tj. tím, které odjelo naposledy z první zastávky).

Pro každé nalezené vozidlo se z *Panelu odjezdů* předposlední zastávky získá jeho zpoždění. Pomocí něho se dále určí, jaká je předpokládaná pozice vozidla na trati, tj. do jaké zastávky pravděpodobně právě míří. Tato zastávka se určí binárním vyhledáváním.

Pro znázornění zde uveďme příklad:

Čas t_{web} na webu PMDP je po synchronizaci (viz kapitola 6.6.2) určen jako 21:27. Jedna z tramvají linky 1 ve směru *Slovany* → *Bolevec* odjížděla z první zastávky v 21:15 a z webu bylo zjištěno, že má v čase t_{web} 3 minuty zpoždění. Protože u každé z tramvají je v databázi uloženo pořadí, kolikátá v daný den vyjížděla z první zastávky, lze pro tuto tramvaj identifikovat časy odjezdů z kterékoli další zastávky na trati.

Linka 1 má dohromady 18 zastávek, proto binární vyhledávání začíná v deváté v pořadí (zastávka *Náměstí republiky*), tj. v polovině intervalu 1–17. Poslední zastávka se neuvažuje, protože neobsahuje odjezdy vozidel. Z databáze se následně zjistí, že hledaná tramvaj má čas odjezdu z této zastávky ($t_{\check{r}\acute{a}d}$) 21:25. Protože má ale 3 minuty zpoždění, počítá se s tím, že je tramvaj v místě, kde by běžně byla v čase $t_{web} - zpoždění$, tedy v 21:24. Jelikož čas 21:24 je menší než $t_{\check{r}\acute{a}d}$, pro další hledání se vezme první polovina 17-ti zastávek trasy a všechny ostatní od zastávky *Náměstí republiky* (včetně) se už dále pro vyhledávání neuvažují. Další zastávkou, pro kterou se porovnávají časy pak bude pátá v pořadí (zaokrouhlená polovina intervalu 1–8) a pokračuje se stejným postupem.

Tímto způsobem se pro každé vozidlo nalezne zastávka, do které v čase synchronizace pravděpodobně míří. Po spočítání pravděpodobného umístění vozidla nastává kontrola s webem PMDP, jestli je opravdu vozidlo před určenou zastávkou. Z jejího *Panelu odjezdů* se určí, zda už v ní vozidlo bylo nebo do ní teprve přijede. Pokud už v ní bylo, vezme se další zastávka v pořadí a i na jejím *Panelu odjezdů* se vozidlo hledá. Nyní mohou nastat dvě možnosti:

- Pokud do této zastávky vozidlo ještě nedojelo, je tímto nalezena zastávka, kam vozidlo míří.
- Pokud i v této zastávce vozidlo bylo, prozkoumá se opět další zastávka v pořadí a použije se znovu stejný algoritmus.

Naopak když je v předem odhadnuté zastávce odjezd vozidla nalezen, hledá se v *Panelu odjezdů* předchozí zastávky a tam se opět určuje, zda už jí vozidlo projelo. Tímto se přesně zjistí, mezi kterými zastávkami vozidlo opravdu je.

6.6.2 Synchronizace času

V kapitole 6.6.1 byla zmíněna synchronizace času prováděná na začátku měření. Ta je potřeba z toho důvodu, že časy se zpožděním vozidel na webu PMDP se aktualizují vždy jednou za minutu.

Synchronizace lokálního času s webovým probíhá pomocí aktivního čekání. Nejprve se z webu zjistí poslední čas, pro který byla zpoždění vyhodnocena. Tento čas se uloží a následně se aktivní vlákno uspí na určitou dobu (v aplikaci nastaveno na tři vteřiny). Poté vždy po probuzení vlákno zkontroluje čas na webu a pokud se neliší od původního uloženého, zase se uspí. Nanejvýš minutu tedy vlákno čeká, než se na webu čas změní. Tento změněný čas se vezme jako základní pro výpočet a vlákno může začít hledat vozidla.

6.6.3 Problémy s odhady pozice vozidel

Kapitola 6.6.1 naznačila, že i přes lokální výpočet pozice vozidla (tj. před jakou je zastávkou) se tato pozice nemusí slučovat s realitou. Důvodů je pro to několik. Jedním z nich je, že časy zpoždění uvedené v *Panelu odjezdů* zastávek jsou zaokrouhlené na minuty. Proto při uvedeném zpoždění např. 2 minuty může být skutečné zpoždění vozidla např. 1:31–2:30 (minut:sekund). Také samotné časy odjezdů vozidel jsou zaokrouhlené na celé minuty, proto odjezd vozidla v 21:26:00 může být ve skutečnosti 21:25:31 nebo 21:26:30.

Při kombinaci těchto dvou extrémů tak dochází k rozdílu až jedné minuty, která v odhadu pozice vozidla hraje důležitou roli. Časy odjezdů mezi dvěma po sobě jdoucími zastávkami s kratší vzdáleností jsou totiž právě 1 minuta.

6.6.4 Výpočet souřadnic

Samotný výpočet souřadnic vozidla spočívá zejména v určení, mezi kterými průjezdními body se vozidlo nachází. Nejprve se tedy pro celou cestu z jedné zastávky (odkud vozidlo naposledy vyjelo) do druhé (kam nyní míří) spočítá její vzdálenost v metrech. K tomu slouží algoritmus pro výpočet vzdálenosti mezi dvěma GPS body [24]. Poté se v závislosti na poměru ujeté vzdálenosti snadno určí, mezi kterými body se vozidlo nachází a kde přesně. Tento poměr se počítá z odjezdových časů vozidla. Tedy pokud doba jízdy mezi zastávkami trvá 3 minuty a vozidlo je mezi nimi zatím 1 minutu, tak ujelo přibližně třetinu cesty.

6.6.5 Opakovaná měření

Uživatel může prostřednictvím grafického rozhraní nastavit počet měření, které se mají provést, a také časový interval mezi nimi. Časovým intervalem je zde myšlen časový rozdíl mezi začátky měření. Tedy ve chvíli, kdy měření začíná, musí se vypočítat začátek dalšího měření. Po skončení prvního výpočtu souřadnic se pak určí, zda další měření začne hned (tj. získávání souřadnic trvalo delší dobu, než je velikost časového intervalu) nebo bude vlákno čekat do začátku dalšího měření.

6.7 Přístup k webu PMDP

Pro přístup k webu PMDP se využívá třídy `WebAccess`, která poskytuje nejen objekty obsahující HTML elementy kompletní webové stránky, ale i vnitřní HTML elementy nebo třeba jejich parametry. Oddělením kódu pro přístup k webu se aplikace stává více udržitelnou pro budoucí úpravy. Ty by byly potřeba zejména při změně struktury webu, kdy by nebylo možné nalézt některé z potřebných informací na místech, kde je aplikace předpokládá.

6.8 Export dat

Pro zajištění exportu vozidel pomocí parseru JAXB vybraného v kapitole 5.4.4 se nejprve navrhla struktura XML dokumentu (viz příloha B). Práce s JAXB vyžaduje vytvoření XSD souboru, který popisuje strukturu XML dokumentu, aby se z něho následně vygenerovaly objekty pro práci s elementy v programu.

6.8.1 XSD soubor

Při vytváření XSD souboru se začíná od nejjednodušších datových typů pro základní elementy a parametry a postupně se z těchto datových typů skládají složitější XML elementy. Příkladem je jednoduchý datový typ (obsahující kladné celé číslo):

```
<xs:simpleType name="idStopType">
   <xs:restriction base="xs:nonNegativeInteger"></xs:restriction>
</xs:simpleType>
```

který se následně přiřazuje jako atribut ID zastávky (`idStop`) složitějšího elementu:

```
<xs:complexType name="stopType">
   <xs:sequence>
      <!-- zde jsou nastaveny vnořené elementy -->
   </xs:sequence>
```

```
   <xs:attribute name="idStop" type="idStopType" use="required" />
</xs:complexType>
```

Pro vygenerování balíku s JAXB objekty z XSD souboru je využito sestavovací utility *Ant*, která spouští soubor `xjc.exe`. Ten je součástí Java JDK od verze 1.6.

6.8.2 Export vozidel

Ve chvíli, kdy je vygenerován (a do projektu naimportován) balík s objekty potřebnými pro práci s XML dokumentem, mohou se tyto objekty začít používat v programu. Vytváření objektů reprezentujících složitější elementy probíhá pomocí instance třídy `ObjectFactory`, která se vždy vygeneruje do balíku s JAXB objekty. Např. pro výše uvedený element se zastávkou bude vytvoření jeho objektu v programu následující:

```
StopType stop = objectFactory.createStopType();
```

Další třídou potřebnou pro export je `DatatypeFactory` z balíku:

```
javax.xml.datatype
```

Pomocí této třídy lze vygenerovat datový typ `XMLGregorianCalendar` reprezentující datum a čas v XML.

