UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Informační systém přepravní společnosti

Bc. Zdeněk Špulák

Diplomová práce
2013

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2012/2013**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Zdeněk Špulák**
Osobní číslo:	**I10422**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Informační systém přepravní společnosti**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

Cílem této práce je vytvoření funkční aplikace s optimalizovanými SQL dotazy v databázovém systému, která bude obsahovat nástroje sloužící pro evidenci a logistické řízení přepravní společnosti.
Aplikace bude umožňovat tyto funkcionality:
1. Registraci uživatelů systému.
2. Přístup uživatelů do systému podle jejich práv.
3. Příjímání a zpracování objednávek zákazníků.
4. Evidenci automobilů a řidičů.
5. Vytváření rozpisů denních jízd pro jednotlivá auta a řidiče.
6. Generování sestav dle zadaných kritérií pro potřeby vedoucích pracovníků.
7. Řízení údržby vozového parku.
8. Logistické řešení nakládky zboží a přepravy mezi jednotlivými lokalitami z pohledu minimalizace nákladů.

V úvodní části je nutno provést rešerši stávajících SW řešení pro řízení přepravní společnosti. Rešerši je nutné doplnit o porovnání s nově navrhovaným systémem, který bude předmětem této práce. Úvodní část musí obsahovat analýzu navrhovaného řešení, která bude obsahovat popis použitých technologií, návrh databáze a aplikačního řešení. Pro vytvoření aplikace bude využit skriptovací jazyk PHP nebo JAVA a databáze Oracle nebo MySQL. V práci bude navržen postup pro provedení optimalizace vybraných SQL dotazů s následnou analýzou a porovnáním výkonu nalezených alternativ exekučních plánů s původním dotazem.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

1. LACKO, Luboslav. Oracle. Správa, programování a použití databázového systému. Brno: Computer Press, 2007.
2. GROFF, James R. a WEINBERG, Paul N. SQL kompletní průvodce. Brno: Computer Press, 2005.
3. VRÁNA, Jakub. 1001 tipů a triků pro PHP. Brno: Computer Press a.s., 2010.
4. BRYLA, Bob; LONEY, Kevin. Mistrovství v Oracle Database 11g. Jiří Huf. Vydání první. Brno: Computer Press, a.s., 2009.
5. NOWICKI, Steven D. a LECKY-THOMSON, Ed. PHP 6. Programujeme profesionálně. Brno: Computer Press, 2009.
6. DRUSKA, P. CSS a XHTML - tvorba dokonalých webových stránek krok za krokem, Grada 2006.
7. LONEY, K., THERIAULT, M. Mistrovství v Oracle. Praha, Computer Press, 2002.
8. www.oralce.com

Vedoucí diplomové práce: **Ing. Miloslav Macháček, Ph.D.**
Katedra informačních technologií

Datum zadání diplomové práce: **31. října 2012**
Termín odevzdání diplomové práce: **17. května 2013**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

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

V Pardubicích dne 15. listopadu 2012

Prohlášení autora

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

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

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 20. 8. 2013 Bc. Zdeněk Špulák

Anotace

Tato práce se zabývá tvorbou informačního systému pro přepravní společnost. Analyzuje a řeší problémy, které mohou při tvorbě takového softwaru nastat. Součástí práce je vlastní implementace IS pro přepravní společnost, včetně detailního popisu použitých technologií a komponent.

Klíčová slova

informační systém, přepravní společnost, problém obchodního cestujícího, PHP, MySQL databáze, Nette Framework

Title

Information System for the Transport Company

Annotation

This work talks about information system for the transport company. It analyzes and solves problems, which can occur during development. Part of this work is also own implementation of IS for the transport company including detailed description of used technology and components.

Keywords

Information system, Transport Company, The travelling salesman problem, PHP, MySQL database, Nette Framework

Obsah

Seznam zkratek

IS	Informační systém
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CSS	Cascading Style Sheets
HTML	HyperText Markup Language
OS	Operační systém
PHP	Hypertext Preprocessor
SEO	Search engine optimization
SQL	Structured Query Language
XML	Extensible Markup Language
IE	Internet Explorer
VLSI	Very-large-scale integration
URL	Uniform Resource Locator
DOM	Document Object Model
JSON	JavaScript Object Notation
HTTP	Hypertext Transfer Protocol
SVG	Scalable Vector Graphics
3NF	Třetí normální forma
WGS84	World Geodetic System 1984
GPX	GPS eXchange Format
POI	Point of interest
VML	Vector Markup Language

Seznam obrázků

Seznam tabulek

Úvod

V dnešní době existuje mnoho firem zabývající se přepravou různého zboží a zásilek. Vezmeme-li v úvahu obrovské nadnárodní společnosti nebo jen o menší firmy, všichni potřebují nějakým způsobem řídit logistiku – rezervaci objednávek, rozvážku zboží apod. Moderní doba s sebou neodmyslitelně nese moderní nástroje, proto je dnes veškerá logistika řízena elektronicky pomocí vhodného softwaru. Tato práce se zabývá právě problémem řízení logistiky přepravní společnosti, resp. tvorbou softwarového díla pro takovou společnost.

V prvotní fázi je rozebíráno obecné hledisko softwaru – co by měl požadovaný program umožňovat, jaké funkcionality by měl nabízet, a v neposlední řadě jsou popsány vybrané softwarové balíky dostupné na trhu. Teoretický rozbor pokračuje i v dalších kapitolách, kde jsou popisovány algoritmy pro vyhledání nejkratší cesty – rozebrány jsou i algoritmy nevhodné pro tuto konkrétní aplikaci. Porovnávány jsou různí mapoví poskytovatelé dat pro vykreslování trasy, a to včetně popisu dostupných API. Zmíněn je také problém s nakládkou zboží do kontejneru. Poslední kapitolou teoretické části je nevyhnutelná optimalizace SQL dotazů, která nesmí v každé kvalitní databázové aplikaci chybět.

Praktická část je rozdělena do mnoha podkapitol. Nejprve jsou popisovány použité technologie, frameworky a doplňky, na kterých aplikace stojí. Následuje popis architektury vytvořené aplikace – databázový návrh, adresářová struktura a detailní popis jednotlivých modulů. Pro představu kompletní funkčnosti je k dispozici také UseCase diagram, podobně je tomu u UML diagramu, který napoví o vnitřní architektuře vyvinutého softwaru. Následují kapitoly, které popisují předpokládaný scénář použití aplikace a seznamují s hlavním algoritmem aplikace, což je výpočet rozvážených objednávek pro jednotlivá vozidla spolu s omezením rozměrů a hmotnosti. V závěru praktické části je popsán vizuální styl aplikace a je zmíněn popis formátu souborů pro navigační zařízení, které dokáže aplikace generovat. Úplně na konci praktické části je uvedena přehledová tabulka s licencemi použitých komponent a doplňků třetích stran.

V závěru práce je shrnuto, jak celý vývoj probíhal a zda byly všechny cíle zadání splněny. Uvedeno je i několik možných rozšíření vytvořené aplikace. Součástí publikace jsou také přílohy, které přibližují vzhled vyvinuté aplikace pomocí screenshotů.

1 Informační systém pro přepravní společnost

Obecný termín „informační systém“ je dnes již zřejmě všem známý – je to jakýsi soubor technologických prostředků a metody, které zabezpečují sběr, přenos a uchovávání dat. A to primárně za účelem prezentace informací pro potřeby uživatelů informačního systému. Pojem informační systém bývá mnohdy zkracován pouze na iniciály IS a v dnešní počítačové době jej většina z nás vnímá především jako nějaké softwarové dílo.

Z výše uvedené definice nám vyplyne, že informační systém pro přepravní společnost je jedna konkrétní implementace IS, právě pro potřeby přepravní společnosti. Obecná definice takového softwaru je poměrně problematická, neboť pod tímto názvem můžeme nalézt nepřeberné množství vlastností a funkcionalit. Pokusme se tedy alespoň definovat základní moduly, které by v každém informačním systému pro přepravní společnost neměly chybět.

- Správa vozového parku – možnost definovat všechny dostupné automobily, kterými společnost disponuje. Umožnit konkrétní vozidla rušit, přidávat a obecně s nimi v rámci aplikace pracovat (automobil je nedostupný kvůli technickým problémům apod.).

- Správa lidských zdrojů – myšleno především řidiče automobilů. Nutnost mít přehled o všech zaměstnancích (řidičích) a pro každý automobil definovat příslušného řidiče. Pokud daná společnost vyžaduje nakládku zboží v depu apod., nutnost řídit i pomocné díly (nakladače zboží apod.).

- Logistické řízení objednávek – umožnit do systému vložit novou objednávku, mít možnost definovat den doručení, adresáta, cenu atd. Samozřejmě podpora pro případné storno objednávky, případně pro možné změny.

- Výstupy pro rozvážku zboží – pravidelné (denní) generování rozvážky objednávek. Na základě definovaných automobilů, řidičů a objednávek vytvořit rozpis pro rozvážku.

- Optimalizace trasy – každému automobilu předem určit danou trasu, kterou pojede. Výhodné pro minimalizaci nákladů za najeté kilometry.

- Přehledy a statistiky – modul pro potřeby managementu společnosti. Umožnit přehledné zobrazování rozvážek, vytíženosti automobilů a řidičů, přehledy o najetých kilometrech apod. Ideálně vyobrazit v podobě přehledných grafů a tabulek.

- Uživatelské role a oprávnění – možnost definovat přístup k různým funkcionalitám pro různé osoby. Management musí vidět odlišné funkce než třeba dispečer vkládající objednávky od zákazníků.

- Model klient-server – využití především pro potřebu používat aplikaci více uživateli najednou (na více zařízeních).

- Zabezpečení – obecný požadavek pro každou aplikaci. Rozhodně není žádoucí, aby nepovolaná osoba viděla informace, která vidět nemá. Dále minimálně ochrana heslem pro každý uživatelský účet.

1.1 Dostupná řešení na trhu

Na trhu dnes existuje poměrně hodně (řádově desítky) produktů, které se zabývají problematikou řízení přepravní společnosti. V drtivé většině se však jedná o veřejně nedostupné aplikace, které jsou každé společnosti nasazovány a upravovány na míru a dle jejich potřeb. Běžný člověk se s takovýmito produkty má pouze malou šanci reálně setkat. Z tohoto důvodu je poměrně složité provést srovnání existujících řešení. V následujících podkapitolách bude uvedeno několik vybraných produktů a ty budou popsány. Nicméně informace o produktech jsou vždy z oficiálních stránek výrobců a nejsou detailně ověřeny a odzkoušeny. Je proto možné, že informace mohou být do jisté míry zkreslené a neúplné.

1.1.1 Logio – Distribuční strategie a optimalizace dopravy

Roadnet Transportation Suite je komplexní nástroj pro efektivní řízení dopravy. Tento soubor aplikací zahrnuje strategické a taktické optimalizační nástroje v dopravě, využívající běžné uživatelské prostředí MS Windows. Software obsahuje následující moduly:

- Roadnet Transportation Suite – používán k optimalizaci a vyvážení doručování a zlepšení zákaznického servisu tím, že navrhuje efektivní teritoria a trasy. Jedná se o systém zastřešující následující volitelné moduly. Umístění skladů, návrh rozvozových teritorií, tras a teritorií pro obchodní zástupce.

- Territory Planner – strategický plánovací nástroj, který analyzuje historické dopravní informace a na jejich základě navrhuje teritoria a vytěžuje trasy.

- Roadnet – taktický nástroj pro denní trasování, který generuje nejlevnější trasu při dodržení požadované úrovně zákaznického servisu.

- Fleetloader – aplikace určená pro řízení expedice ze skladu a nakládky zboží na vozidla zejména v nápojovém průmyslu. Online sledování, dynamické řízení, oboustranná komunikace, maximální kontrola vozidel.

- MobileCast – poskytuje aktuální polohu a výkon na trase. To umožňuje operativně řešit případné problémy, potvrzovat čas přesného doručení a sledovat zásilky (tracking).

- Roadnet Info Center – soubor webových nástrojů provozovaných na Intranetu firmy, který zobrazuje plány tras a/nebo reálně zpracovává data.

1.1.2 Logicon – Software pro optimalizaci dopravy

Software od společnosti Logicon je rozdělen na tři moduly.

- PLANTOUR – umožňuje na základě každodenního zpracování objednávek navrhovat optimální trasy pro závoz dodacích míst včetně zohlednění zpětných svozů. Trasy jsou navrhovány na základě aktuálních objednávek a vozového parku dynamicky tak, aby byly optimální z hlediska nákladů a zároveň splňovaly všechny zadané restrikce (doba závozu, limit vytížení vozidla, požadavky na vybavenost vozidla aj.). PLANTOUR přináší možnost plné každodenní kontroly nad náklady na distribuci až na úroveň dodacího místa s možností jejich dalšího snižování pomocí optimalizace tras. Přímých úspor je možno dosáhnout redukcí tras, nákladů, počtu vozidel, ujetých kilometrů. Úspora přímých dopravních nákladů 10 – 30 % je ověřena desítkami implementací systému v ČR i zahraničí. Navržené trasy jsou prezentovány několika způsoby. Jedná se o tiskové sestavy, dispečerský přehled, itineráře pro řidiče. Veškeré informace o trasách se zobrazují v přehledném datovém seznamu, mapovém okně a na časové ose. Využitím analytických možností produktu je možno definovat potřeby vozového parku, optimální rozložení dep, přiřazení dodacích míst depům apod.

- CARMANAGER – kontrolní a evidenční systém pro správu a řízení vozového parku, údajů o trasách, řidičích a odběrných místech s možností provádění detailní evidence a správy dat. Data je možno zpracovávat ve formě řady rozborů a vyhodnocovacích analýz. Systém je plně integrován s plánovacím systémem PLANTOUR. Jeho součástí je specializovaný modul pro automatické srovnávání plánovaných a skutečných tras. Integrované funkce pro analýzy plánovaných a skutečných nákladů tak umožňují zhodnotit efektivitu všech zdrojů zapojených do přepravy zboží. Odchylkové a srovnávací analýzy plánu a skutečnosti slouží k účinné identifikaci slabých míst naplánovaných tras a nabízejí možnosti korekce vstupů pro zohlednění skutečné situace při sestavování plánů.

- TRACKMANAGER – naplánované trasy nejsou statické, při jejich realizaci dochází k neustálým změnám na základě dynamické dopravní situace. Propojením plánování s on-line sledováním pohybu vozidel na trasách umožňuje operativní zohlednění naplánovaných tras. Dispečer má možnost na základě informace o exaktní poloze vozidla operativně aktualizovat průběh trasy a včas reagovat na odchylky od plánu. Systém TRACKMANAGER je sofistikované řešení pro online sledování pohybu vozidel na trasách, sledování a analýzy telemetrických údajů o vozidlech (spotřeba PHM, otevíraní dveří, teplotní údaje, apod.). Řešení poskytuje rovněž přehledné analytické možnosti srovnávání odchylek plánovaných a skutečných tras.