Samotná transformace objektů z databáze na objekty pro export probíhá v samostatných metodách pojmenovaných podle názvu objektů `getNazevXMLFromNazev()`, např. `getStopXMLFromStop()`. Zde se nastavují všechny proměnné objektům. Nevýhodou použití JAXB je nutná přeměna ze základních číselných typů na složitější. Z datového typu `Integer` se tak stane `BigInteger` a z `Double` se stane `BigDecimal`.

6.8.3 Export MHD linek

Oproti základní plánované funkčnosti byla do aplikace přidána i možnost exportu jízdních řádů – tedy linek, jejich směrů a pro každý směr pořadí zastávek (včetně jejich GPS souřadnic). Časy odjezdů nejsou do výsledného XML dokumentu zahrnuty. Struktura tohoto dokumentu je popsána v příloze C.

Výsledný dokument vzniká stejně jako exportovaná vozidla pomocí knihovny JAXB. I pro něj byl tedy vytvořen XSD soubor a vygenerované objekty importovány do aplikace. Exportovaný XML soubor s jízdními řády může být dále využit pro potřeby simulačních systémů.

6.9 Grafické uživatelské rozhraní

V této kapitole bude popsána prezentační vrstva aplikace, která je implementována pomocí knihovny *Swing*. Základní komponenty proto využívají třídy právě z této knihovny.

6.9.1 Struktura tříd

Třídy uživatelského rozhraní patří do balíku `gui`, který je dále rozdělen do několika částí:

- `gui.main` – obsahuje všechny základní komponenty, které se zobrazují uživateli. Je zde např. panel s mapou.
- `gui.mapObjects` – zde jsou objekty, které se vkládají do mapy. Např. kruh vykreslující se na pozici zastávky.
- `gui.trees` – obsahuje všechny potřebné třídy pro seznamy se zastávkami a směry. Ty mají totiž stromovou strukturu a jsou vytvořeny pomocí třídy `JTree`.
- `gui.listeners` – pro interakci s uživatelem se využívá tzv. posluchačů (listenerů), které jsou právě v této části.
- `gui.actions` – obsahuje obsluhy pro akce, kdy uživatel vybere položku v menu nebo panelu nástrojů.

Konkrétní provázání většiny tříd z balíku `gui` viz příloha E s UML diagramem tříd.

6.9.2 Hlavní okno aplikace

Hlavní okno aplikace (třída `MainFrame`) je v podstatě kontejner, do kterého jsou vkládány ostatní komponenty – viz Obrázek 12 s návrhem hlavního okna. Umožňuje také ostatním objektům vypisovat informace uživateli a to prostřednictvím stavového řádku (méně důležité informace, které po vypršení času v `Timeru` samy zmizí) nebo informačního dialogu, který zůstane na obrazovce, dokud ho uživatel nezavře.

Další funkcí této třídy je udržování informace o tom, zda probíhá měření pozic vozidel. To se děje v metodě `setCollectingStatus(int status)`, která je označena přívlastkem `synchronized`. Je tomu tak z toho důvodu, že v momentě, kdy skončí jedno z periodických měření a čeká se na další, má uživatel možnost další měření zrušit. V tu chvíli by ale mohly k proměnné určující stav měření přistupovat dvě vlákna.

Stavy mohou být dohromady čtyři:

- *měření vůbec neprobíhá* – počáteční stav při spuštění aplikace. Nastaveno také ve chvíli, kdy se ukončí vlákno sbírající pozice vozidel.
- *měření probíhá* – vlákno sbírající pozice nyní pracuje.
- *měření je zastaveno uživatelem* – uživatel ukončil měření v době, kdy vlákno sbírající data čekalo na začátek dalšího měření.
- *měření je dokončeno, ale ještě se měřit bude* – vlákno dokončilo měření, ale protože je nastaveno jeho periodické opakování, bude po čase znovu pokračovat.

6.9.3 Seznamy zastávek a směrů

V levé části hlavního okna je umístěn `JTabbedPane` obsahující panely se seznamy zastávek a směrů uložených v databázi. Tyto seznamy mají v obou případech stromovou strukturu, protože pro jeden název zastávky může existovat více míst s různými souřadnicemi a směry se slučují pod názvy linek MHD. Oba seznamy jsou vytvořeny pomocí tříd (`StopTree`, `DirectionTree`), jejichž rodičem je třída `JTree`.

Prvky jsou do stromů vkládány hned při spuštění aplikace a aby bylo pro uživatele snazší najít nějakou zastávku nebo směr, jsou abecedně řazeny. Pro oba stromy jsou tedy vytvořeny třídy reprezentující jeden uzel ve stromu – `StopTreeNode` a `DirectionTreeNode` – a dále pro každý další třída implementující `Comparator`. Ten určuje, podle čeho se budou uzly řadit.

Oba seznamy tedy mají velmi podobnou strukturu a nabízelo by se vytvořit rozhraní, které by oba implementovali. Protože ale už nyní dědí od stejné třídy, bylo by toto rozhraní nadbytečné.

6.9.4 Objekty na mapě

Do panelu obsahujícího mapu (viz kapitola 6.9.5) lze vkládat různé objekty. Ty představují zastávky a průjezdní body mezi nimi (vykresleny jako kruhy), spojení trasy jednoho směru (vykresleno jako čáry mezi kruhy) a nalezená vozidla na trati (vykreslena jako čtverce s vepsaným názvem linky).

Čtverce a kruhy mají společné vlastnosti jako je pozice středu (tj. souřadnice x a y pro zobrazení na mapě, ale i GPS souřadnice skutečného vozidla nebo zastávky) a rozměry. Také mají podobné metody na vykreslení svého obsahu a obvodu. Pro tyto společné

vlastnosti vznikla abstraktní třída `GeometricShape`, od které třídy `Circle` a `Square` dědí.

Čáry mezi kruhy jsou instancemi třídy `LinkLine`. Ta má s dvojicí tříd některé parametry a metody také podobné, např. má určitou barvu a obsahuje metodu pro své vykreslení. Protože se ale v aplikaci používá jen pro propojení kruhů, nemusí obsahovat souřadnice GPS, které mají již kruhy nastaveny, a samozřejmě nemůže rozdělit vykreslování na svůj obsah a obvod. Proto tato třída nemá společného předka jako třídy `Circle` a `Square`.

6.9.5 Třída MapPanel

V panelu `MapPanelWrapper`, který je potomkem třídy `JScrollPane` a tedy umožňuje pro velké objekty zobrazit posuvníky, je umístěn objekt typu `MapPanel`. Ten obsahuje jako podklad mapu Plzně a jejího okolí a na ní vykresluje objekty zmíněné v kapitole 6.9.4. Metoda, jež celý panel vykresluje, nejprve zobrazí podkladovou mapu podle uživatelem zvoleného přiblížení (viz kapitola 6.9.6) a pak až všechny objekty. Při vykreslování objektů se začíná spojovacími čarami, aby nepřekrývaly ostatní tvary.

Při vytváření objektu `MapPanel` mu musí být předány souřadnice jeho horního levého a dolního pravého rohu. Pomocí těchto souřadnic se pro každý geometrický útvar před umístěním do mapy počítá jeho x, y souřadnice v mapě. Díky použití mapy z Google Maps [19] je tento přepočet jednoduchý, protože poměr vzdáleností dvou bodů na mapě vůči šířce nebo výšce mapy je stejný jako poměr rozdílu jejich GPS souřadnic vůči souřadnicím rohů.

6.9.6 Akce uživatele

Uživatel interaguje s aplikací nejen spouštěním měření. Aby bylo možné vůbec měření spustit, musí nejprve nastavit souřadnice zastávkám. Pro tuto a další činnosti slouží v aplikaci třídy zděděné od `AbstractAction`. Tyto třídy musí vždy překrýt metodu pro obsloužení akce – `actionPerformed()`.

Zmíněné přiřazování pozic zastávkám probíhá nastavením parametru `actionActive` v objektu s hlavním oknem `MainFrame`. Tento parametr udává právě probíhající akci a lze ho zjišťovat např. v posluchačích (viz kapitola 6.9.7) a podle jeho nastavení provádět různé činnosti při kliknutí myší do mapy atd.

Další důležitou akcí, kterou může uživatel spouštět, je zoomování (neboli přibližování) mapy. Mapa je v aplikaci uložena ve dvou velikostech a podle zvoleného přiblížení se uživateli zobrazí s více či méně detaily. Obsluha přiblížení (a oddálení) probíhá ve třídě `ActionZoom`. Protože při zobrazení mapy mohou být nastaveny posuvníky u `JScrollPane` na nějaké specifické místo, je žádoucí, aby i při změně velikosti mapy zůstaly tyto posuvníky na stejných pozicích. Toho se dosáhne spočítáním poměru středu posuvníku vůči délce (nebo výšce) panelu a po změně velikosti mapy se z tohoto poměru vypočítá nová pozice posuvníku a opět se nastaví. Tyto změny a opětovné vykreslení mapy včetně všech jejích objektů musí být pro správnou funkčnost provedeny pomocí metody `SwingUtilities.invokeLater()`. Ta zajišťuje asyn–chronní provedení změn na uživatelském rozhraní.