1.1.3 KSH-Data – Doprava 3K

Program DOPRAVA 3K je aplikace typu klient-server s použitím moderních technologií pro vývoj aplikačního software a představuje novou generaci software pro dopravce a

speditéry. Základem jsou mnohaleté zkušenosti pracovníků firmy s vývojem programů v oblasti autodopravy a spedice a dále připomínky a zkušenosti několika stovek uživatelů programu DOPRAVA 2000.

Aplikace je vyvinuta ve vývojovém prostředí DELPHI na straně klient a server je postaven na relační databázi FireBird. Systém pracuje pod operačními systémy Windows 2000, NT, XP, Windows 7 jak na straně server, tak na straně klient. Databázový server může pracovat také pod operačním systémem Linux. Design a uspořádání formulářů (oken) systému je řešeno tak, aby se co nejvíce blížilo standardům (např. Microsoft Office). Uživatel se tak snadno v programu orientuje a ovládá ho intuitivně bez podrobného studia příručky.

Program DOPRAVA 3K je vybaven pro komunikaci s okolím. Prakticky libovolné informace lze ze systému exportovat v požadovaném formátu a dále zpracovat v jiném programu (účetnictví, tabulkový procesor, elektronický diář apod.). Systém je dále otevřen pro import dat zvenčí (import zakázek ze systémů nabídek přeprav, podkladů pro záznamy o provozu vozidel z GPS systémů a z palubních počítačů, import dat z Internetu), umožňuje komunikaci pracovních stanic prostřednictvím Internetu apod. Základní komunikaci shrňme v těchto bodech:

- systémy sledování vozidel,
- účetní programy,
- banka,
- platební karty,
- čerpací stanice,
- databanka přeprav,
- internet,
- objednávkový systém on-line.

1.1.4 Ostatní produkty

Mezi dalšími společnostmi nabízející produkty pro plánování softwaru můžeme jmenovat následující:

- Paragon – Strategic Transport and Logistics Planning
- Ortec – Routing and Scheduling, Pallet and Load Building
- TMW Systems – Schedule Pro, DirectRoute

Vlastní software pro logistické řízení přepravy nabízí i softwaroví giganti jako jsou IBM a jejich produkt Sterling Management System. Dalším velikánem je také společnost SAP, která vyvíjí produkt SAP Transportation Management (SAP TM). Je však třeba podotknout, že tyto systémy jsou již opravdu velice sofistikované a jejich primární využití je pro obrovské nadnárodní společnosti.

2 Algoritmy pro výpočet nejkratší cesty

Výpočet nejkratší trasy je v této práci jeden ze stěžejních úkolů. Pokud používaná aplikace obsahuje kvalitní algoritmus pro výpočet nejkratší cesty, lze denně ušetřit nemalé prostředky – a to jak časové, tak i finanční (nafta, více rozvezených objednávek apod.).

2.1 Nevhodné algoritmy

V rámci studia se autor práce setkal s několika algoritmy pro výpočet optimální trasy. Nicméně všechny níže popsané (Dijkstrův, Bellmanův-Fordův, Floydův-Warshallův, A*) jsou nevhodné pro tento typ aplikace, a to ze stejného důvodu – počítají optimální trasu pouze mezi dvěma body.

2.1.1 Dijkstrův algoritmus

Dijkstrův algoritmus je grafový algoritmus sloužící k vyhledání nejkratší cesty z počátečního uzlu do všech ostatních uzlů ohodnoceného grafu. Funguje však pouze jen s nezáporným ohodnocením hran. Algoritmus poprvé popsal nizozemský informatik Edsger Dijkstra, po kterém je pojmenován. Často se používá v routovacích protokolech, například v algoritmu OSPF (Open Shortest Path First).

Vstupem do algoritmu je ohodnocený graf `G` a počáteční uzel `s`. Výstupem je asociativní pole `D(u)`, udávající nejkratší vzdálenost mezi uzlem `s` a uzlem `u`. Algoritmus si pro každý uzel `u` uchovává informaci o nejkratší cestě, kterou se k němu lze dostat. V prvotní fázi (inicializace) se všechny délky nastaví na maximální hodnotu (nekonečno) – vyjma počátečního uzlu, ten délku rovnou nule. Dále je uchováván seznam nenavštívených uzlů a seznam navštívených uzlů. Dijkstrův algoritmus cyklicky prochází seznam nenavštívených uzlů, dokud není prázdný. Přičemž v každém průchodu cyklu se přesune jeden uzel ze seznamu nenavštívených do seznamu navštívených. Přesouvá se ten uzel, který má nejmenší vzdálenost. Pro každý uzel, do kterého vede hrana, se provede porovnání, jestli není vzdálenost kratší než uložená. Pokud je podmínka vyhodnocena jako pravdivá (cesta je kratší), provede se aktualizace délky. V opačném případě se neprovede žádná změna. Níže je uveden bodový postup algoritmu.

- INICIALIZACE:
    - vytvoření množiny uzlů `X`,
    - do množiny `X` vložit počáteční uzel `s`,
    - vytvoření asociativního pole pro každý uzel `D(u)`,
    - inicializace hodnoty pole `D` takto:
        - pro počáteční uzel `s = 0`,
        - pro každý uzel `u` sousedící s počátečním uzlem `s = ohodnocení hrany (s; u)`,
        - pro ostatní uzly = nekonečno

- VÝPOČET:
  - dokud nejsou v množině `X` všechny uzly grafu `X`, opakovat:
    - nalezení uzlu `w` s minimální hodnotou `D(w)`,
    - přidání uzlu `w` do množiny `X`,
    - pro každý uzel `u` sousedící s uzlem `w`, který není v množině `X`, provést:
      - hodnota `D(u)` je minimum ze stávající hodnoty a `D(w)` plus `ohodnocení hrany (w; u)`.

Efektivita Dijkstrova algoritmu závisí na implementaci. Nejjednodušší implementace používá pro uložení prioritní frontu a operace nalezení nejbližšího uzlu je lineární prohledávání. V tomto případě je asymptotická časová složitost `O(V2+E)`, kde `V` je počet vrcholů a `E` je počet hran. Pro řídké grafy může být algoritmus implementován mnohem efektivněji – graf je uložen pomocí seznamu sousedů a funkce nalezení nejbližšího uzlu je implementována pomocí binární nebo Fibonacciho haldy. Pak je složitost `O((E+V) log V)`, resp. pro Fibonacciho haldu `O(E+V log V)`.

2.1.2 Bellmanův-Fordův algoritmus

Algoritmus počítající nejkratší cestu v ohodnoceném grafu z jednoho uzlu do koncového. Výhodou oproti Dijkstrovu algoritmu je možnost použít i záporně ohodnocené hrany. Nicméně Dijkstrův algoritmus je rychlejší. Používá se např. ve směrovacím protokolu RIP.

Princip fungování je velice podobný Dijkstrovu algoritmu, neboť shodně používá metodu relaxace hran, která zjišťuje aktuálně nastavenou hodnotu nejkratší vzdálenosti od počátečního uzlu. Následující postup je opět totožný – pokud je zjištěno, že vzdálenost uzlu je vyšší než hodnota z nynějšího uzlu plus ohodnocení hrany, je vzdálenost aktualizována (snížena). Zásadní odlišnost od Dijkstrova algoritmu tkví v průchodu grafem – při průchodu všemi následníky uzlu se tento uzel „neuzavře“, nýbrž je procházen vícekrát. Více napoví následující pseodokód.

```
bellman-ford(vrcholy, hrany, zdroj)

// krok 1: inicializace grafu
for each v in vrcholy
  if v=zdroj then v.vzdálenost := 0
             else v.vzdálenost := nekonečno
  v.předchůdce := null

// krok 2: opakovaně relaxovat hrany
for i from 1 to size(vrcholy)-1
  for each h in hrany    // h je hrana z u do v
    u := h.počátek
    v := h.konec
    if u.vzdálenost + h.délka < v.vzdálenost
      v.vzdálenost := u.vzdálenost + h.délka
      v.předchůdce := u
```

```
// krok 3: kontrola záporných cyklů
for each h in hrany
  u := h.počátek
  v := h.konec
  if u.vzdálenost + h.délka < v.vzdálenost
    error "Graf obsahuje záporný cyklus."
```

Asymptotická složitost algoritmu je `O(V*E)`, kde `V` je počet vrcholů a `E` počet hran.

2.1.3 Floydův-Warshallův algoritmus

Algoritmus pro hledání nejkratší cesty v orientovaném grafu s hranami různých vah. Jediný průchod algoritmu vypočítá nejkratší cestu mezi všemi dvojicemi vrcholů. Funguje na principu porovnávání všech možných cest v grafu mezi všemi dvojicemi vrcholů. Postupně vylepšuje odhad na nejkratší cestu do té doby, než je zřejmé, že je odhad optimální. Více o fungování algoritmu napoví komentovaný pseudokód.

```
// Předpokládáme funkci cenaHrany(i, j) vracející cenu hrany z i do j
// (pokud hrana neexistuje, cenaHrany = nekonečno)
// Dále, N je počet vrcholů a cenaHrany(i, i) = 0

int cesta[][]; // Dvourozměrné pole. V každém kroku algoritmu je cesta[i][j]
               // nejkratší cesta z i do j použitím 1. až k-té hrany.
               // Všechny hrany cesta[i][j] jsou inicializovány funkcí
               // cenaHrany(i,j);

procedure FloydWarshall ()
   for k := 1 to N
   begin
      foreach (i,j) in (1..N)
      begin
        cesta[i][j] = min(cesta[i][j], cesta[i][k] + cesta[k][j]);
      end
   end
endproc
```

2.1.4 A*

Algoritmus vycházející z Dijkstrova algoritmu, využívá totožné principy, navíc však přidává heuristický prvek. Principem algoritmu je použití hladového principu pro nalezení optimální trasy z bodu A do bodu B. Optimální trasou se rozumí nejkratší, nejlevnější, nejrychlejší apod. – v závislosti na požadavcích. Důležitým prvkem algoritmu je funkce `f(x)`, která ohodnocuje jednotlivé uzly pro určení pořadí, ve kterém se mají procházet. Tato funkce se skládá ze dvou dílčích funkcí: (i) funkce představující vzdálenost mezi počátkem a daným uzlem, (ii) heuristická funkce odhadující správnost postupu. Příkladem může být heuristika vzdálenosti vzdušnou čarou, jelikož je to nejkratší možná cesta.

A* algoritmus probíhá následovně – vytvoří se prioritní fronta nenavštívených uzlů, přičemž čím je hodnota funkce `f(x)` menší, tím má uzel vyšší prioritu. Následně se v každém kroku odebere uzel s nejvyšší prioritou a pro jeho sousedící uzly jsou spočítány dílčí funkce (vzdálenost a heuristika). Tyto uzly jsou přidány do prioritní fronty. Algoritmus pokračuje, dokud nemá konečný uzel menší vzdálenost od počátku než kterýkoliv jiný uzel z fronty nebo dokud není fronta vyprázdněna. Hodnota vzdálenosti

koncového uzlu je výsledek – nejkratší cesta grafem. Chceme-li znát konkrétní cestu, je nutné udržovat seznam uzlů na této cestě. Níže je uveden pseudokód A* algoritmu.

```
function A*(start, cíl)
    closedset := prázdná množina // Množina již uzavřených uzlů.
    openset := množina obsahující počáteční uzel // Množina otevř. uzlů.
    g_skore[start] := 0 // Délka aktuální optimální cesty.
    h_skore[start] := heuristický_odhad_vzdálenosti(start, cíl)
    f_skore[start] := h_skore[start] // Předpokládaná délka cesty mezi
                                     // startem a cílem jdoucí přes y.
    while openset is not empty
        x := otevřený uzel s nejmenši hodnotou f_skore[]
        if x = cíl
            return rekonstruuj_cestu(přišel_z[cíl])
        vyjmi x z openset
        přidej x do closedset
        foreach y in sousední_uzly(x)
            if y in closedset
                continue
            stávající_g_skore := g_skore[x] + d(x, y)

            if y not in openset
                add y to openset
                stávající_je_lepší := true
            elseif stávající_g_skore < g_skore[y]
                stávající_je_lepší := true
            else
                stávající_je_lepší := false
            if stávající_je_lepší = true
                přišel_z[y] := x
                g_skore[y] := stávající_g_skore
                h_skore[y] := heuristický_odhad_vzdálenosti(y, cíl)
                f_skore[y] := g_skore[y] + h_skore[y]
    return failure

function rekonstruuj_cestu(aktuální_uzel)
    if přišel_z[aktuální_uzel] is set
        p = rekonstruuj_cestu(přišel_z[aktuální_uzel])
        return (p + aktuální_uzel)
    else
        return aktuální_uzel
```

Časová složitost A* algoritmu je přímo závislá na zvolené heuristice. V nejhorším případě je počet prozkoumaných uzlů exponenciální vzhledem k délce řešení. V optimálním případě je složitost polynomiální.

2.2 Problém obchodního cestujícího

Jedná se o jeden z nejznámějších a nejpopulárnějších kombinatorických problémů, který přímo vystihuje zadání této práce – najít nejkratší „okružní“ cestu mezi mnoha body. Příčinou populárnosti algoritmu je bohatá historie, která začíná již kolem roku 1759, kdy se úlohou tohoto typu zabýval švýcarský matematik Leonhard Euler. Impulsem pro další zájem se stal rozvoj lineárního a celočíselného programování (druhá polovina dvacátého století). Problém obchodního cestujícího má obrovské množství praktických aplikací – v logistice, plánování, výrobě VLSI obvodů, krystalografii apod. Proto ani v současné době

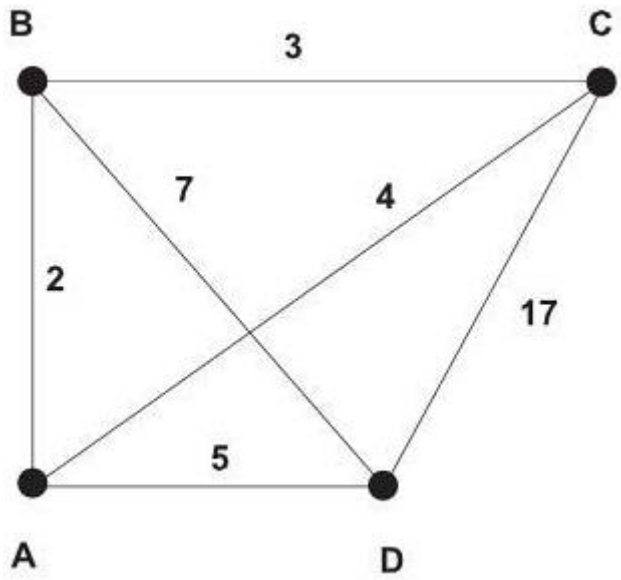
nepolevuje zájem o řešení problému. Jelikož se jedná o NP-těžkou úlohu, je pravděpodobné, že se nepodaří nalézt optimální řešení v polynomiálně omezeném čase. Což vedlo k návrhu mnoha aproximativních algoritmů.

Problém obchodního cestujícího lze snadno definovat tak, že obchodník musí při své cestě navštívit každé město právě jedenkrát a vrátit se do výchozího bodu. Přičemž je třeba minimalizovat jeho cestovní náklady, což je cena přemístění mezi jednotlivými městy a což obvykle přímo závisí na vzdálenosti mezi městy. Cílem je tedy navrhnout takovou okružní cestu, kde součet nákladů nutných pro uskutečnění cesty bude minimální.

2.2.1 Tradiční algoritmy

Úplný (dokonalý) algoritmus, který zaručuje nalezení optimálního řešení pro `n` měst, musí vyzkoušet všechny permutace množiny měst `1` až `n`, kterých je `n!`. Na první pohled je jasné, že tento algoritmus je již pro relativně malé hodnoty `n` z hlediska časové náročnosti nepoužitelný. Jak již bylo řečeno, zřejmě se nepodaří nalézt algoritmus, který by umožňoval nalezení optimální trasy v polynomiálně omezeném čase. Bylo a je tedy věnováno velkého úsilí pro nalezení alespoň přibližných algoritmů.

Zřejmě nejjednodušší je hladový algoritmus. Ten je založen na myšlence nalezení nejbližšího souseda k právě navštívenému městu. V prvotní fázi je náhodně zvoleno město `x`, ke kterému je vyhledáno nejbližší (dosud nenavštívené) město `y`. Obdobně je pokračováno dále až do doby, než je množina nenavštívených měst prázdná. Ačkoliv bylo nalezeno přípustné řešení, mnohdy se může lišit od optimálního. Na následujícím obrázku je zobrazen případ „chybného“ (neoptimálního) výsledku, kdy algoritmus využije drahou cestu mezi body `C` a `D`. Výsledná cesta vypočítaná algoritmem bude 27 (2 + 3 + 17 + 5) – cesta `A-B-C-D-A`. Přičemž optimální trasa je `A-C-B-D-A`, tedy cena 19 (4 + 3 + 7 + 5).



Obrázek 1 – Demonstrace neoptimálního výpočtu trasy

Mnohem úspěšnější jsou algoritmy založené na lokálním prohledávání. Často je použita jednoduchá heuristika, která je v literatuře značena jako 2-opt algoritmus. V takovém případě algoritmus začne s náhodně zvolenou permutací měst `T` a snaží se ji dále vylepšovat. Následně je definováno okolí cesty `T` – do této množiny patří všechny cesty, které se od `T` liší výměnou dvou přímo nenavazujících hran z cesty `T`. Tato záměna se nazývá 2-výměnou. Pokud se v okolí cesty `T` nalézá cesta `T'` s nižšími náklady, pak `T'` nahradí původní `T`. Dále se postupuje obdobně až do doby, kdy již neexistuje v okolí `T` žádná cesta s nižšími náklady. Ve výsledku je trasa označena jako 2-optimální, což se ovšem nemusí rovnat optimální trase.

Z algoritmu 2-opt lze snadno odvodit zobecnění nazývané k-opt, kde je vybíráno právě `k` hran, resp. maximálně `k` hran pro záměnu. Se zvyšujícím se parametrem `k` je algoritmus schopen vypočítat sofistikovanější řešení, avšak logicky roste náročnost úlohy, a tedy i čas potřebný k vyřešení. V praxi se proto lze pouze výjimečně setkat s volbou parametru `k` vyšším než 3. Na této strategii je založen dnes zřejmě nejúspěšnější heuristický algoritmus, který navrhli Lin a Kernighan. Zásadním rozdílem je dynamická hodnota parametru `k`, která se mění v každé iteraci. Navíc se algoritmus nespokojí s prvním nalezeným vylepšením cesty `T`, ale hledá maximální snížení nákladů v dané iteraci.

2.2.2 Genetické algoritmy

Genetický algoritmus je obecně aplikací principů evoluční biologie pro nalezení řešení složitých problémů. V algoritmech jsou používány evoluční procesy známé z biologie – dědičnost, mutace, přirozený výběr a křížení. V osmdesátých letech docházelo k intenzivnímu zkoumání genetických algoritmů pro problém obchodního cestujícího. Postupně byly zdokonalovány tři reprezentace genetických operátorů, v nichž je cesta definována pomocí vektoru, jehož délka je rovna počtu měst, tedy `n`.

Prvním způsobem kódování je tzv. sousedská reprezentace (adjacency representation), kde se číslo `i` vyskytuje na pozici `j` právě tehdy, když cesta reprezentovaná tímto vektorem zahrnuje přesun z města `j` do města `i`. Problémem je, že ne všechny vektory obsahující `n` čísel reprezentují přípustná řešení problému. Za tímto účelem byly vytvořeny opravné algoritmy, nicméně ani po jejich aplikaci nedochází k dostatečně efektivním výpočtům, proto byl tento způsob zamítnut.

Druhým typem je ordinální reprezentace (ordinal representation), která vychází z historicky nejstarších pokusů o řešení problému obchodního cestujícího pomocí genetických algoritmů. Zakódovaná cesta je reprezentována jako seznam `n` měst, kde `i`-té číslo v seznamu může nabývat hodnoty $j \in \{1, \dots, (n - i + 1)\}$. Pro tento typ reprezentace je nutný referenční seznam měst, kde jsou uvedena všechna města v pevně daném pořadí. Výhodou tohoto způsobu kódování je skutečnost, že vznikne ve všech případech přípustné řešení – bez nutnosti aplikovat jakékoliv další opravné procedury. Nicméně pouhou kombinací ordinální reprezentace a jednobodového křížení nelze při řešení problému obchodního cestujícího dosáhnout dobrých výsledků, a proto byl i tento přístup zamítnut.

Na druhou stranu je nutno podotknout, že ordinální reprezentaci lze použít v jiných problémech, např. plánovací úlohy z oblasti tvorby rozvrhů.

Zřejmě nejjednodušším a zároveň nejúspěšnějším způsobem kódování je přirozená reprezentace cesty (path representation), kdy chromozom (5, 6, 3, 1, 4, 2) reprezentuje okružní cestu 5 – 6 – 3 – 1 – 4 – 2 – 5.

2.2.3 Hybridní genetické algoritmy

Přes veškeré úsilí při návrhu vhodných operátorů genetických algoritmů se nepodařilo vážně konkurovat heuristickému algoritmu, který navrhli Lin a Kernighan. A to jak po stránce kvality nalezeného řešení, tak i času potřebného k výpočtu. Což byla inspirace pro vytvoření různých hybridních spojení – evoluční algoritmus s použitím heuristiky založené na lokálním prohledávání. Postupně tak bylo navrženo velké množství hybridních genetických algoritmů, které využívají hladový algoritmus, 2-opt, 3-opt a mnohé další heuristiky.

Příkladem může být přístup, který zabraňuje možnému incestu při tvoření nových individuí a tím zlepšuje výkon genetického algoritmu pro řešení problému obchodního cestujícího. Jiným příkladem může být algoritmus používající generování náhodných (fiktivních) souřadnic měst a následného výpočtu na posunuté mapě.

Úplně na konec je možné uvést zajímavý přístup založený čistě na evolučním algoritmu využívající tzv. inver-over operátor. Ten zpracovává jednotlivá individua s jinými, ale po dokončení se nově vzniklé individuum porovná s původním a lepší z nich v populaci zůstává. Na základě výsledků, které autoři algoritmu prezentovali, dospěli k závěru, že tento algoritmus je v současnosti pravděpodobně nejlepším evolučním algoritmem pro řešení úlohy obchodního cestujícího.

3 Srovnání poskytovatelů mapových podkladů

Neodmyslitelnou součástí aplikace je bezesporu vizualizace. V případě potřeby softwaru transportní firmy bude vhodné vizualizovat trasu cesty pro každý automobil. Případně zobrazit cílové místo rozvážky na mapě. Právě z tohoto důvodu je nezbytné, aby aplikace disponovala mapou, kde budou uvedené informace zobrazeny. Jelikož poskytovatelů mapových dat existuje poměrně velké množství, je v této kapitole provedeno srovnání nejznámějších z nich.

3.1 Mapy.cz

Velice populární český mapový portál vlastněný společností Seznam.cz. Pro Českou republiku obsahuje nejpodrobnější mapová data, neboť společnost provádí vlastní mapování terénu (viz dále). Obrovskou výhodou map je také rozsáhlá databáze bodů zájmu, tzv. POI. Velice snadno je tedy k nalezení např. nejbližší kino, restaurace nebo třeba benzínová pumpa.

Práce s mapou je na dobré úrovni, obsahuje základní funkce, kterými je posun a přibližování/oddalování mapy, možnost zjistit GPS souřadnice daného místa nebo sdílení aktuálního pohledu na mapu emailem apod. Vyhledávací modul je poměrně inteligentní, dokáže vyhledat přesnou adresu, ale také zadané GPS souřadnice, či dokonce název (sídlo) firmy. V posledním případě – hledání sídla společnosti – je aplikace napojena na webový portál firmy.cz a přebírá z této databáze informace. Poměrně vydařená funkce je plánovač tras. Je možné si zvolit nejen startovní a cílové místo, ale také několik průjezdných míst. Trasa může být vypočítána pro automobil, cyklistu nebo pěší túru. Uživatel není omezen na zadávání místa skrze klávesnici, neboť místa (start, cíl, průjezdy) je možné interaktivně zadávat pomocí myši přímo v mapě.

Aplikace obsahuje několik mapových podkladů.

- Obecný mapový podklad – klasický kreslený mapový podklad, který známe z autoatlasů. Obsahuje zjednodušené zobrazení silniční sítě s aktivními body zájmu. Součástí jsou podrobné plány měst, na kterých jsou vyznačeny jednosměrky a jiná dopravní značení. Jelikož se jedná o velice rozsáhlé dílo, jsou mapy pravidelně aktualizovány. Na webových stránkách projektu uvádějí aktualizaci 2-4 krát ročně.

- Fotografický mapový podklad – kombinace satelitního a leteckého snímkování. Pro celou Evropu jsou dostupné fotografie ze satelitního snímkování. Pro oblast České republiky využívá firma vlastní letecké snímky, které každoročně aktualizuje. Dle údajů se za jeden rok nafotí zhruba 1/3 republiky. Fotografické mapové podklady tedy mohou být až tři roky staré.

- Turistický mapový podklad – obdoba klasické papírové turistické mapy s měřítkem 1 : 5000. Obsahuje základní topografické vrstvy, výškopis, stínovaný reliéf, turistické značené trasy, naučné stezky, cyklotrasy a turistické body zájmu.

V mapách jsou znázorněny vrstevnice s odstupem 10 výškových metrů – v nejvyšším přiblížení až 5 výškových metrů. Tato mapa je dostupná pouze na území České republiky.

- Historický mapový podklad – zobrazuje naši republiku před více než 150 lety. Podklady pro tuto mapu jsou z let 1836-1852, přičemž jsou v měřítku 1 : 28800. Tento mapový podklad je spíše pro zajímavost, než pro nějaké praktické použití.

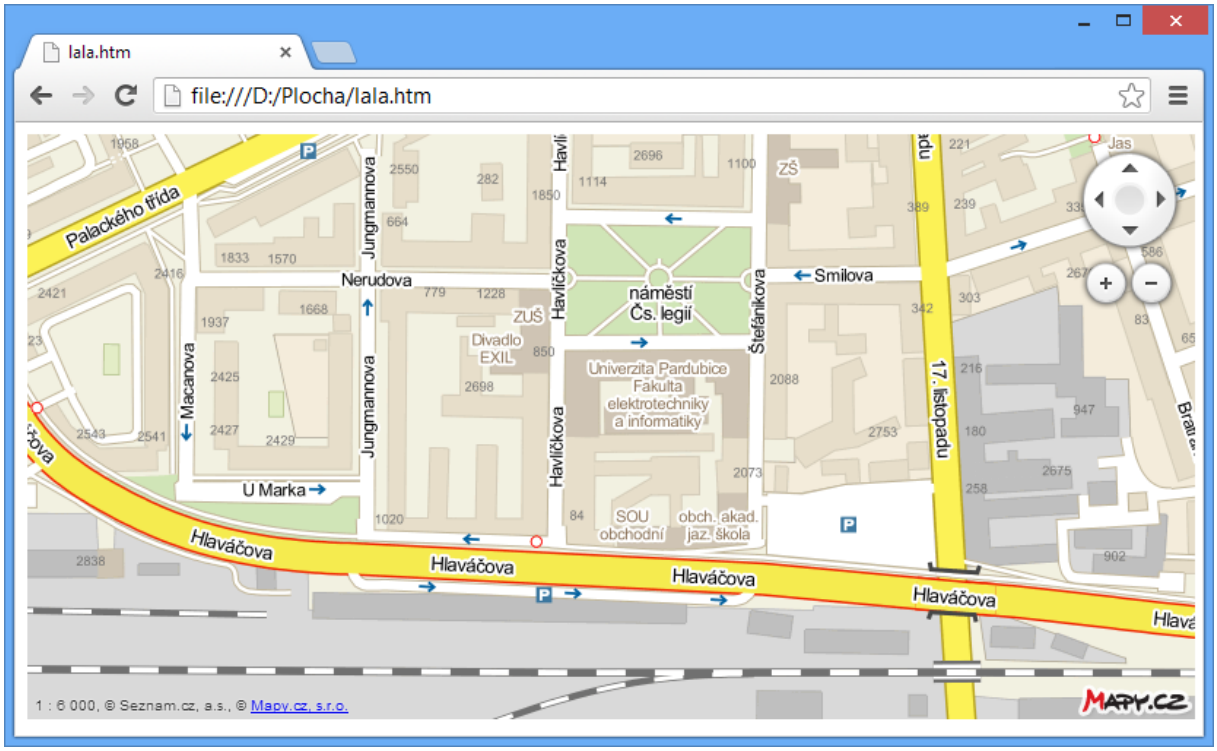
3.1.1 API

Rozhraní API pro vložení na vlastní stránky poskytuje poměrně slušné možnosti a variabilitu výsledného zobrazení. Používání API je zcela zdarma, a to dokonce i pro komerční použití.

Níže je uveden zcela elementární kód, který zobrazí na stránce mapu, kde bude jako střed označena budova Univerzity Pardubice – Fakulta elektrotechniky a informatiky.

```
// načtení a inicializace API
<script src="http://api4.mapy.cz/loader.js"></script>
<script>Loader.load()</script>

// mapa bude vložena do následujícího prvku DIV
<div id="mapa" style="width:800px; height:400px;"></div>

// nastavení zobrazení - střed mapy, přiblížení, typ mapy a ovládací prvky
<script type="text/javascript">
      var stred = SMap.Coords.fromWGS84(15.767, 50.033);
      var mapa = new SMap(JAK.gel("mapa"), stred, 15);
      mapa.addDefaultLayer(SMap.DEF_BASE).enable();
      mapa.addDefaultControls();
</script>
```

Výše uvedený kód zobrazí následující výstup v prohlížeči.



Obrázek 2 – Ukázka výstupu poskytovatele Mapy.cz

Značky

Pokud potřebujeme na mapě označit vlastní bod, poskytuje nám API metodu addMarker. Mnohdy je tímto způsobem označováno sídlo firmy nebo její provozovny. Značky mají mnoho dalších nastavení a vlastností. Ke každé značce je možné přidat zcela libovolný popisek (ve formátu HTML), dokonce můžeme změnit ikonu značky za vlastní. Dále je možné vytvořit tzv. posuvnou značku. Uživatel může díky této funkci vybrat libovolné místo na mapě a sdělit tak provozovateli stránek např. svoji adresu.