6.9.7 Posluchači

V kapitole s uživatelskými akcemi bylo zmíněno, že v aplikaci jsou použity také posluchači (neboli instance tříd dědících od třídy `NAZEVListener` – například `ChangeListener` – nebo od tříd `NAZEVAdapter` – například `MouseAdapter`). V případě, že pak nastane nějaká událost, kterou posluchači mají za úkol sledovat, provedou samy její obsluhu.

Objektu typu `MapPanel` je v aplikaci přiřazen posluchač `MapPanelMouseList`, který obsluhuje události vygenerované uživatelem pomocí myši. Tento posluchač zpracovává přidávání nových zastávek a průjezdních bodů do mapy, posunování zastávek na mapě nebo také zobrazení kontextového menu při kliknutí pravým tlačítkem myši na jednu zastávku. Důležitá je i funkce tohoto posluchače, která zajišťuje změnu zobrazené části mapy při tzv. drag-and-drop, tedy držení tlačítka myši při jejím pohybu.

7 TESTOVÁNÍ APLIKACE

V této kapitole se budeme zabývat testováním vytvořené aplikace a to jak její správné funkčnosti, tak i její rychlosti.

7.1 Testy funkčnosti

Pro ověření aplikace byla vytvořena sada testů, kterými musí aplikace úspěšně projít, aby mohla být považována za správně fungující. Testy jsou vytvořeny pro zkontrolování základní funkčnosti celé aplikace – tedy činností, podle nichž můžeme rozhodnout, zda aplikace splňuje zadání či nikoliv. Je doporučeno provádět testy v pořadí, v jakém jsou uvedeny v této kapitole.

7.1.1 Vytvoření databáze

Před začátkem tohoto testu: V případě, že byla aplikace již používána a má nastaveny souřadnice pro zastávky (nebo dokonce obsahuje naměřené pozice vozidel), je nutné tuto používanou databázi nejprve zazálohovat.

Pro testy je doporučeno vytvořit novou databázi, ve které nebudou nastaveny žádné zastávky a směry. V podadresáři `doc` umístěném ve složce s aplikací tedy nejprve smažeme všechny soubory. Poté aplikaci spustíme a provedeme následující činnosti:

1) V menu `Soubor` zvolíme možnost `Vytvořit nový DB model`.
2) Zkontrolujeme, že se ve stavovém řádku aplikace objevila hláška o úspěšném vytvoření modelu: *Nový model databáze úspěšně vytvořen*.
3) Zkontrolujeme, že se ve složce `doc` vytvořily soubory:
   - `publicTransportDB.properties`,
   - `publicTransportDB.script`.

Pokud kontrola proběhla dle očekávání, je tento test úspěšný a může se přejít k testům funkčnosti aplikace, jež jsou popsány v následujících kapitolách.

7.1.2 Aktualizace databáze

Testování aktualizace databáze probíhá následujícím způsobem:

1) V menu `Soubor` zvolíme možnost `Aktualizovat databázi`. Zobrazí se dialog s postupem aktualizace.
2) V průběhu aktualizace provedeme jednu z následujících činností:

a) Stiskneme tlačítko `Cancel` v dialogu s postupem aktualizace a zkontrolujeme, že byla vypsána hláška informující o nekompletní databázi: *Aktualizace databáze zrušena. Data jsou nyní neúplná.* Ukončíme aplikaci a ujistíme se, že při jejím opětovném spuštění dojde k vypsání chybové hlášky: *Databáze není aktualizována. Jízdní řády nejsou kompletní.*
   Pokračujeme bodem 1).
b) Odpojíme počítač, na kterém je spuštěna aplikace, od přístupu k internetu vytažením kabelu nebo vypnutím wi-fi. Zkontrolujeme, že byla vypsána hláška: *Nastala chyba při přístupu k webu PMDP. Data jsou nyní neúplná.* Ukončíme aplikaci a ujistíme se, že při jejím opětovném spuštění dojde k vypsání chybové hlášky: *Databáze není aktualizována. Jízdní řády nejsou kompletní.*
   Pokračujeme bodem 1).
c) Budeme sledovat postup aktualizace bez jakékoli interakce.

3) Zkontrolujeme zobrazení informační hlášky o jejím úspěšném provedení: *Aktualizace databáze úspěšně dokončena.*
4) Zkontrolujeme, že se naplnil seznam zastávek a směrů. Seznam směrů můžeme pro všechny nebo náhodně vybrané porovnat s webem PMDP na záložce *Zastávkové jízdní řády* (viz kapitola 4.3.3).

Pokud se směry s webem PMDP shodují, lze tento test považovat za úspěšně splněný.

7.1.3 Nastavení směru

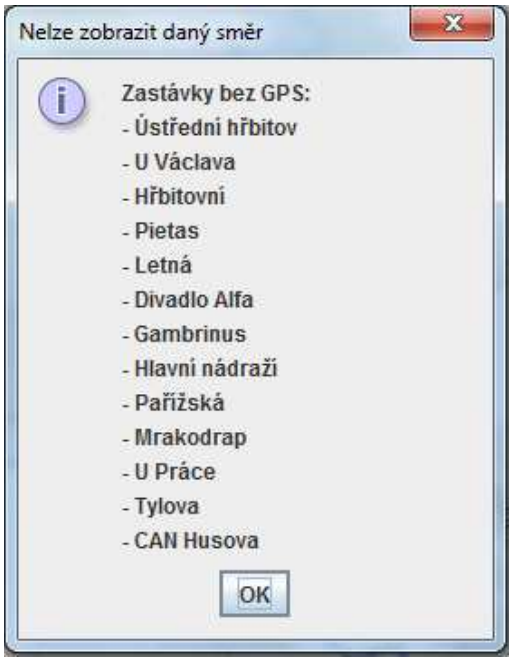
Po aktualizaci databáze otestujeme možnost nastavení souřadnic zastávkám a zobrazení směru. Nejprve otestujeme, že směr, jehož zastávky nemají nastavené souřadnice, nelze zobrazit na mapě:

1) V levém panelu zobrazíme seznam směrů.
2) Dvojklikem zvolíme linku, jejíž směr chceme zobrazit.
3) Dvojklikem zvolíme směr pro zobrazení na mapě.
4) Zkontrolujeme, že se vypsala hláška oznamující chybějící nastavení souřadnic zastávek (viz Obrázek 14 pro linku 11).
5) Nastavíme souřadnice několika (ne všech) zastávkám vypsaným v chybové hlášce v bodě 4) a změny uložíme stisknutím tlačítka `Uložit` v panelu nástrojů.
6) Znovu dvojklikem zvolíme požadovaný směr.

7) Zkontrolujeme, že se vypsala hláška oznamující chybějící nastavení souřadnic zastávek. V seznamu zastávek nejsou námi přidané zastávky v bodě 5) (viz Obrázek 15).

Dále testujeme, že směr s nastavenými zastávkami zobrazit lze:

8) Nastavíme souřadnice zbylým zastávkám směru a změny uložíme stisknutím tlačítka `Uložit` v panelu nástrojů.
9) Znovu dvojklikem zvolíme požadovaný směr.
10) Zkontrolujeme, že se směr správně zobrazil na mapě. Tj. jsou zde všechny zastávky a jsou propojeny spojovacími čarami.



Obrázek 14 – Chybějící nastavení souřadnic 1

Obrázek 15 – Chybějící nastavení souřadnic 2

Pokud kontrola v bodě 10) proběhla úspěšně, můžeme označit test za splněný.

7.1.4 Měření pozic vozidel

Před tímto testem je vhodné restartovat aplikaci, aby v mapě nebyl zobrazen žádný směr. Nyní se pokusíme spustit měření pozic vozidel:

1) Stiskneme tlačítko `Spustit sběr` na panelu nástrojů.
2) Zkontrolujeme, že se vypsala chybová hláška znemožňující měření: *Nebyl vybrán žádný směr. Nelze spustit sběr dat.*
3) Dvojklikem na vybraný směr ho zobrazíme na mapě.
4) Znovu stiskneme tlačítko `Spustit sběr` na panelu nástrojů.
5) V okně s nastavením necháme defaultní hodnoty. Zahájíme sběr tlačítkem *Spustit*.
6) Vyčkáme na dokončení sběru dat a poté si prohlédneme počet nalezených vozidel zapsaný ve stavovém řádku.
7) Zkontrolujeme, že nalezená vozidla jsou zobrazena v mapě na spojovacích čarách mezi zastávkami. A to i při přiblížení a oddálení mapy.

Aplikace prošla testem, pokud byla všechna nalezená vozidla umístěna do mapy na pozice mezi zastávkami.