Poměrně praktickou funkcí jsou tzv. shluky značek. Pokud máme velké množství značek v mapě (řádově stovky), a ty se navzájem překrývají, je možné je sloučit do shluku. Uživatel v takovém případě vidí v oblasti výskytu mnoha značek jen jednu – shluk s číslem udávající počet značek v dané oblasti. Jakmile provede přiblížení oblasti a již nedochází k překrytí jednotlivých značek, shluk se „rozpadne“ a zobrazí se již konkrétní značky.

Vektorové prvky

Skrze rozhraní geometrické vrstvy je možné do vlastní mapy vložit geometrické útvary. Což je velice silný nástroj v případě potřeby zobrazit nějakou oblast nebo pole působnosti. Dobrým příkladem může být např. oblast, kam daná firma rozváží své zboží za nižší cenu.

Rozhraní dovoluje vkládat nejen primitivní prvky typu čára a obdélník, podporovány jsou i složitější geometrické útvary. Zároveň nejsme omezeni jen na statickou definici vektorových obrazců. Data je možné načítat vzdáleně pomocí technologie AJAX.

Ostatní

Výše byly zmíněny pouze zajímavé (pro autora) vlastnosti a funkce API Mapy.cz. Těch je však mnohem více. Jmenovat můžeme třeba podporu pro formát GPX (standard pro popis dat z GPS) nebo geokódování – hledání zeměpisné pozice dle zadaného řetězce (dopředné geokódování) a hledání zeměpisných objektů na zadané souřadnici (zpětné geokódování).

3.2 Google Maps

Jak již samotný název napovídá, jedná se o mapovou aplikaci společnosti Google. Výhodou je pokrytí celého světa a lokalizace celé aplikace do desítek jazyků, včetně češtiny. Google se snaží o co nejširší podporu napříč zařízeními, proto jsou v současné době jejich mapy dostupné pro mobilní telefony, tak i pro tablety. Jelikož je Google vlastníkem mobilního operačního systému Android, logicky je aplikace přítomna v každém zařízení s Androidem.

Podobně jako ostatní mapové portály, tak i Google Maps nabízí mnoho mapových podkladů. Zvolit můžeme klasické zobrazení, které známe z autoatlasů. Zobrazení je dále možné přepnout do satelitního zobrazení. V tomto „módu“ jsou namísto mapy zobrazeny satelitní snímky. Bohužel jsou na mnoha místech tyto snímky poměrně staré (řádově až několik let), přičemž jejich aktualizace je nepravidelná. Některé snímky jsou zachyceny ve vyšší kvalitě – což bývají především populární místa, na druhé straně místa, která nejsou až tolik oblíbená, jsou zachycena v mnohem nižší kvalitě. Poměrně zajímavým pohledem je tzv. Aerial View, což byla odezva na konkurenční Bird’s-eye view v Bing Maps (viz dále). Jedná se o fotografie pořízené pod úhlem 45 stupňů, což zaručuje lepší pohled na detail mapy než klasický horní pohled. V České republice je však tímto způsobem nasnímáno pouze omezené množství míst – jedná se především o velká (krajská) města.

Samotná mapová aplikace nabízí jednoduché ovládání se všemi standardními funkcemi – posun mapy, přibližování/oddalování mapy, možnost sdílet aktuální pohled na mapu apod. Díky propojenosti se sociální síti Google+ je možné ihned jakékoliv místo sdílet v této síti. Google Maps nabízí poměrně povedený plánovač tras, tedy vyhledání nejkratší trasy z bodu A do bodu B. Je možné přidat několik dalších průjezdných míst – něco jako funkce itinerář známá z navigačních přístrojů. Pro plánování trasy je možné využít výpočet pro osobní automobil nebo pro pěší chůzi. Integrovaný vyhledávací modul je velice inteligentní, dokáže vyhledat nejen adresu nebo místo, cizí mu nejsou ani názvy firem nebo třeba GPS souřadnice v mnoha různých formátech. Aplikace přidává několik dalších funkcí – je to počasí, které se zobrazí pro aktuálně zobrazené místo na mapě. Velice praktická je také funkce „Provoz“ zobrazující hustotu dopravy na silnicích nebo třeba funkce „Fotografie“, která obsahuje obrovskou databázi fotografií (převážně od obyčejných uživatelů) pro konkrétní místo. Pokud je uživatel přihlášen ke svému Google

účtu, získá navíc několik dalších funkcionalit – např. historii hledání, svoje domovské místo nebo třeba správu oblíbených míst na mapě.

V květnu 2007 přibyla do Google Maps převratná novinka – Street View. Google Maps s funkcí Street View umožňují prohlížet nejrůznější místa z celého světa prostřednictvím panoramatických snímků pořízených z ulice. Pro zachycení panoramatických fotografií je použit automobil se speciálním snímacím zařízením, který projíždí ulicemi. Uživatel může pomocí těchto fotografií mnohem snáz vyhledat požadované místo a ihned se podívat, jak daný objekt/ulice v reálu vypadá. Velice pozitivním faktem je prakticky 100% pokrytí České republiky v rámci Street View. Důležitou vlastností je soukromí – Google každou fotografii analyzuje a rozmazává obličeje osob a také poznávací značky automobilů.

Google Maps, to nejsou jenom klasické „venkovní" mapy, k dispozici je mnoho dalších funkcí a vlastností. Na závěr je vybráno ještě několik funkcionalit pro představu, jak široký záběr tato aplikace poskytuje. Funkce nazvaná Indoor umožňuje libovolné soukromé osobě (společnosti) zmapovat vlastní prostory a zveřejnit je v rámci Google Maps. Pokud zákazník vstoupí do objektu podporovaného touto technologií, může v rámci budovy/haly/objektu zobrazovat svoji pozici či nalézt požadované místo, zkrátka stejně jako v klasické mapě. Úplně z jiné oblasti je aplikace Google Latitude. Jedná se o lokalizační aplikaci pracující v rámci Google Maps. Pokud uživatel dovolí sdílet svoji aktuální polohu s ostatními uživateli (které musí kvůli ochraně soukromí osobně povolit), mohou jeho přátelé vidět, kde se aktuálně nachází.

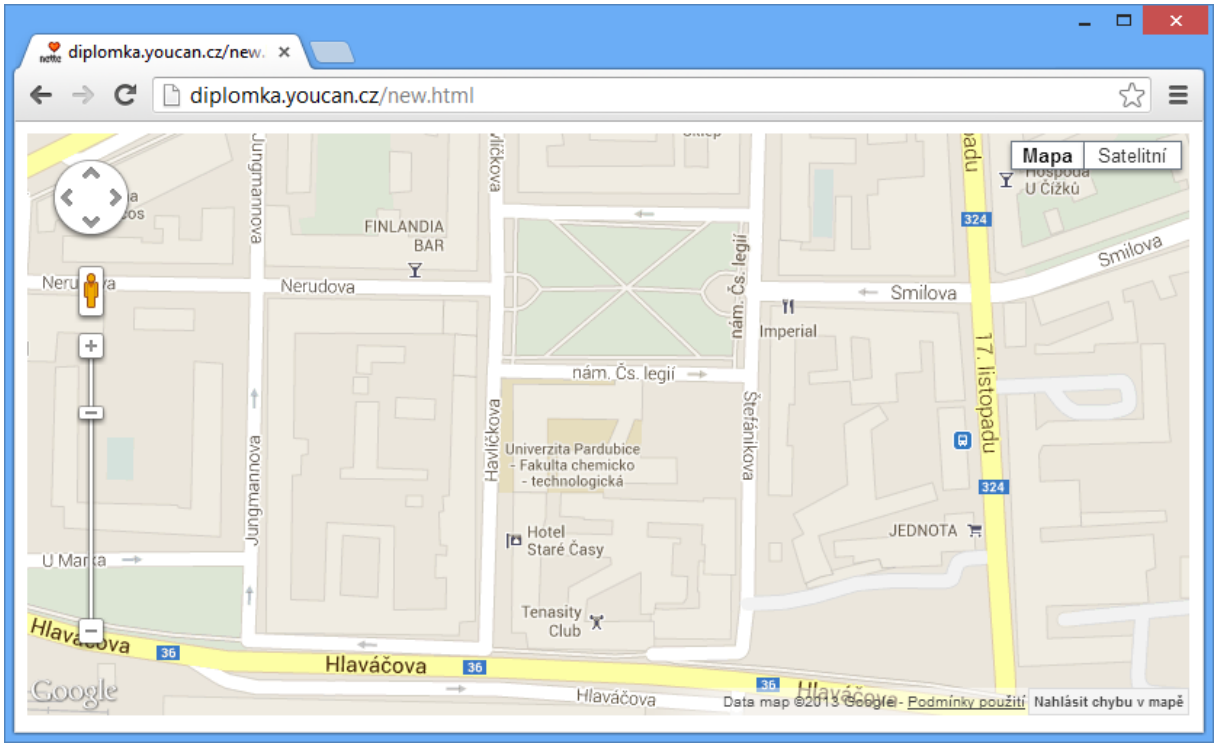
3.2.1 API

Díky Google Maps API může vývojář vložit vlastní mapu do své internetové stránky. A nejen to, podporováno je mnoho dalších funkcionalit. Používání API je zdarma, nicméně pouze pro nekomerční použití a s jistými limity. Pro komerční prostředí je možné zakoupit vyšší verzi, tzv. Maps API for Business. Na rozdíl od neplacené verze jsou mnohonásobně navýšeny limity používání – např. neplacená verze umožňuje výpočet trasy z bodu A do bodu B pouze pro 10 průjezdných bodů a je limitováno 2500 požadavků za den. U placené varianty je možné zadat až 23 průjezdných bodů a limit je 100.000 požadavků za den. Dalším příkladem může být rozlišení obrázků pro funkce Street View, u neplacené verze je to maximálně 640x640 px, u placené verze až 2048x2048 px. Pokud chceme API používat, je nutné se zaregistrovat a získat tzv. API key. Jedná se o přístupový klíč, který je pevně svázán s účtem a především zadanou doménou, na kterou se vážou dané limity. Možnosti Google Maps API jsou velmi rozsáhlé, je myšleno doslova na každý detail, a tak je možné nastavit a přizpůsobit téměř cokoliv. I přes rozsáhlost API (a tudíž i dokumentace), je manuál k API velice přehledný a vždy doplněný o praktické příklady.

Níže je uveden elementární kód pro vykreslení mapy pomocí Google Maps. Střed mapy je nastaven na budovu Univerzity Pardubice – Fakulta elektrotechniky a informatiky.

```
// element DIV, do kterého bude vykreslena mapa
<div id="map-canvas" style="width: 800px; height: 400px"></div>

// načtení API
<script type="text/javascript"
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyAi1dDUK6UwPv0xKF_8E
QACPXwfgNtFjVg&sensor=false"></script>

<script type="text/javascript">
    // nastavení mapy (přiblížení, střed a typ mapy)
    var mapOptions = {
        zoom: 17,
        center: new google.maps.LatLng(50.033330, 15.767395),
        mapTypeId: google.maps.MapTypeId.ROADMAP
    }

    // inicializace mapy s definovaným nastavením
    var map = new google.maps.Map(
                        document.getElementById("map-canvas"),
                        mapOptions
    );
</script>
```

Výsledek předešlého zdrojového kódu naleznete na následujícím obrázku. Všimněme si neaktuálnosti map, na obrázku je budova označena jako Fakulta chemicko-technologická.



Obrázek 3 – Ukázka výstupu poskytovatele Google Maps

Události

Pomocí nastavení dané mapy můžeme definovat tzv. eventy (česky události), k dispozici máme několik typů: click, dbclick, mouseup, mousedown, mouseover, mouseout. Jak již názvy napovídají, jedná se o ekvivalenty událostí jazyka JavaScript. Jedná se o velice

užitečnou věc, neboť můžeme definovat akce, které se vykonají při dané události. Pokud bychom měli být konkrétní, můžeme si díky událostem od uživatele vyžádat jeho adresu a tu si pak uložit do své lokální databáze.

Ovládací prvky

Při vykreslování vlastní mapy můžeme ovlivnit také zobrazování ovládacích prvků. Je možné povolit zobrazení/skrytí těchto ovládacích prvků.

- panControl – navigační šipky sloužící k posunu v mapě,
- zoomControl – ovládací prvek přiblížení/oddálení,
- mapTypeControl – přepínač mapového podkladu (satelitní, klasická mapa),
- scaleControl – měřítko mapy,
- streetViewControl – ikona „panáčka“ indikující funkci Street View,
- overviewMapControl – malá přehledová mapka.

Pro každý výše zmíněný ovládací prvek si můžeme určit i jeho pozici na mapě. Je možné použít konstanty jako je `TOP_LEFT`, `TOP_CENTER` nebo `TOP_RIGHT`. Obdobně pak `LEFT_CENTER`, `LEFT_BOTTOM` apod.

Programátor není omezen pouze na definované ovládací prvky, nýbrž lze do mapy dodat zcela vlastní ovládací prvek. Ten se definuje pomocí klasického HTML jazyka a je možné jej opět libovolně v mapě umístit a samozřejmě mu přiřadit specifickou funkci – např. po kliknutí na tlačítko se mapa vystředí na sídlo společnosti.

Vrstvy

Vrstvy jsou objekty na mapě, které jsou pevně svázány s konkrétním místem (souřadnicí), tudíž se posouvají při pohybu nebo přibližování mapy. Google Maps dále rozděluje vrstvy následovně.

- Markers – označení pro jediný bod na mapě, jsou zobrazeny jako ikona a využívány pro označení jednotlivých konkrétních míst, např. provozoven dané společnosti,
- Lines – česky čáry, tedy úsečky z bodu A do bodu B,
- Areas – ohraničené oblasti (obdélník, trojúhelník, mnohoúhelník…), např. vymezení sídla společnosti,
- informační okna – speciální typ vrstvy, většinou zobrazuje textové nebo obrázkové popisky k danému místu.

Nastavení a vlastní přizpůsobení vrstev je opravdu bohaté a umožní každému vytvořit výsledný produkt přesně podle jeho představ. Pro úplnost je zde uvedeno (dle autora) několik zajímavých postřehů pro vrstvy – markery je možné animovat, měnit jim velikost a nastavovat i vlastní ikonu. Při tvorbě čar je možné použít nejen plné čáry, ale také čáry přerušované nebo tečkované. Poměrně zajímavou funkcí je také možnost interaktivně vybrat oblast z mapy.

Služby

Google Maps API neposkytují pouze zobrazení vlastní mapy, součástí onoho balíku jsou i tzv. Services (česky služby). Jedná se v podstatě o jednoduché volání skrze protokol HTTP, přičemž odpověď vrací požadované informace. Odpověď je možné získat ve formátu XML nebo JSON. Volitelně je možné specifikovat jazyk odpovědi, můžeme tak např. získat čas v českém jazyce (3 dny, 21 hodin) nebo třeba ve francouzštině (3 jours 21 heures).