7.1.5 Export dat

Před tímto testem provedeme postup popsaný v kapitole 7.1.3 alespoň pro dva směry různých linek. Pro tyto směry pak několikrát zopakujeme test uvedený v kapitole 7.1.4. Potvrzení správné funkčnosti exportu provedeme následujícím způsobem:

1) V menu `Soubor` vybereme možnost `Export vozidel do XML`.
2) V dialogu exportu vyzkoušíme jednu z následujících možností:
   a) Export všech linek – v pravém sloupci se seznamem linek vybereme možnost *Všechny linky*.
   b) Export jedné z linek – v pravém sloupci se seznamem linek vybereme jen jednu z linek, pro kterou jsme dříve provedli měření.
3) Zvolíme cestu k souboru, do kterého se mají XML elementy uložit a stiskneme tlačítko *Uložit*.
4) Zkontrolujeme, že soubor existuje a jsou v něm uloženy požadované XML elementy podle výběru možnosti v bodě 2).

Aplikace prošla testem, pokud se vygenerovaly soubory XML dodržující navrženou strukturu (viz příloha B) a s požadovaným obsahem podle volby v bodě 2).

7.1.6 Vyhodnocení testů

Postupně byly provedeny všechny testy popsané v kapitolách 7.1.1–7.1.5. Všechny testy byly vyhodnoceny jako úspěšné (viz Tabulka 1), proto můžeme říci, že aplikace je plně funkční a splňuje zadání práce.

Název testu	Vyhodnocení
Vytvoření databáze	Test proběhl úspěšně.
Aktualizace databáze	Test proběhl úspěšně.
Nastavení směru	Test proběhl úspěšně.
Měření pozic vozidel	Test proběhl úspěšně.
Export dat	Test proběhl úspěšně.

Tabulka 1 – Vyhodnocení testů

7.2 Testování rychlosti hledání vozidel

Po otestování funkčnosti aplikace bylo provedeno měření rychlosti sběru dat pro několik případů. Důležitý je zejména extrémní případ, kdy na trati nebylo žádné vozidlo. K měření byla vybrána linka č. 4, jejíž vozidla jezdí v průběhu dne s krátkými časovými rozestupy.

7.2.1 Testovací sestava

Testování proběhlo na sestavě s následujícími parametry:

- procesor Intel Core i5-3450,
- operační paměť 8 GB RAM,
- operační systém Windows 7 64-bit,
- průměrná rychlost připojení k internetu 22 093 kbit/s.

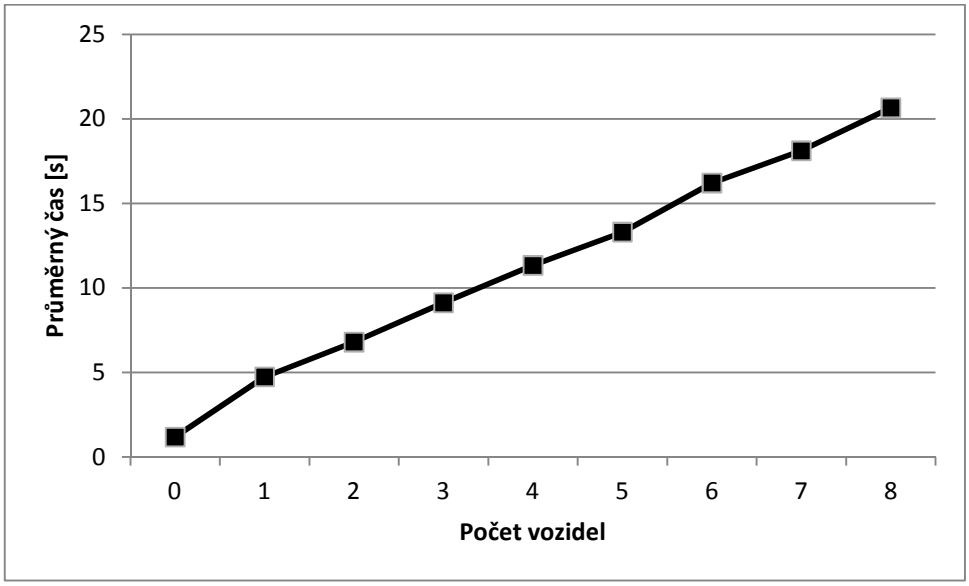
7.2.2 Naměřené hodnoty

Naměřené hodnoty (viz Tabulka 2) ukazují průměrný čas získání pozic vozidel určený vždy z dvaceti měření. Testovací časy jsou uvedeny bez času potřebného pro synchronizaci s webem, protože ten je může být vždy různý.

Počet vozidel	0	1	2	3	4	5	6	7	8
Čas [s]	1,18	4,75	6,80	9,13	11,34	13,30	16,21	18,11	20,65

Tabulka 2 – Testování rychlosti měření

Pokud výsledky vložíme do grafu, můžeme snadno vypozorovat, že časy měření rostou přibližně lineárně v závislosti na počtu nalezených vozidel (viz Obrázek 16).



Obrázek 16 – Průměrné časy měření

Je třeba zdůraznit, že rychlost měření je výrazně ovlivněna čekáním jednu vteřinu před každým přístupem k serveru, proto ani pro vysokorychlostní internet a nejvýkonnější hardware není možno dosáhnout výrazně lepších výsledků.

8 MOŽNÁ ROZŠÍŘENÍ APLIKACE

Přestože aplikace pokrývá všechny požadované funkce, existuje několik vylepšení, které by do ní mohly být v budoucnu zakomponovány a rozšířily by tak její možnosti. V této kapitole se tedy budeme zabývat tím, čím by se dala aplikace vylepšit.

8.1 Java RMI

Pro komunikaci s externími programy bylo v analýze řešení zvoleno použití XML souborů. Existují ale i další možnosti, jak mohou aplikace mezi sebou komunikovat. Rozšířením aplikace by tedy mohlo být např. implementování Java RMI – technologie, která umožňuje komunikaci mezi aplikacemi běžícími na stejném i jiném stroji.

8.2 Konfigurace

V aplikaci je použito několik parametrů, které jsou defaultně nastavené, a uživatel je nemůže nijak ovlivnit. Funkčnosti aplikace to nijak nezabraňuje, ale bylo by vhodné přidat možnost tyto parametry změnit.

8.2.1 Nastavení vyhledávání na webu

Veškerý přístup k webu je prováděn skrze třídu `WebAccess` a lze tedy najít všechny potřebné metody na jednom místě. V případě změny ve struktuře webu by ale bylo nutné tuto třídu a její konstanty změnit a aplikaci znovu přeložit. Bylo by vhodné vytvořit konfigurační soubor, ve kterém by byla popsána struktura např. pomocí XML elementů. V této struktuře by bylo jasně označeno, na jakém místě v HTML dokumentu jsou hledané informace o jízdních řádech a zpoždění vozidel.

Aplikace by mohla tento XML dokument procházet vždy při svém spuštění nebo by v nabídce menu mohla být možnost jednorázové aktualizace hledaných elementů a jejich parametrů. Struktura by se pak mohla vnitřně ukládat do speciální tabulky databáze.

8.2.2 Nastavení objektů mapy

Objekty, které se v mapě zobrazují pro zastávky, průjezdní body, jejich spojovací čáry a dále pro samotná vozidla, jsou také defaultně nastavené. Nastavením se myslí tvary objektů – tj. pro zastávky a průjezdní body jsou zde kruhy a pro vozidla čtverce – a jejich barvy.

Jednou z možností rozšíření aplikace se tedy nabízí vytvořit konfigurační soubor, který by mohl uživatel měnit prostřednictvím grafického rozhraní. Tam by pro každý z objektů mohl vybrat barvu a tvar zobrazení.

8.3 Nápověda

K aplikaci je přidána uživatelská příručka – viz příloha F. Běžným standardem je ale v dnešní době nápověda obsažená přímo v aplikaci. Možností je i vytvoření průvodce aplikací, který by radil uživateli při prvním použití určité funkce.

Dalším rozšířením pro nápovědu k aplikaci může být i video s tutorialem, které by obsahovalo základní možnosti využití aplikace.

8.4 Měření více směrů najednou

V aplikaci uživatel před měřením pozic vozidel vybírá směr linky, pro který chce vozidla hledat. V současné době může uživatel pro jedno měření zvolit jen jeden směr. Bylo by nicméně možné změnit algoritmus vyhledávání, aby prohledával více směrů nebo linek najednou.

Otázkou zůstává, jestli by toto vylepšení přineslo větší užitek. Způsobilo by jistě více přístupů k webovému serveru a jeho zbytečné zatěžování. Protože použití aplikace slouží spíše k zaznamenávání pozic vozidel z dlouhodobého hlediska, mohou se tyto pozice měřit pro každý ze směrů samostatně.

9 ZÁVĚR

Předložená práce splňuje zadání ve všech bodech. Hlavním cílem práce bylo pomocí zpoždění vozidel MHD uvedených na webu PMDP [1] určit co nejpřesnější pozice těchto vozidel a ty dále poskytnout pro externí aplikace. Po prozkoumání webu byly nalezeny všechny informace pro potřeby aplikace a v části s analýzou řešení byly vybrány technologie pro její implementaci.

Bylo rozhodnuto uchovávat informace o jízdních řádech ve vnořené databázi a pro poskytování pozic vozidel externím aplikacím byla vybrána technologie XML. Ta je platformě nezávislá a při správném návrhu snadno čitelná strojově i lidským okem.

Nad rámec zadání byl vytvořen i export struktury jízdních řádů – tedy všech linek s jejich směry a zastávkami. Export vytváří XML dokument s jednoduchou strukturou a může být, stejně jako export pozic vozidel, dále použit v systémech pro simulaci silniční dopravy.

Aplikace také poskytuje uživatelům intuitivní grafické rozhraní s možností zadávat souřadnice zastávkám a vkládat průjezdní body vozidel. Při získávání pozic vozidel je pak zobrazuje přehledně přímo na mapě. Po implementování byla aplikace důkladně otestována.

PŘEHLED ZKRATEK

AWT Abstract Window Toolkit

DOM Document Object Model

GPS Global Positioning System

GUI Graphical User Interface

JAXB Java Architecture for XML Binding

JDK Java Development Kit

JŘ Jízdní řády

MHD Městská hromadná doprava

PMDP Plzeňské městské dopravní podniky

SAX Simple API for XML

StAX Streaming API for XML

UML Unified Modeling Language

XML Extensible Markup Language

XSD XML Schema Definition

SEZNAM ZDROJŮ

[1] *Plzeňské městské dopravní podniky* [online]. 2012 [cit. 2014-04-20]. Dostupné z: http://www.pmdp.cz/.

[2] HAMILTON, John A., POOCH, Udo W. a NASH, David A. *Distributed simulation*. Boca Raton: CRC Press, ©1997. 390 s. CRC Press computer engineering series. ISBN 0-8493-2590-0.

[3] FUJIMOTO, Richard M. *Parallel and distributed simulation systems*. New York: John Wiley & Sons, ©2000. xvii, 300 s. Wiley series on parallel and distributed computing. ISBN 0-471-18383-0.

[4] POTUŽÁK, Tomáš. *Distributed Traffic Simulation* [online], State of the Art and Future Research, Technical Report No. DCSE/TR-2008-02, University of West Bohemia, Plzeň, 2008 [cit. 2014-04-20]. Dostupné z: http://www.kiv.zcu.cz/site/documents/verejne/vyzkum/publikace/technicke-zpravy/2008/tr-2008-02.pdf.

[5] CETIN, N., BURRI, A., NAGEL, K. *A Large-Scale Agent-Based Traffic Microsimulation Based on Queue Model*, Proceedings of 3rd Swiss Transport Research Conference, Monte Verita, 2003.

[6] POTUŽÁK, Tomáš. Current trends in distributed traffic simulation. In: *Modelling and simulation of systems: proceedings of the conference*. Ostrava: MARQ, 2007. s. 34-41. ISBN 978-80-86840-30-7.

[7] NÖKEL, Klaus, SCHMIDT, Matthias. *Parallel DYNEMO: Meso-Scopic Traffic Flow Simulation on Large Networks*, Networks and Spatial Economics, 2002, vol. 2, no. 4, s. 387-403. ISSN 1566-113X.

[8] NAGEL, K., SCHRECKENBERG, M. *A Cellular Automaton Model for Freeway Traffic*, Journal de Physique I [online], 1992, vol. 2, no. 12, s. 2221-2229. ISSN 1155-4304. [cit. 2014-04-20]. Dostupné z doi: 10.1051/jp1:1992277. Dostupné též z: http://fix.bf.jcu.cz/~berec/NagelSchreckenberg1992.pdf.

[9] WAGNER, P., LUBASHEVSKY, L. *Empirical Basis for Car Following Theory Development*. Eprint arXiv:condmat/0311192v1, 2004.

[10] KIESLING, T., LÜTHI, J. Towards Time-Parallel Road Traffic Simulation. In: *Proceedings of the Workshop on Principles of Advanced and Distributed Simulation (PADS 2005)*. Monterey, 2005. s. 7-15. ISBN 0-7695-2383-8.

[11] *IDOS – jízdní řády* [online]. 2012 [cit. 2014-04-21]. Dostupné z: http://jizdnirady.idnes.cz/.

[12] *HyperSQL DataBase* [online]. 2001 [cit. 2014-04-22]. Dostupné z: http://hsqldb.org/.

[13] HEROUT, Pavel. *Java a XML*. 1. vyd. České Budějovice: Kopp, 2007. 313 s. ISBN 978-80-7232-307-4.

[14] *Abstract Window Toolkit* [online]. 1993 [cit. 2014-05-04]. Dostupné z: http://docs.oracle.com/javase/7/docs/technotes/guides/awt/index.html.

[15] *JavaFX* [online]. 2011 [cit. 2014-05-04]. Dostupné z: http://docs.oracle.com/javafx/2/.

[16] *Swing* [online]. 1993 [cit. 2014-05-04]. Dostupné z: http://docs.oracle.com/javase/6/docs/technotes/guides/swing/.

[17] *Jsoup: Java HTML Parser* [online]. 2009 [cit. 2014-04-22]. Dostupné z: http://jsoup.org/.

[18] *Joda-Time - Java date and time API* [online]. 2002 [cit. 2014-04-22]. Dostupné z: http://www.joda.org/joda-time/.

[19] *Google Maps* [online]. 2005 [cit. 2014-04-22]. Dostupné z: https://maps.google.com/.

[20] *Mapy.cz* [online]. 2003 [cit. 2014-04-22]. Dostupné z: http://www.mapy.cz/.

[21] *Open Street Map* [online]. 2004 [cit. 2014-04-25]. Dostupné z: http://www.openstreetmap.org/.

[22] Použití Google Maps. *Google* [online]. 2005 [cit. 2014-04-26]. Dostupné z: http://www.google.com/permissions/geoguidelines.html.

[23] Algoritmus pro výpočet velikonoční neděle. *DZone* [online]. 2005 [cit. 2014-05-04]. Dostupné z:

http://www.dzone.com/snippets/algorithm-calculating-date.

[24] Vzdálenost dvou GPS bodů. *Android Snippets* [online]. 2009 [cit. 2014-04-28]. Dostupné z: http://www.androidsnippets.com/distance-between-two-gps-coordinates-in-meter.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

PŘÍLOHY

A Struktura HTML v Panelu odjezdů

```
<table id="cphMain_SpojeGridView">
  <tr>
    <!-- Buňky hlaviček tabulky -->
  </tr>
  <tr>
    <td class="stationMarker_LineNumberItemStyle">
      <!-- Označení linky -->
    </td>
    <td class="stationMarker_TractionItemStyle">
      <img src="Obrázek: trolejbus/tramvaj.." />
    </td>
    <td class="stationMarker_EndStationNameItemStyle">
      <!-- Název konečné zastávky -->
    </td>
    <td class="stationMarker_DepartureTimeItemStyle">
      <a href="Odkaz na detail spoje">
        <!-- Plánovaný odjezd -->
      </a>
    </td>
    <td class="stationMarker_DelayItemStyle">
      <img src="Obrázek: s/bez zpoždění"/>
      <!-- Počet minut zpoždění -->
    </td>
  </tr>
  ---------
  <tr>
    <!-- Řádka s údaji o dalším vozidle -->
  </tr>
  ---------
</table>
```