Jsou k dispozici následující služby.

- Directions API – vypočítá vzdálenost mezi dvěma body, přičemž je možné přidat i průjezdné body. Jedná se v podstatě o plánovač tras, který je dostupný na „hlavním“ mapovém portálu Google Maps.
- Distance Matrix API – poskytuje vzdálenost a čas cestování mezi dvěma místy.
- Elevation API – na základě souřadnic určí přibližnou nadmořskou výšku.
- Geocoding API – jedná se o převod „textové adresy“ na souřadnice. Např. po zadání textu `Univerzita Pardubice` vrací `50.06223490, 15.77294790`.
- Time Zone API – služba vracející časové pásmo pro zadané souřadnice.

Pro úplnost jeden příklad. Po zavolání následující adresy je vrácena vzdálenost mezi `Univerzita Pardubice` a `ČVUT Praha`. Počítána je cesta osobní automobilem. Výsledkem je 116 km a čas 1 hodina a 19 minut. Výstup je ve formátu JSON.

```
http://maps.googleapis.com/maps/api/distancematrix/json?origins=Univerzita+Pardubice&destinations=ČVUT+Praha&sensor=false
```

Ostatní

Jak bylo v úvodu psaní o Google Maps API zmíněno, jedná se o velice rozsáhlou a komplexní službu a rozhodně není součástí této práce ji zcela detailně popisovat. Nicméně je na následujících řádcích zmíněno ještě několik dalších (dle autora) zajímavých funkcionalit.

V rámci vlastní mapy si může každý programátor zcela libovolně definovat styly mapy. Je tedy možné nadefinovat např. zobrazení silnic fialovou barvou, barvu lesů oranžově a

zakázat zobrazení vodních ploch. Jako poslední vyjmenovávanou funkci můžeme uvést tzv. knihovny (Libraries). Existuje jich celá řada, za zmínku určitě stojí možnost zobrazit v mapě AdSense reklamy nebo třeba ikony s počasím.

3.3 Bing Maps

Poměrně nový mapový portál, který byl oficiálně spuštěn až v květnu roku 2009. Patří pod hlavičku vyhledávače Bing, který vlastní společnost Microsoft. Microsoft logicky své mapy hodně prosazuje, tudíž jsou k nalezení v jejich výrobcích – v operačním systému Microsoft Windows 8 a také v mobilním operačním systému Windows Phone.

Bing Maps pokrývají svými mapami celý svět. Při prohlížení mapy si může uživatel vybrat z několika mapových podkladů. Podporováno je základní zobrazení, na které jsme zvyklí z autoatlasů, satelitní snímky nebo také několik poměrně unikátních zobrazení. Mezi ně rozhodně patří tzv. Bird's-eye view, což je pohled z ptačí perspektivy. Snímky jsou pro tento podklad fotografovány z nízko letícího letadla, přičemž jsou pořizovány pod úhlem 45 stupňů. Bohužel jsou tyto záběry dostupné pouze pro vybrané lokace (velká americká a evropská města). Dalším velice zajímavým pohledem je tzv. Streetside. Snímky pro tuto variantu zobrazení jsou zaznamenány pomocí speciálního vozidla, které projíždí ulicemi a pořizuje panoramatické fotografie. Ty jsou pak zpracovány a je z nich vytvořen mapový podklad. Uživatel si tak může snadno zobrazit požadované místo tak jako by na místě stál. Za zmínku stojí také možnost prohlížení noční oblohy.

Součástí je inteligentní vyhledávač podporující hledání lokací a také firem. Je napojen na databázi wikipedia.org, ze které dodatečně získává relevantní informace o hledaném místu. K dispozici je modul pro výpočet trasy, je tedy možné naplánovat si nejkratší cestu z místa A do bodu B. Výpočet trasy je dostupný pro osobní automobil, nákladní automobil a pěší chůzi.

Drobnou nevýhodou pro českého uživatele je špatná lokalizace aplikace do češtiny. Jenom malé množství popisků, nabídek a nastavení je lokalizováno do češtiny, převážná většina je zobrazena anglicky.

V nedávné době (únor 2012) Microsoft navázal spolupráci s firmou Nokia, která vyvíjí vlastní mapový software nazvaný Nokia Maps. Vývojové týmy těchto dvou projektů (Bing Maps, Nokia Maps) by měly spojit svoje síly a především svoje know-how. V brzké době bychom se tedy mohli těšit na vylepšenou verzi Bing Maps.

3.3.1 API

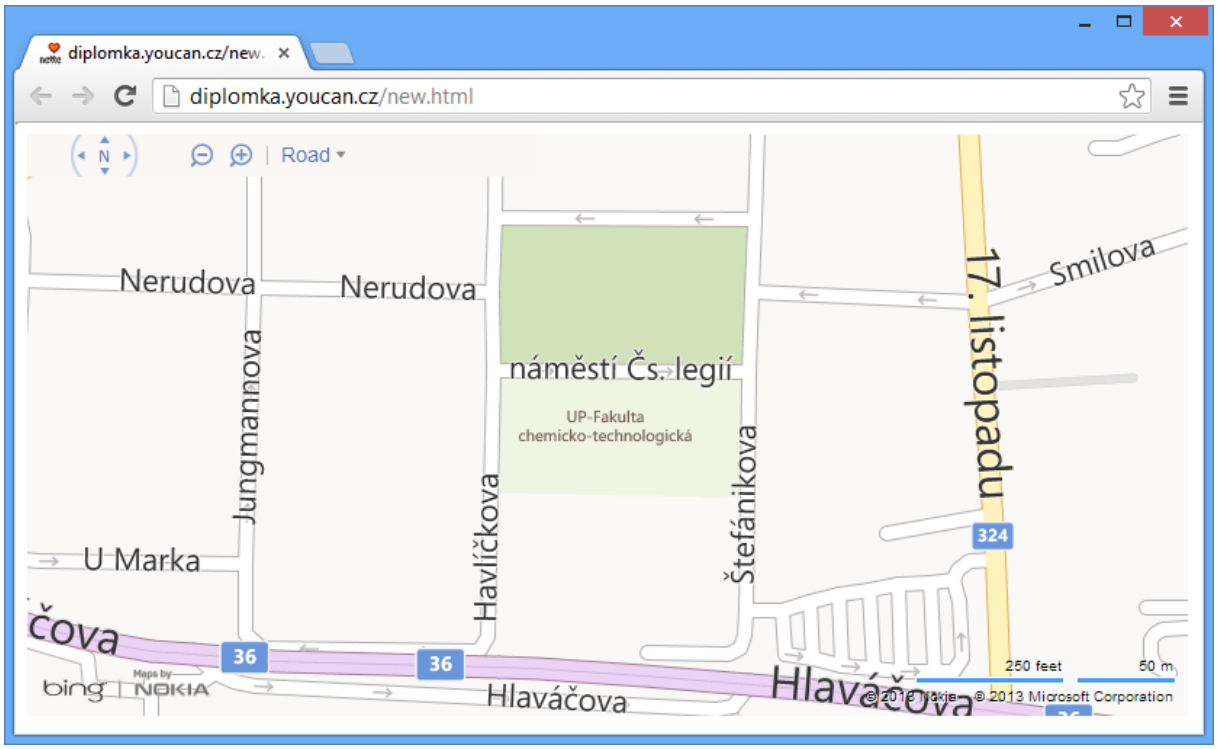
První věcí, která na API pro Bing Maps zaujme, je obrovská rozsáhlost dokumentace. Kompletní dokumentace v PDF formátu má neuvěřitelných 354 stran. Na druhou stranu je nutné zmínit, že dokumentace je poměrně přehledná a obsahuje mnoho příkladů, tedy zdrojových kódů. Nutností pro používání API je založení speciálního účtu na bingmapsportal.com. Následně je nutné získat tzv. API key. K dispozici je buď Basic varianta, která je omezena na 50 tisíc transakcí za 24 hodin nebo neomezená varianta

Enterprise, která je ovšem zpoplatněna. API key se registruje vždy na jednu konkrétní doménu a není možné jej použít jinde.

Elementární kód použití API Bing Maps je následující (API key je určen pro doménu diplomka.youcan.cz). Střed mapy je zaměřen na budovu Univerzity Pardubice – Fakulta elektrotechniky a informatiky.

```
// mapa bude vložena do tohoto prvku DIV
<div id='mapDiv' style="width:800px; height:400px;"></div>

<script type="text/javascript"
src="http://ecn.dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=7.0">
</script>

<script type="text/javascript">
    // vytvoření nového objektu za použití API key,
    // nastavení středu, přiblížení a typu mapy
    var map = new Microsoft.Maps.Map(
        document.getElementById("mapDiv"),
        {
            credentials: "AhzXxdaQbgbfcQl2qtEWqCATIuZj-QUlV_teb-
Bdv2KcMUyfZUaCdSKuUGxOhPwj",
            center: new Microsoft.Maps.Location(50.033330, 15.767395),
            mapTypeId: Microsoft.Maps.MapTypeId.road,
            zoom: 17
        }

    );
</script>
```

Na obrázku níže je zobrazen výsledek výše zmiňovaného kódu. Všimněme si, že mapy obsahují stará data. Namísto popisku Fakulta elektrotechniky a informatiky je zobrazen starý nápis Fakulta chemicko-technologická.



Obrázek 4 – Ukázka výstupu poskytovatele Bing Maps

Entity

Bing Maps umožňují vkládat do vlastní mapy tzv. Entity. Jedná se v podstatě o vkládání značek a různých prvků do mapy jako jsme zvyklí z jiných mapových služeb. Entity jsou rozděleny do dvou kategorií – první jsou tzv. Pushpin (česky připínák), což je obdoba vlastní značky v mapě. Druhou kategorií jsou tzv. Shape (česky tvary), přičemž je podporováno mnoho geometrických tvarů.

3.4 Ostatní poskytovatelé mapových dat

Mapových služeb existuje poměrně hodně. Nicméně není obsahem této práce popisovat všechny produkty na trhu. Níže je uvedeno několik dalších – dle autora zajímavých – projektů.

3.4.1 Yahoo! Maps

Ačkoliv není společnost Yahoo! v České republice tolik známá, jedná se o velikou mezinárodní internetovou společnost, která je spjata s vyhledáváním na internetu – a to především ve Spojených státech a Velké Británii. Součástí služeb této společnosti je mapový portál nazvaný Yahoo! Maps.

Jedná se o poměrně povedený mapový portál, který vedle standardních funkcí (posun, přiblížení apod.) obsahuje také mnoho pokročilých vlastností. Jmenovat můžeme např. několik mapových vrstev, zobrazování aktuální hustoty dopravy, aktuální počasí pro zobrazovanou lokalitu nebo pokročilou funkci plánování trasy s podporou průjezdových bodů. Nechybí integrace inteligentního vyhledávače, který umožňuje najít nejen místa, ale také firmy a zajímavá místa (body zájmu – POI).

Nevýhodou pro českého uživatele je jistě absence české lokalizace, nicméně pokrytí a mapové podklady pro Českou republiku jsou na slušné úrovni. Služba poskytuje API pro zobrazení vlastní mapy na stránkách, to je bohužel od srpna 2011 mimo provoz, neboť Yahoo! Maps využívají mapové podklady služby Nokia Here (viz dále) a celé API „poskytují“ pouze zprostředkovaně skrze partnerskou službu.

3.4.2 Nokia HERE

Značku Nokia zná díky mobilním telefonům dnes zřejmě každý. S přibývajícím počtem mobilních zařízení obsahující GPS čip se proto i tato společnost rozhodla vytvořit vlastní mapovou aplikaci (navigaci). Původní název Ovi Maps se změnil v Nokia Maps, přičemž od loňského roku (2012) se používá dnes aktuální pojmenování Nokia Here. Ačkoliv je tato služba primárně určena jako mobilní aplikace pro všechna různá zařízení (Windows Phone, Symbian, Series 40, MeeGo, iOS, Firefox OS a Android), je dostupná i verze pro klasický internetový prohlížeč.

Už na první pohled je vidět, že je portál nový – zaujme moderní minimalistický design a mnoho nadstandardních funkcí. Za samozřejmost lze brát základní funkce, na které jsme zvyklí z jiných mapových portálů. Rozhodně ale potěší přítomnost mnoha výjimečných funkcí – 3D pohled vykresluje mapu z ptačí perspektivy a umožňuje tak uživateli realistické zobrazení mapy. Dostupná je též obdoba funkce Street View u Google Maps (panoramatické fotografie přímo z ulice) a také velice podrobný plánovač trasy. Ten umožňuje při plánování cesty zvolit použití/vynechání tunelů, dálnic, nezpevněných cest, placených úseků nebo třeba lodní přepravy. Z dalších funkcí lze jmenovat zobrazování veřejné dopravy, hustoty provozu a lokálních informací (body zájmu).

Naprostou novinkou – uvedenou koncem května 2013 – je tzv. LiveSight. Tato funkce je popisována jako rozšíření reality. Pokud v aplikaci z nabídky vybereme vrstvu LiveSight, zapne se fotoaparát umístěný v používaném zařízení a začne se snímat okolí. Výstup z fotoaparátu bude opatřen dodatečnými značkami informujícími o zajímavých místech a vzdálenosti od pozorovatele. V naprosto neznámém prostředí tak lze snadno vyhledat např. kavárnu nebo zjistit název nejbližší zastávky.

Podobně jako u Yahoo! Maps, bohužel i zde chybí česká lokalizace webového portálu a obecně je podpora pokročilých funkcí (rozšířená realita, panoramatické fotografie) pro Českou republiku velice omezena.

Součástí je i rozsáhlé API, které není omezeno pouze pro webové stránky. Tuto službu je možné využívat i jako součást klasické desktopové nebo mobilní aplikace. Dokumentace je

poměrně dobře zpracovaná a přehledná, přičemž je přítomno množství ukázkových zdrojových kódů. Obecně obsahuje API pro Nokia Here mnoho funkcí a možností přizpůsobení – dostupné jsou všechny funkcionality popisované u jiných mapových produktů (markery, vrstvy, oblasti apod.).

3.4.3 OpenStreetMap

Všechny doposud popisované služby a k nim příslušící mapové portály byly založené na profesionálních (komerčních) mapových datech. Projekt OpenStreetMap, jak již samotný název napovídá, se snaží jít opačnou cestou – všechna geografická data jsou k dispozici zcela zdarma. Co je však ještě podstatnější, celý projekt je založen na komunitě. Kdokoliv může cokoliv přidat nebo upravit.

Obecně žádné API pro zobrazení map na webových stránkách neexistuje. Projekt totiž poskytuje pouze „surová data“ ve vlastním formátu založeném na XML. Programátor se tedy musí postarat o kompletní vizualizaci dat z výše uvedeného formátu. Což v sobě nese nesporné výhody – vývojář není nijak omezen a může si vše stvořit přesně podle vlastních představ. Nicméně tato výhoda je logicky i nevýhodou – kompletní tvorba vizualizace geografických dat je velice složitá a časově náročná.