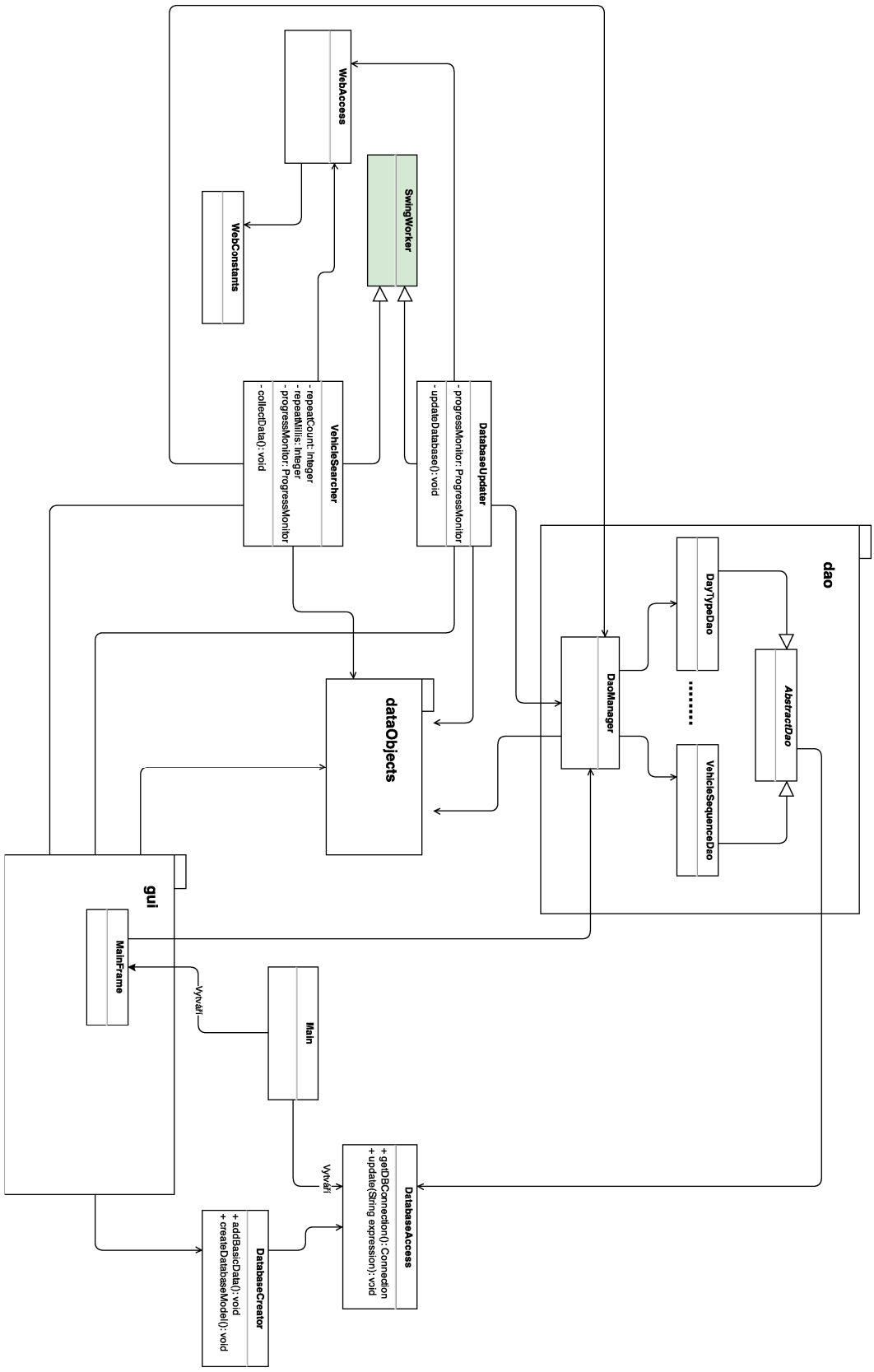
B Struktura XML pro export měření vozidel

```
<vehicleLocations>
  <stops>
    <stop idStop="Číslo (ID)">
      <nameStop>Název zastávky</nameStop>
      <gpsN>GPS N zastávky</gpsN>
      <gpsE>GPS E zastávky</gpsE>
    </stop>
    -------
  </stops>
  <measurings>
    <measuring date="Datum sběru" time="Čas sběru">
      <vehicle stopStartDeparture="Čas výjezdu vozidla">
        <line>Značení linky</line>
        <idStopStart>ID počáteční zastávky</idStopStart>
        <idStopTarget>ID cílové zastávky</idStopTarget>
        <idStopNext>ID následující zastávky</idStopNext>
        <gpsN>GPS N vozidla</gpsN>
        <gpsE>GPS E vozidla</gpsE>
        <delay>Zpoždění v minutách</delay>
      </vehicle>
      -------
    </measuring>
    -------
  </measurings>
</vehicleLocations>
```

C Struktura XML pro export linek

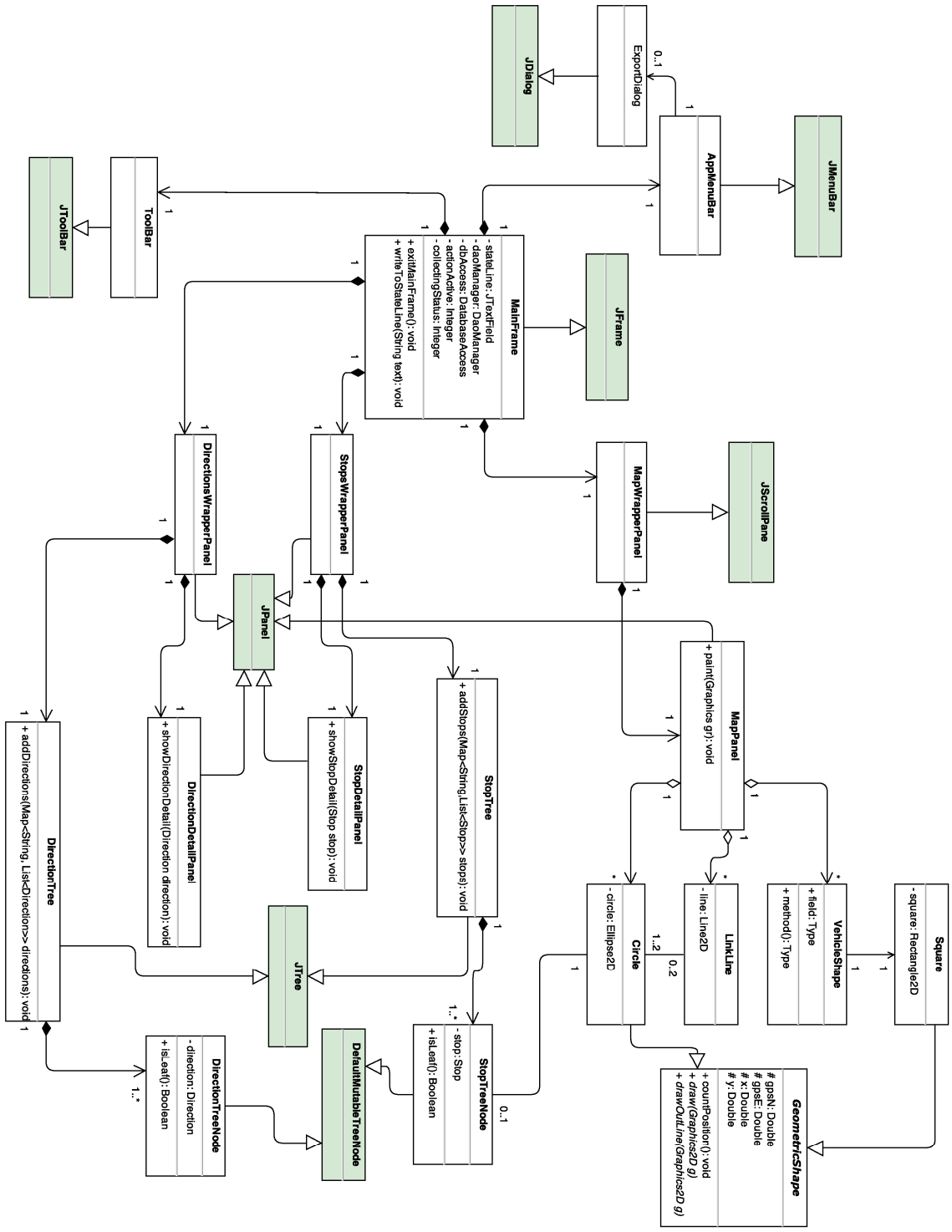
```
<publicTransport>
  <stops>
    <stop idStop="Číslo (ID)">
      <nameStop>Název zastávky</nameStop>
      <gpsN>GPS N zastávky</gpsN>
      <gpsE>GPS E zastávky</gpsE>
    </stop>
    -------
  </stops>
  <lines>
    <line marking="Označení linky">
      <direction sign="Označení směru">
        <stopSequence idStop="ID první zastávky" />
        <stopSequence idStop="ID druhé zastávky" />
        -------
      </direction>
      -------
    </line>
    -------
  </lines>
</publicTransport>
```

D Zjednodušený diagram tříd aplikace



Obrázek přílohy I – Zjednodušený diagram tříd

E Zjednodušený diagram tříd pro GUI



Obrázek přílohy II – Diagram tříd pro GUI

F Uživatelská příručka

F.1 O aplikaci

Aplikace vznikla jako diplomová práce za účelem získávání informací o pozicích vozidel městské hromadné dopravy na území města Plzně.

F.2 Instalace a spuštění

Pro chod aplikace je nutné mít v počítači nainstalované prostředí Java SE ve verzi 1.7 nebo vyšší[1]. Samotná aplikace se neinstaluje, jen se zkopíruje se všemi potřebnými soubory a složkami do příslušného adresáře. Pro spuštění aplikace je nutné mít v adresáři soubory a složky:

- soubor `VehicleSearcher.jar,`
- složku `images` se všemi jejími soubory.

Spuštění:

- dvojím kliknutím na soubor `VehicleSearcher.jar` (v případě asociace přípony `.jar` s nainstalovaným prostředím Java) nebo
- příkazem v konzoli `java -jar VehicleSearcher.jar`.

Pozn.: Pro zajištění správné funkčnosti aplikace je nutné připojení k internetu.

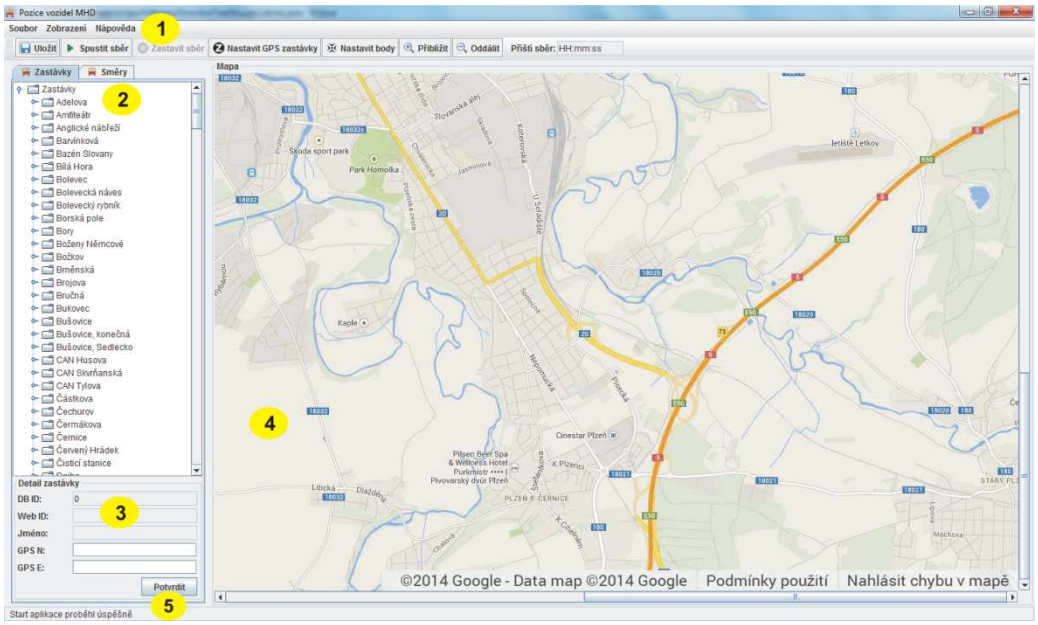
F.3 Hlavní okno aplikace

1) Menu a panel nástrojů

Na obrázku III je hlavní okno aplikace s několika označenými částmi. Část **1** je klasické menu aplikace sloužící k základním úkonům. K menu je připojen *Panel nástrojů*, který obsahuje (postupně zleva doprava) nástroje pro:

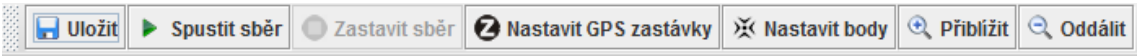
- ukládání provedených změn,
- spuštění sběru dat,
- zastavení sběru dat,
- nastavení GPS souřadnic zastávkám,
- nastavení průjezdních bodů mezi zastávkami jednoho směru,
- přiblížení mapy,
- oddálení mapy.

[1] Instalace Java: http://www.oracle.com/technetwork/java/javase/downloads/index.html



Obrázek přílohy III – Hlavní okno aplikace

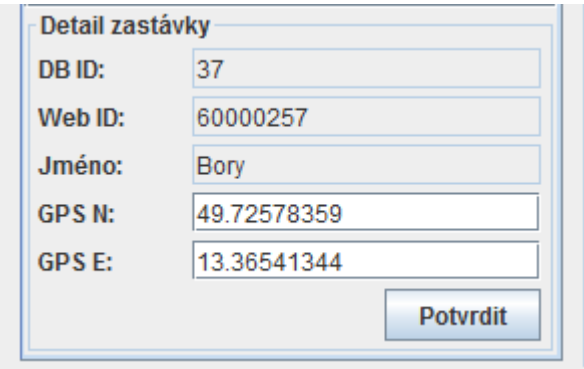
Detail panelu s nástroji je na obrázku IV.



Obrázek přílohy IV – Panel nástrojů

2) Seznamy zastávek a směrů

Část okna označena na obrázku III číslem **2** slouží k zobrazení seznamu zastávek a směrů uložených v databázi aplikace. Oba seznamy mají stromovou strukturu. V seznamu se zastávkami jsou jednotlivé zastávky shromažďovány podle svého názvu. Pro jeden název může v aplikace existovat více zastávek s různými souřadnicemi. Po kliknutí na jednu konkrétní zastávku je v části označené číslem **3** zobrazen její detail (viz obrázek V). Zde je možné ručně upravovat zastávkám GPS souřadnice a následně je stisknutím tlačítka *Potvrdit* zanést do aplikace.



Obrázek přílohy V – Detail zastávky

Po přepnutí na kartu *Směry* (nad výpisem zastávek) se zobrazí seznam směrů. Ty jsou shromažďovány podle označení jejich linek. Také pro směry existuje panel pro zobrazení jejich detailu, nelze u nich ale nic editovat. Zastávky i směry jsou v seznamu řazeny podle české abecedy.

3) Mapa

V části označené číslem **4** na obrázku III je mapa Plzně a jejího okolí. Mapu lze přibližovat a oddalovat pomocí tlačítek *Přiblížit* a *Oddálit* umístěných v *Panelu nástrojů* nebo v položce menu *Zobrazení*. Aplikace umožňuje zobrazit dvě úrovně detailu mapy.

4) Stavový řádek

V dolní části okna na obrázku III je označen číslem **5** tzv. *Stavový řádek.* Zde se zobrazují informace, které jsou spíše informativního charakteru a nevyžadují další interakci. Např. je zde zobrazena informace o úspěšném uložení dat při stisknutí tlačítka *Uložit*.

F.4 Vytvoření nové databáze

Kliknutím na položku menu `Soubor -> Vytvořit nový DB model` se do složky `db` v adresáři s aplikací vygeneruje nová databáze.

Pozn.: V případě, že je v této složce již databáze umístěna, je automaticky přepsána nově vytvořenými soubory a dojde tak ke ztrátě uložených dat. Pro jejich uchování je proto nutné před vygenerováním nové databáze složku `db` zazálohovat.

F.5 Aktualizace databáze

Na webu PMDP dochází několikrát do roka ke změnám v jízdních řádech. Při změně na webu je pro správnou funkčnost nutné aktualizovat databázi aplikace. To se provede vybráním položky menu `Soubor -> Aktualizovat databázi`. Aktualizace trvá typicky několik desítek minut, proto je její průběh zobrazen ve speciálním dialogu. Během aktualizace nelze s aplikací pracovat.

Aktualizaci je možné zrušit kliknutím na tlačítko *Cancel* v dialogu s jejím průběhem. Tím ale nebude databáze naplněna všemi údaji, proto je v tomto případě nutné spustit aktualizace v nejbližší době znovu od začátku.

F.6 Nastavení souřadnic zastávkám

Pro nastavení GPS souřadnic zastávkám stiskneme tlačítko *Nastavit GPS zastávky* v *Panelu nástrojů*. Postup pro nastavení souřadnic jedné ze zastávek je následující:

1. Vybereme kliknutím myši konkrétní zastávku – tj. zastávku se zobrazenými souřadnicemi – ze seznamu (část **2** na obrázku III).
2. Klikneme do mapy na požadovanou pozici. Při správném postupu se na mapě zobrazí červený kruh.
3. V případě, že zastávku chceme posunout na jinou pozici, lze toto provést stisknutím myši na červeném kruhu zastávky a přetažením na požadované místo.

Postup je možný opakovat pro více zastávek. Po ukončení činnosti vypneme aktivitu nastavování pozic opětovným stisknutím tlačítka *Nastavit GPS zastávky* a poté stiskneme tlačítko *Uložit*, aby se zapsaly nové pozice do databáze.

Při opětovném spuštění aplikace lze zastávky zobrazit v mapě dvojím kliknutím na danou zastávku v jejich seznamu.

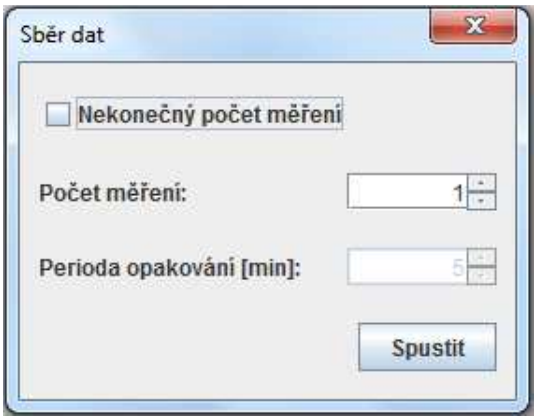
F.7 Měření pozic vozidel

Před spuštěním měření pozic vozidel je nutné v mapě zobrazit směr, pro který chceme vozidla vyhledávat. Zobrazení směru na mapě dosáhneme dvojím kliknutím na požadovaný směr v seznamu směrů. Aby se směr skutečně zobrazil na mapě, je nutné mít nastaveny souřadnice pro všechny zastávky směru.