V současné době existuje několik webových portálů, které jsou na těchto mapách postaveny. Obecně jsou OpenStreetMap určeny spíše pro nadšence než pro nějaké komerční nasazení. Je sice pravdou, že pokrytí je už na slušné úrovni, stále jsou ale v mapách k nalezení chyby.

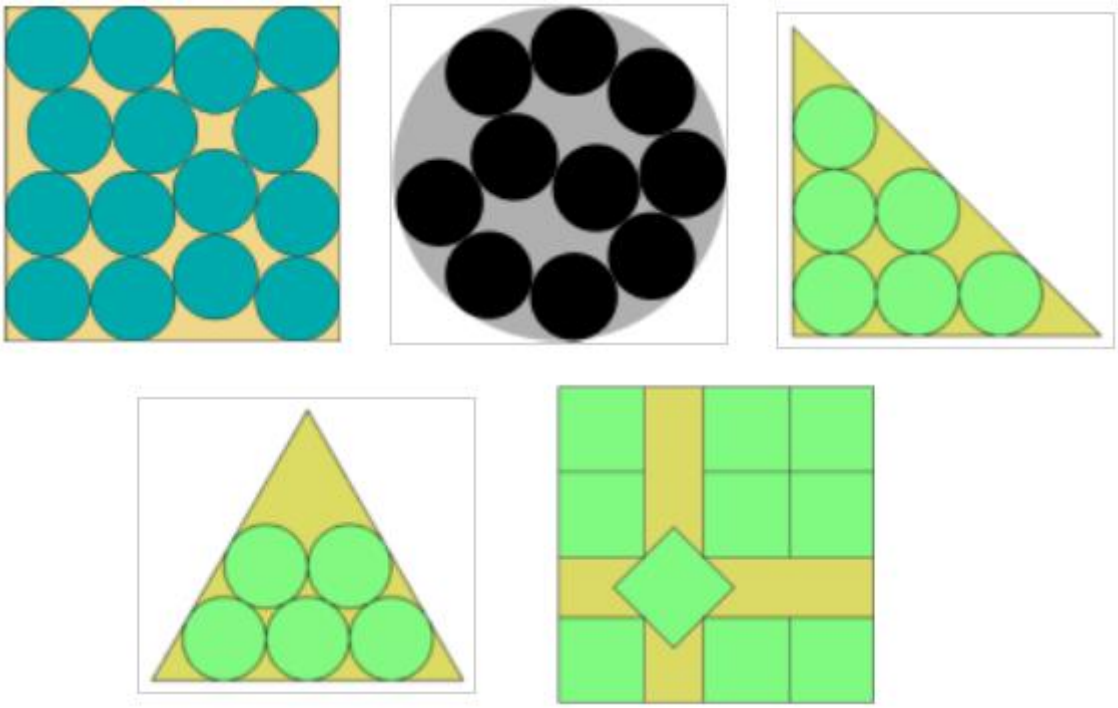
4 Container loading problem

Česky volně přeloženo jako problém naložení kontejneru. Jedná se o optimalizační proces, při kterém je definováno n objektů, které je třeba umístit do kontejneru (nákladní automobil, přepravní box apod.). Přičemž je cílem ušetřit co nejvíce místa a definované objekty rozmístit v kontejneru tak, aby vzniklo co nejméně volného prostoru. Jedná se o velice obecný problém, neboť může být definováno více kontejnerů a zároveň se problém může řešit ve více dimenzích (2D, 3D). Velmi problematické je pak řešení problému pro nestejné tvary objektů – jeden objekt je krychle, další kužel apod.

V rámci této práce bude uvažováno pouze s nakládkou ve 2D prostoru. Pro dvě dimenze lze definovat několik typů a podtypů balících algoritmů:

- balení kruhů:
    - kruhy v kruhu,
    - kruhy v obdélníku,
    - kruhy v pravoúhlém rovnoramenném trojúhelníku,
    - kruhy v obecném rovnoramenném trojúhelníku,
- balení obdélníků:
    - obdélníky v obdélníku,
    - obdélníky v kruhu,
- balení trojúhelníků:
    - stejné trojúhelníky v trojúhelníku,
    - rozdílné trojúhelníky v trojúhelníku.

Na obrázku níže je zobrazeno několik optimálních vyplnění prostoru danými obrazci. Zleva nahoře je to optimální vyplnění čtverce kruhy, vyplnění kruhu kruhy, vyplnění pravoúhlého rovnoramenného trojúhelníka kruhy, vyplnění obecného rovnoramenného trojúhelníka kruhy a nakonec vyplnění obdélníku obdélníky.



Obrázek 5 – Ukázka optimálního vyplnění obrazců[1]

[1] Zdroj: http://en.wikipedia.org/wiki/Packing_problem

5 Optimalizace SQL dotazů

Jelikož je jazyk SQL velice flexibilní, je možné více způsoby (rozdílnými dotazy) získat stejná data. Rychlost dotazů je však většinou rozdílná, mnohdy zpracování výsledku jednoho dotazu zabere mnohonásobně delší čas než zpracování výsledku jiným dotazem vracející stejná data. Proto je vhodné se zaměřit na optimalizaci dotazů. Prioritním důvodem optimalizace je snížení času potřebného pro SQL dotaz, což má přímý vliv na minimalizaci nákladů. Pokud budeme mít desítky „špatných“ (pomalých) dotazů, velice rychle narazíme na hardwarové limity a celá aplikace bude pomalá. V případě přepisu SQL dotazů na „lepší“ poběží aplikace na stávajícím hardwaru zcela bez problému, jelikož nebudou tolik vytěžovány prostředky serveru.

5.1 Obecná pravidla pro psaní SQL dotazů

Výkon databáze a SQL dotazů je přímo závislý na návrhu databázového modelu. Máme-li špatně navrženou databázi, ani sebelepší optimalizace ji zásadně nezrychlí. Proto by mělo být samotnému návrhu databáze věnováno dostatečné množství času a předem rozmyšleno, jaká data budou uložena a co se od celé aplikace požaduje. Samozřejmostí by mělo být dodržování normálních forem, ideálně třetí normální formy (3NF).

Při samotném psaní SQL dotazů lze nezávisle na zvoleném databázovém systému vymezit několik pravidel, která urychlí vykonávání dotazu:

- Vyjmenovat sloupce – ve většině případů není třeba vybírat úplně všechny sloupce z tabulky. Je vhodné vypisovat pouze ty sloupce, které opravdu potřebujeme. Tedy namísto dotazu `SELECT * FROM tabulka` použít dotaz typu `SELECT id, jmeno, prijmeni FROM tabulka`. I v případě využití všech sloupců, je vhodné je všechny vypisovat, neboť je ušetřena operace, kdy databáze zjišťuje názvy sloupců.

- Omezit použití klauzule `LIKE` – v případě tisíců a milionů záznamů je klauzule `LIKE` pomalá a hodně zatěžuje databázi. Nejkritičtější je použití tzv. full wildcard, tedy prefixu a sufixu zároveň, např. `SELECT sl1, sl2 FROM tabulka WHERE sl3 LIKE '%hodnota%'`.

- Vyhnout se klauzulím `IN` a `NOT IN` – obrovsky pomalá klauzule, především v databázovém systému MySQL. Ve většině případů lze nahradit mnohonásobně rychlejší variantou. Např. pomalý dotaz `SELECT sl1, sl2 FROM tabulka WHERE sl3 IN ('CZ', 'SK', 'UK')` přepsat na rychlejší variantu `SELECT sl1, sl2 FROM tabulka WHERE sl3='CZ' OR sl3='SK' OR sl3='UK'`.

- Používat klauze typu `LIMIT` – v případě potřeby omezeného množství řádků nezískávat z databáze úplně všechny záznamy, ale omezit jejich počet. Např. `SELECT sl1, sl2 FROM tabulka ORDER BY sl3 DESC LIMIT 10`.

- Omezit subdotazy – pokud možno nevyužívat vnořené dotazy, tzv. subquery. Jedná se o obrovský výkonnostní problém, opět především na MySQL. Namísto dotazu `SELECT sl1, sl2 FROM tabulka WHERE sl3=(SELECT sl11 FROM tabulka2 WHERE sl22='CZ')` použít spojení `JOIN` – `SELECT tabulka.sl1, tabulka.sl2 FROM tabulka JOIN tabulka2 ON tabulka.sl3=tabulka2.sl22 WHERE tabulka2.sl22='CZ'`.

- Na začátek dávat obecnější podmínky – v klauzuli `WHERE` dát na první místo nejvíce obecnou podmínku, která vyřadí nejvíce záznamů. Např. `SELECT sl1, sl2 FROM tabulka WHERE pohlavi='muž' AND vek>20`.

- Vhodný výběr pořadí spojení – v případě spojování více množin dohromady dbát na správné pořadí. Je rychlejší provést spojení menšího množství (vyfiltrovaných) řádku než spojení rozsáhlých tabulek.

- Definice a využívání indexů – provést vhodnou analýzu a na často využívané sloupce vytvořit indexy. Vhodné je například nastavit index pro sloupce, nad kterými se provádí řazení (`ORDER BY`). Nicméně je třeba brát v úvahu doporučení pro tvorbu indexů – ačkoliv budou dotazy typu `SELECT` rychlejší, vkládání (`INSERT`) bude pomalejší. Proto je třeba zvážit využití indexů pro konkrétní aplikaci.

Praktický příklad optimalizace si ukážeme na příkladu z reálného provozu. Díky spolupráci autora práce na projektu MultiShare.cz je možné prezentovat optimalizaci na milionech záznamů z reálné databáze. Jedná se o klasickou webovou aplikaci běžící na databázovém systému MySQL. Níže je uveden dotaz využívající vnořeného `SELECT`u a operátoru `IN`.

```
SELECT MS.ID, ML.ID_uzivatele, ML.datum
FROM MMS_log ML
JOIN MMS_soubory MS ON MS.ID=ML.ID_souboru
WHERE MS.ID_serveru IN (
    SELECT ID FROM MMS_download_server WHERE nazev IN(,
        'uloz.to',
        'quickshare.cz'
    )
)
```

Ve stručnosti si popíšeme databázi, resp. použité sloupce a tabulky pro lepší pochopení dotazu. Tabulka `MMS_soubory` (alias `MS`) shromažďuje informace o souborech, které byly skrze službu MultiShare.cz staženy. Tabulka `MMS_log` (alias `ML`) obsahuje historii stahování (log), kde je uchováváno ID staženého souboru, datum stažení a ID uživatele. Poslední tabulka – `MMS_download_servery` (alias `MDS`) – sdružuje názvy a další vlastnosti podporovaných serverů pro stahování. Cílem našeho dotazu je vybrat soubory z logu stahování, které byly staženy ze serverů `ulož.to` a `quickshare.cz`.

Následuje přepsaný dotaz:

```
SELECT MS.ID, ML.ID_uzivatele, ML.datum
FROM MMS_log ML
JOIN MMS_soubory MS ON MS.ID=ML.ID_souboru
JOIN MMS_download_server MDS ON MS.ID_serveru=MDS.ID
WHERE MDS.nazev='uloz.to' OR MDS.nazev='quickshare.cz'
```

V porovnání s původním dotazem byly upraveny dvě části: (i) namísto operátoru `IN` bylo použito klauzule `WHERE`, (ii) vnořený dotaz byl přepsán na spojení typu `JOIN`. Oba zmíněné dotazy vrací zcela identické záznamy, nicméně ve zcela odlišných časech. Původní dotaz byl zpracováván téměř 50 sekund, přičemž druhý (optimalizovaný) dotaz byl vykonán v mnohonásobně kratším čase. A to pouze 17,2 sekundy. Časová úspora je zde již na první pohled zcela zřejmá, po jednoduché optimalizaci dotazu jsme se dostali na třetinu původního času. Pro úplnost uvedeného měření je v tabulce níže uveden počet záznamů v jednotlivých tabulkách a jejich přibližná fyzická velikost na pevném disku.

Tabulka 1 – Mohutnost demonstrovaných tabulek

Tabulka	Počet záznamů	Typ	Velikost	Počet sloupců
MMS_log	4.485.301	MyISAM	388,9 MB	7
MMS_soubory	4.638.468	MyISAM	1,7 GB	12
MMS_download_server	36	MyISAM	14,9 kB	18

5.2 Optimalizace SQL dotazů v Oracle databázi

Databázový systém Oracle obsahuje tzv. optimalizátor (optimizer). Jedná se o integrovaný software, který vyhodnotí nejlepší postup pro zpracování zadaného SQL dotazu. Je však nutné podotknout, že aktuální verze Oracle Database 12c obsahuje velice kvalitní automatickou optimalizaci dotazů. V nejnovějších verzích Oracle Database je tedy optimalizace dotazů brána již za jakýsi přežitek, neboť vše za nás řeší software již na úrovni databáze. Nicméně techniky známé a používané v dřívějších dobách (např. verze 10g) jsou uvedeny v následujících odstavcích.

5.2.1 Detekce problematických SQL dotazů

Zásadním úkolem při optimalizaci je nalezení SQL dotazů, které způsobují snížení výkonu databáze, a tedy celé aplikace. Takovéto dotazy je nezbytné analyzovat a provést opatření, která zamezí nadměrnému zatěžování prostředků serveru. Pro nalezení „problémových“ dotazů lze využít několik způsobů:

- V$SQLAREA – interní seznam statistik databáze Oracle. Pro uživatele se jeví jako klasická tabulka obsahující informace (statistiky) o dotazech.
- Automatic Database Diagnostic Monitor (ADDM),
- Automatic Workload Repository,
- SQL Trace.

5.2.2 Trace nástroje

Databáze poskytuje nástroje pro monitorování a analýzu aplikací, které se k databázi připojují:

- Oracle Trace – sbírá informace o výkonu, využití dat apod. Obsahuje celkem tři části: (i) Oracle Trace Manager, což je klient-server aplikace s grafickým prostředím shromažďující informace z předdefinovaných událostí, (ii) Oracle Trace Collection Services, (iii) Oracle Trace Application Programming Interface (API).

- Oracle Autotrace – povolení/zakázání automatického zobrazování exekučního plánu (viz dále) a statistik pomocí příkazu `SET AUTOTRACE ON/OFF`. Statistiky obsahují např. informace o počtu bloků načtených z disku (physical read), počet bajtů přenášených ke klientovi (Bytes sent via SQL*Net to klient), počet zpracovávaných řádků (rows processed) nebo počet třídících operací (sorts).