Měření spustíme stisknutím tlačítka *Spustit sběr* na *Panelu nástrojů*. Zobrazí se okno s nastavením měření (viz obrázek VI). Zde je možné nastavit, kolikrát se budou pozice vozidel měřit a s jakým časovým intervalem. Zaškrtnutím možnosti *Nekonečný počet měření* dojde k neustálému opakování měření až do zastavení uživatelem nebo do ukončení aplikace. Po nastavení měření zahájíme činnost stisknutím tlačítka *Spustit*.

Zobrazí se dialog ukazující postup při měření pozic vozidel. Před začátkem samotného měření dochází k synchronizaci času mezi počítačem a webovým serverem, proto je nutné několik (desítek) sekund vyčkat. Během čekání při synchronizaci i během samotného měření lze sběr dat přerušit stisknutím tlačítka *Cancel*.

Ve chvíli, kdy měření úspěšně doběhlo do konce, jsou nalezená vozidla zobrazena do mapy. Vozidla jsou v mapě vykreslena jako modré čtverce s vepsaným názvem jejich linky. S vozidly na mapě nelze hýbat jako se zastávkami posouváním myši.

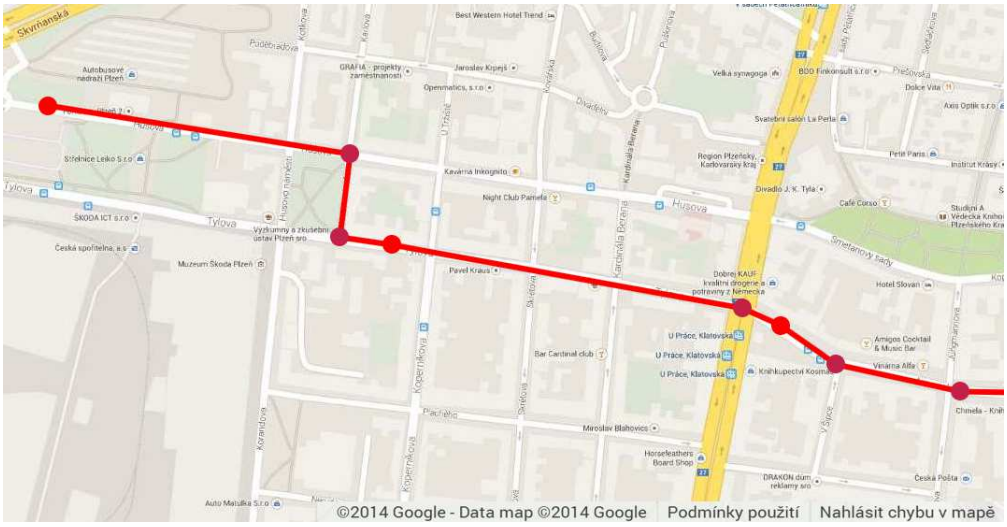


Obrázek přílohy VI – Nastavení sběru dat

Po ukončení jednoho měření se v případě nastaveného opakování zapíše čas začátku dalšího měření vedle *Panelu nástrojů* do pole označeného *Příští sběr*. Toto měření se automaticky spustí v nastavený čas. V mezičase mezi dvěma sběry dat lze další měření ukončit stisknutím tlačítka *Zastavit sběr* umístěném v *Panelu nástrojů*.

F.8 Průjezdní body

Zobrazenému směru na mapě lze upravovat trasu přidáváním průjezdních bodů mezi zastávky (viz obrázek VII). K tomuto účelu slouží tlačítko *Nastavit body* umístěné v *Panelu nástrojů*. Po stisknutí tlačítka lze kliknutím na spojovací čáru mezi dvěma zastávkami přidat průjezdní bod. Stejně jako lze přesouvat po mapě zastávky, lze tažením myši přesunout i průjezdní bod na požadované místo.



Obrázek přílohy VII – Část směru zobrazená na mapě

Průjezdní bod lze ze směru odebrat kliknutím pravého tlačítka myši. Po ukončení činnosti vypneme aktivitu opětovným stisknutím tlačítka *Nastavit body* a poté *Uložit*, aby se zapsaly nové body do databáze.

F.9 Přepojení zastávky

Pro jeden název zastávky může existovat více konkrétních zastávek s různými souřadnicemi. Zastávky, které jsou zobrazeny na mapě lze proto rozdělit na dvě. Rozdělení provedeme kliknutím pravého tlačítka na zastávku a vybráním možnosti *Rozdělit zastávku*. Tím vznikne nová zastávka (v tuto chvíli překrývající původní), se kterou můžeme pohybovat pomocí myši a přesunout ji na požadované místo. Nyní stiskneme tlačítko *Uložit* pro zapsání změn.

Aby se nově vytvořená zastávka mohla přidat do směru jako průjezdní, musí být daný směr zobrazený na mapě. Poté klikneme na zastávku, která se má nahradit, pravým tlačítkem myši a vybereme možnost *Změnit na jinou zastávku pro daný směr*. Pak kliknutím myši vybereme nově přidanou zastávku.

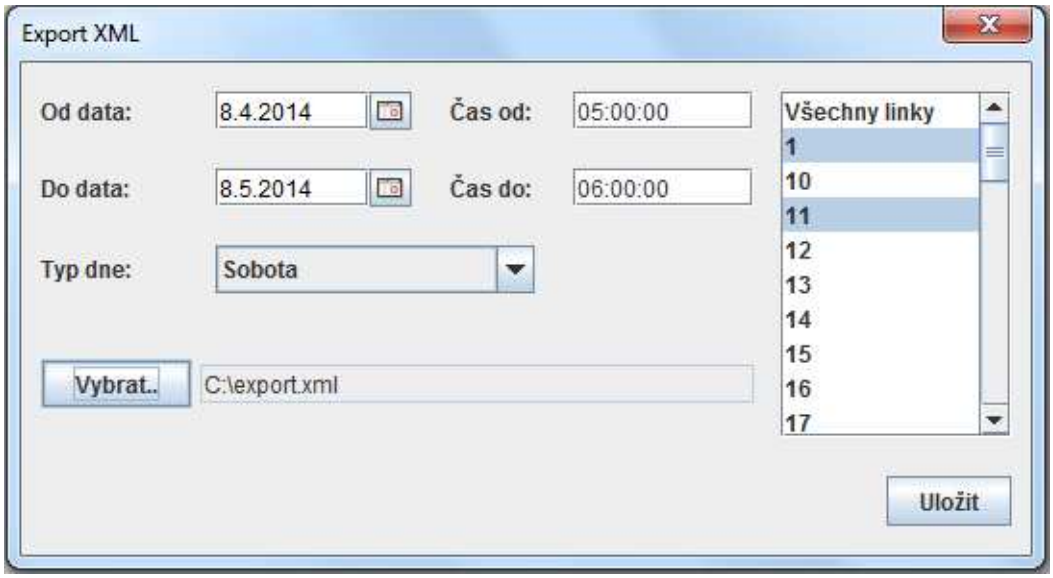
F.10 Export dat

Pokud je databáze naplněna daty, lze je exportovat do XML souborů. Aplikace umožňuje export vyhledaných vozidel (popis XML viz příloha B) a linek MHD (popis XML viz příloha C).

Export vozidel

Kliknutím na položku menu `Soubor -> Export vozidel do XML` se otevře okno s možnostmi nastavení, jaká data se budou exportovat (viz obrázek VIII). Zde je možné nastavit následující seznam parametrů:

- *Od data* – určuje dolní hranici data exportovaných měření. Při ponechání prázdného pole nebude nastavena žádná dolní hranice.
- *Do data* – určuje horní hranici data exportovaných měření.
- *Čas od* – určuje dolní hranici času exportovaných měření.
- *Čas do* – určuje horní hranici času exportovaných měření.
- *Typ dne* – výběrem jedné z položek *Pracovní dny*, *Sobota*, *Neděle a svátky* se exportovaná měření omezí na požadovaný typ dne. V případě ponechání přednastavené možnosti *Všechny dny* nebude brán při exportu ohled na typ dne, kdy se měření konalo.
- *Název linky* – Z pravého sloupce lze vybrat názvy linek, pro které se má export provést. Při zvolení možnosti *Všechny linky* budou vybrána měření všech uložených linek.



Obrázek přílohy VIII – Nastavení exportu

Parametry lze kombinovat, proto při nastavení exportu uvedeném na obrázku VIII lze vygenerovat např. XML dokument pro vozidla linky č. 1 a 11 jezdící v sobotu. Navíc budou exportována jen ta měření, která jsou uložena v termínu 8. 4. 2014 – 8. 5. 2014 a to jen v čase vždy od pěti do šesti hodin ráno.

Po zvolení výstupního souboru kliknutím na tlačítko *Vybrat* a nalezením cesty k souboru lze požadovaný dokument exportovat tlačítkem *Uložit*. Exportovaný XML dokument z obrázku VIII bude uložen do souboru `C:\export.xml`.

Export linek

Kliknutím na položku menu `Soubor -> Export linek do XML` se do otevře dialog pro vybrání názvu a cesty k souboru. Po potvrzení dialogu se vygeneruje soubor se všemi linkami MHD.