5.2.3 Exekuční plány

Při každém zpracovávaném dotazu si databázový systém vytvoří tzv. exekuční plán. Ten pomůže rozhodnout, jak se daný dotaz vykoná – v jakém pořadí budou dílčí operace dotazu provedeny, který index bude zvolen, jaké spojení tabulek bude použito apod. Níže uvedený obrázek zobrazuje grafické znázornění exekučního plánu v prostředí programu Oracle JDeveloper. Je možné vysledovat, že v dotazu bude prvně použita tabulka `TBL_C`, pro spojení tabulek `TBL_C` a `TBL_B` bude využito spojení `HASH JOIN` atd.

```
select * from tbl_a, tbl_b,tbl_c where
tbl_a.join_column_a=tbl_b.join_column_b and
tbl_c.id_c=tbl_b.id_b and
id_c >(select 24 from dual);
```

Script Output | Query Result | Explain Plan

SQL | 0 seconds

OPERATION	OBJECT_NAME	OPTIONS	COST	CARDINALITY
SELECT STATEMENT			9	5
HASH JOIN			7	5
Access Predicates				
TBL_A.JOIN_COLUMN_A=TO_NUMBER(TBL_B.JO				
type="db_version"				
11.2.0.2				
HASH JOIN			5	5
Access Predicates				
TBL_C.ID_C=TBL_B.ID_B				
TABLE ACCESS	TBL_C	FULL	2	5
Filter Predicates				
ID_C> (SELECT 24 FROM SYS.DUAL DUAL				
FAST DUAL			2	1
TABLE ACCESS	TBL_B	FULL	2	100
TABLE ACCESS	TBL_A	FULL	2	100

Obrázek 6 – Exekuční plán v prostředí Oracle JDeveloper

Každý exekuční plán umožňuje zobrazit několik důležitých informací. Na obrázku výše jsou zobrazeny informace `OPERATION`, `OBJECT_NAME`, `OPTIONS`, `COST` a `CARDINALITY`. V následující tabulce je uveden seznam vybraných informací, které nám dovoluje exekuční plán zobrazit.

Tabulka 2 – Sloupce vypisované v exekučním plánu

Informace	Popis
OPERATION	Název operace prováděné v daném kroku.
COST	Cena dané operace. Cena je bezrozměrná hodnota, slouží pouze k porovnání jednotlivých operací.
CARDINALITY	Předpokládaný počet řádků, ke kterým operace bude přistupovat.
BYTES	Předpokládaný počet bajtů, které bude operace číst.
CPU_COST	Předpokládané vytížení procesoru.
IO_COST	Předpokládané náklady na I/O operace.
OBJECT_NAME	Název objektu, se kterým se v daném kroku pracuje.

Informace `OPERATION` může obsahovat mnoho typů záznamů, proto je pro úplnost v tabulce níže uveden seznam vybraných hodnot, kterých může `OPERATION` nabývat. Opět se stručným popisem.

Tabulka 3 – Možné hodnoty sloupce OPERATION

Název	Typ	Popis
TABLE ACCESS	FULL	Přístup ke všem řádkům tabulky.
TABLE ACCESS	SAMPLE	Přístup k části řádků z tabulky.
TABLE ACCESS	SAMPLE BY ROWID RANGE	Přístup k části řádků z tabulky na základě rozsahu ROWID.
INDEX	UNIQUE SCAN	Přístup k jednomu řádku z indexu.
INDEX	FULL SCAN	Přístup ke všem řádkům z indexu.
HASH JOIN		Spojení dvou skupin řádků do jedné tabulky.
COUNT		Počet řádků v dané tabulce.
SORT	JOIN	Třídění skupiny řádků pomocí MERGE-JOIN
MERGE JOIN		Setřídění a následné spojení dvou skupin řádků.

Oracle Database umožňuje použití tzv. hintů (česky tip, nápověda). Jedná se pomocné informace pro optimalizátor, který je použije pro sestavení exekučního plánu. Jsou součástí zadávaného SQL dotazu a přímo ovlivňují exekuční plán. Pomocí hintů lze specifikovat přístupové cesty pro tabulku, určit pořadí spojení `JOIN`, metody, které budou při operaci `JOIN` použity nebo nastavit optimalizační parametry.

```
SELECT /*+ ALL_ROWS */ employee_id, last_name, salary, job_id
FROM employees
WHERE employee_id = 123;
```

Výše uvedený příklad zobrazuje použití hintu `ALL_ROWS`, na základě kterého je použit cost-based přístup bez ohledu na existenci statistik.

5.3 Optimalizace SQL dotazů v MySQL

Databázový systém MySQL obsahuje – podobně jako databáze Oracle – optimalizátor, který je zodpovědný za návrh nejefektivnějšího způsobu dosažení výsledku. Nicméně je

nutné podotknout, že optimalizátor počítá pouze se základními vlastnostmi a neumožňuje nijak široké využití. Reálné srovnání s výše popisovaným optimalizátorem Oraclu je téměř nemožné.

Optimalizátor při rozhodování uvažuje různé faktory – velikost tabulek, index, sloupce dotazu apod. Proto je nutné mít vždy aktuální a přesné informace o stavu jednotlivých tabulek – statistiky. Pro vybraná úložiště (MyISAM, InnoDB a BDB) existuje příkaz `ANALYZE TABLE`, který statistiky vypočítá a uloží. Příkaz `OPTIMIZE TABLE` dokáže tabulky defragmentovat a obnoví strukturu dat a indexů. Používá se v případě mazání nebo úpravy větší části tabulky.

5.3.1 EXPLAIN

Databáze MySQL poskytuje příkaz `EXPLAIN`, který programátorovi přiblíží proces zpracovávání dotazu. Syntaxe je velice snadná, stačí pouze před požadovaný dotaz vepsat klíčové slovo `EXPLAIN`, např. `EXAPLIN SELECT * FROM tabulka`. Výstup takového dotazu je uveden na následujícím obrázku.

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	MDS	range	PRIMARY,nazev	nazev	152	*NULL*	2	Using where
1	SIMPLE	MS	ref	PRIMARY,ID_serveru	ID_serveru	1	multishare.MDS.ID	109561	Using where
1	SIMPLE	ML	ref	ID_souboru	ID_souboru	4	multishare.MS.ID	2	

Obrázek 7 – Výsledek dotazu EXPLAIN v prostředí phpMyAdmin

Jednotlivé sloupce výše uvedené tabulky představují:

- id – číslo kroku,
- select_type – druh zpracovávaného příkazu `SELECT`,
- table – název tabulky,
- type – druh spojení (viz tabulka níže),
- possible_keys – potenciální klíče, které jsou k dispozici pro spojení,
- key – klíč vybraný optimalizátorem,
- key_len – délka vybraného klíče,
- ref – hodnoty nebo klíče, které budou předány do vybraného klíče,
- rows – nejlepší odhad optimalizátoru na počet řádků, které mají být vyhledané v tabulce pro tento dotaz,

- Extra – další doplňující informace. Může nabývat hodnot např. `Using index`, což značí, že výsledek byl vytvořen přímo z indexu, `Distinct` značí vybrání pouze jednoho řádku při shodě sloupců nebo `Using temporary` označuje využití dočasné tabulky.

V tabulce níže je uvedeno několik vybraných typů spojení (sloupec `type` v dotazu `EXPLAIN`).

Tabulka 4 – Vybrané typy spojení tabulek

Typ	Popis
ALL	Prohledávání všech řádku tabulky.
index	Prohledávání celého indexu.
range	Prohledávání rozsahu indexu.
system	Tabulka má pouze jeden řádek.
ref	Několik řádků tabulky vyhovuje indexu.

5.3.2 Ovlivnění optimalizátoru

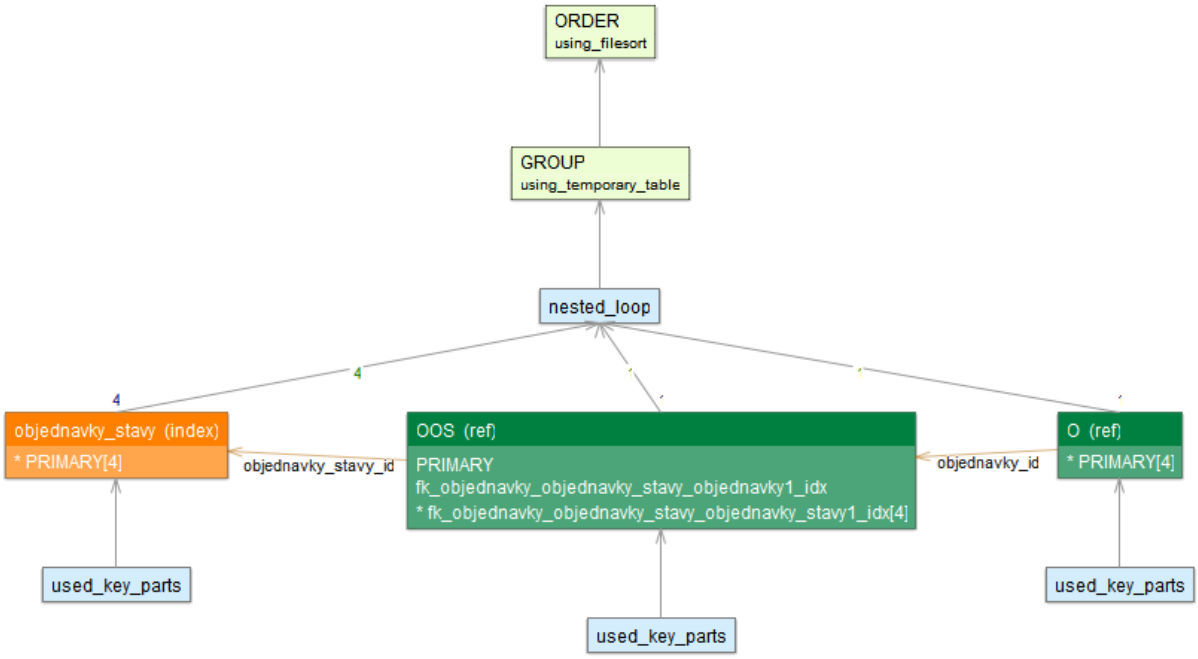
Podobně jako lze u databáze Oracle ovlivnit výběr exekučního plánu pomocí hintů, v MySQL existuje jakási zjednodušená obdoba. K dispozici je například příkaz `STRAIGHT_JOIN`, který vnutí optimalizátoru použít spojení `JOIN` v přesně zapsaném pořadí. `SQL_BIG_RESULTS` optimalizátoru napoví, že se jedná o veliký soubor dat, načež se na pevném disku vytvoří dočasná tabulka pro zpracování dat. Napovídat lze i v případě indexů, příkaz `FORCE INDEX` vynutí použití indexu, naopak při použití `IGNORE INDEX` nebude index využit. Mezi další nápovědy optimalizátoru lze jmenovat: `SQL_BUFFER_RESULTS`, `USE_INDEX`, `HIGH_PRIORITY`, `SQL_SMALL_RESULT`, `SQL_BUFFER_RESULT`, `SQL_CACHE`, `SQL_NO_CACHE` nebo `SQL_CALC_FOUND_ROWS`.

5.3.3 Slow query log

Jak již samotný název funkcionality napovídá, jedná se o log pomalých dotazů. Tato funkce neumožňuje žádné ladění ani detekci konkrétních dotazů, jedná se o čistě monitorovací činnost. V nastavení MySQL lze definovat proměnnou `long_query_time`. Pokud jakýkoliv spuštěný dotaz překročí definovaný čas, bude uveden v logu. Tímto způsobem lze odhalit dlouho vykonávané dotazy. Nicméně se nejedná o zcela stoprocentně spolehlivý nástroj, jelikož čas vykonávání dotazu není závislý pouze na databázovém systému, ale na celém systému jako celku. Snadno lze tento čas ovlivnit zcela jinou – na databázi nezávislou – aplikací, která vytěžuje server a na MySQL se nedostávají prostředky.

5.3.4 Visual EXPLAIN

Novinkou v MySQL 5.6 je podpora pro `Visual EXPLAIN`. Jedná se v podstatě o grafické znázornění dotazu, jakási grafická nadstavba pro klasický příkaz `EXPLAIN`. V kombinaci se softwarem MySQL Workbench lze tímto příkazem poměrně přehledně vyobrazit zpracování dotazu. Jak takový výstup vypadá lze zhlédnout na obrázku níže.



Obrázek 8 – Ukázka vizuálního zobrazení VISUAL EXPLAIN

5.4 Optimalizace vybraných SQL dotazů

V rámci celé aplikace jsou používány především jednodušší dotazy, které získávají data pouze z jedné nebo dvou tabulek. Takové dotazy není třeba optimalizovat, neboť na nich nelze prakticky nic zkazit a žádná optimalizace není nutná. Nicméně je v aplikaci i několik složitějších dotazů – ty budou níže rozebrány, popsány a hlavně optimalizovány.

5.4.1 Dotaz č. 1

První rozebíraný dotaz shrnuje denní přehled jízd – pro každý den zobrazí počet použitých automobilů a počet rozvezených objednávek. Výsledná tabulka dotazu může vypadat například takto:

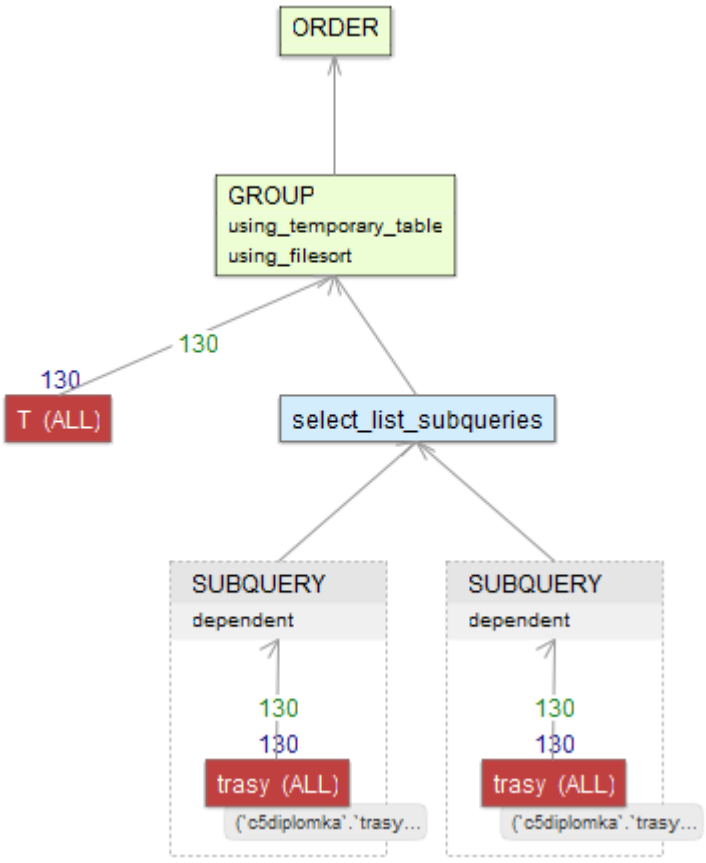
Tabulka 5 – Příklad výstupu dotazu č. 1

den	pocet_objednavek	pocet_automobilu
2013-07-24	26	4
2013-07-23	16	3
2013-07-22	20	3
2013-07-21	17	3
2013-07-20	19	3
2013-07-19	14	3
2013-07-18	18	3

Původní dotaz, který byl vytvořen pro získání výše uvedených dat, byl následující:

```
SELECT
    T.den,
    (SELECT COUNT(*) FROM trasy WHERE den=T.den) AS pocet_objednavek,
    (SELECT COUNT(DISTINCT automobily_id) FROM trasy WHERE den=T.den) AS
pocet_automobilu FROM trasy T
GROUP BY den
ORDER BY den DESC
```

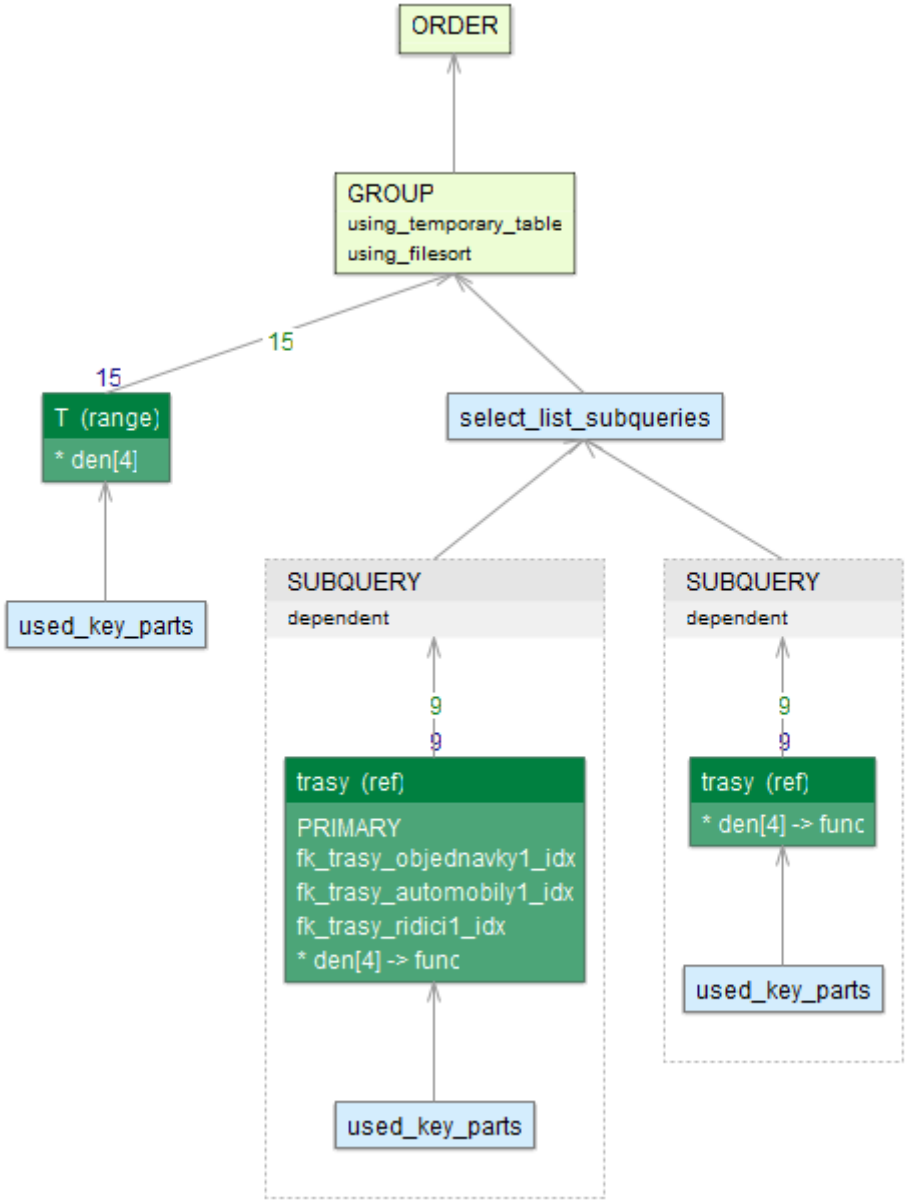
Pomocí programu MySQL WorkBench 5.2 CE byla provedena vizualizace dotazu, tzv. `VISUAL EXPLAIN` (viz předchozí kapitola). Výsledek grafického znázornění je vidět na následujícím obrázku.



Obrázek 9 – Vizualizace původního dotazu č. 1

Obrázek je nutné procházet odspodu, tj. od „první části“. Na první pohled je zřejmé, že je prováděn plný přístup k tabulce `trasy`, a to hned dvakrát. Vždy je vráceno 130 záznamů, tedy celý obsah tabulky. Po vykonání vnořených dotazů je opět přistupováno k tabulce `trasy` (alias `T`), přičemž je obdobně vráceno 130 záznamů – celá tabulka. V konečné fázi se provede seskupení (`GROUP`) nad vybranými daty a seřazení.

Původní dotaz je velice neefektivní, neboť prochází všechna data jedné tabulky hned třikrát. Nabízí se použití indexů – primární klíč pro `trasy_id` a cizí klíče pro sloupce `objednavky_id`, `automobily_id` a `ridici_id`. Po vytvoření indexů opět provedeme `VISUAL EXPLAIN`.



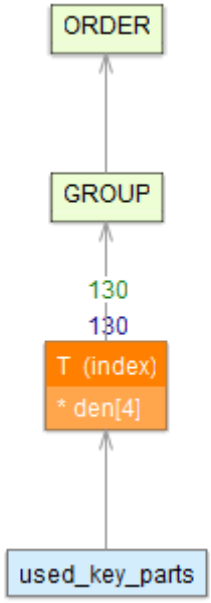
Obrázek 10 – Vizualizace dotazu č. 1 po přidání indexů

Z obrázku je patrné úspornější řešení – ačkoliv se k tabulce `trasy` přistupuje stále třikrát, nejsou již přenášena všechna data (celá tabulka), nýbrž jen část. Všechny přístupy využívají přístupu podle klíče. V případě vnořených dotazů je vráceno jen 9 řádků tabulky – namísto původních 130. I v případě přístupu k aliasu `T`, je patrná úspora – vráceno je pouze 15 řádků z tabulky.

Ačkoliv je uvedený výsledek již poměrně solidní, rozhodně není ideální, aby bylo třikrát přistupováno ke stejné tabulce. Z obecných pravidel pro optimalizaci dotazů je zřejmé, že vnořené dotazy by šlo nahradit něčím jiným. Proto je celý dotaz upraven do následující podoby.

```
SELECT
    den,
    COUNT(*) AS pocet_objednavek,
    COUNT(DISTINCT automobily_id) AS pocet_automobilu
FROM trasy T
GROUP BY den
ORDER BY den DESC
```

Upravený dotaz je velice podobný tomu původnímu a samozřejmě vrací zcela identické výsledky. Hlavní změnou je odstranění vnořených dotazů, čímž by mělo dojít k razantnější úspoře výkonu. Opět následuje `VISUAL EXPLAIN`.



Obrázek 11 – Vizualizace přepsaného dotazu č. 1

Na první pohled je patrné zjednodušení celého procesu. Není třeba vykonávat několik (vnořených) dotazů, k tabulce `trasy` se přistupuje již pouze jednou a díky aplikaci indexu je přístup rychlejší. Nicméně je třeba podotknout, že je opět vráceno celých 130 řádků, tedy celá tabulka.

Nabízí se tedy otázka, zda je přepsaný dotaz výkonnější než původní dotaz s přidanými klíči. Otestování je snadné – změříme čas provádění dotazu. Z toho nám jasně vyjde výkonnější dotaz nový – přepsaný. Ten je na serveru prováděn 0,004 sekundy, tedy 4 ms. Naproti tomu původní dotaz s přidanými klíči se prováděl 0,0107 sekundy (10,7 ms). V konečném důsledku tedy můžeme tvrdit, že přepsaný dotaz s přidanými indexy je optimalizovaný.

5.4.2 Dotaz č. 2

Druhý přehledový dotaz zobrazuje seznam řidičů a k nim počet rozvezených objednávek a počet dní, které daný řidič odjezdil. Zároveň je aplikováno omezení, které počítá pouze s objednávkami do České a Slovenské republiky. Pro zjednodušení je v dotazech uváděno přímo ID státu (1=ČR, 2=SR). Výsledný výstup může vypadat následovně.

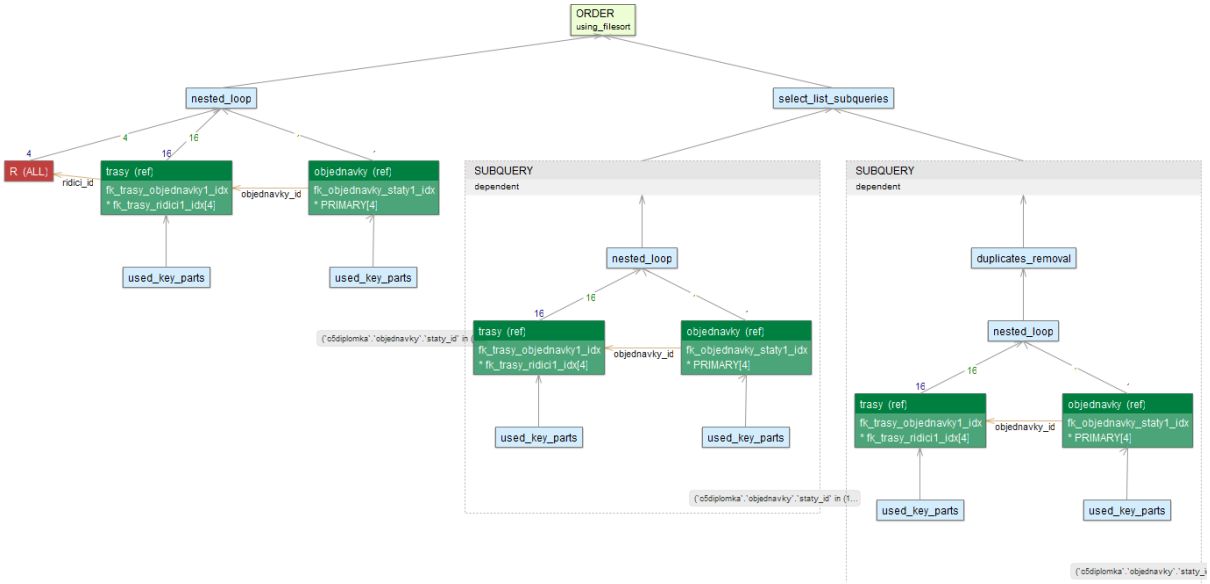
Tabulka 6 – Příklad výstupu dotazu č. 2

ridic	pocet_objednavek	pocet_dni
František Březina	36	7
Michal Trnovský	4	1
Mirek Macenovský	41	7
Vladimír Mrkváč	42	7

Původní dotaz je napsán s ohledem na pozdější maximální optimalizaci, proto je psán odstrašujícím způsobem. V databázi a jednotlivých tabulkách jsou již uvažovány indexy, proto bude tento dotaz optimalizován pouze po stránce formulace.

```
SELECT
    CONCAT(jmeno, ' ', prijmeni) AS ridic,
    (SELECT COUNT(*) FROM objednavky
        WHERE staty_id IN(1,2) AND objednavky_id IN (
            SELECT objednavky_id FROM trasy WHERE ridici_id=R.ridici_id
        )) AS pocet_objednavek,
     (SELECT COUNT(DISTINCT den) FROM trasy
        WHERE ridici_id=R.ridici_id
        AND objednavky_id IN (
            SELECT objednavky_id FROM objednavky WHERE staty_id IN(1,2)
        )) AS pocet_dni

FROM ridici R
WHERE ridici_id IN (
    SELECT ridici_id FROM trasy WHERE objednavky_id IN (
        SELECT objednavky_id FROM objednavky WHERE staty_id IN (1,2)
    ))
ORDER BY ridic
```

Dotaz je z hlediska optimalizace i přehlednosti doslova otřesný. Stejně tak grafické zobrazení pomocí `VISUAL EXPLAIN` zobrazuje odstrašující případ.



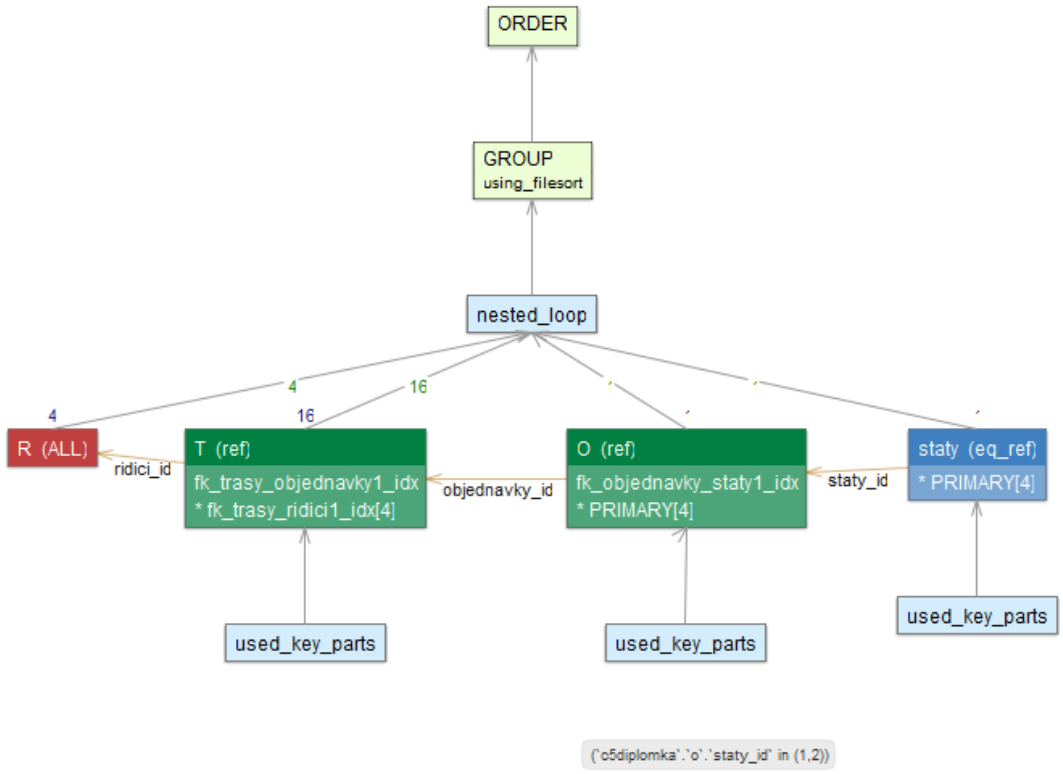
Obrázek 12 – Vizualizace původního dotazu č. 2

Zcela katastrofální jsou vnořené dotazy, kterých je v dotazu veliké množství. Situaci „zachraňuje“ přítomnost indexů, které alespoň částečně eliminují hrozivě sestavený dotaz.

Potřeba optimalizace je zde zcela nevyhnutelná. Dotaz lze poměrně elegantním způsobem přepsat a zcela se zbavit vnořených dotazů a nevhodných operátorů IN.

```
SELECT
    CONCAT(jmeno, ' ', prijmeni) AS ridic,
    COUNT(DISTINCT objednavky_id) AS pocet_objednavek,
    COUNT(DISTINCT den) AS pocet_dni
FROM
    trasy T
    RIGHT JOIN ridici R USING (ridici_id)
    LEFT JOIN objednavky O USING (objednavky_id)
    LEFT JOIN staty USING (staty_id)
WHERE staty_id IN (1,2)
GROUP BY ridic
ORDER BY ridic
```

Grafické znázornění dotazu pomocí `VISUAL EXPLAIN` zobrazuje následující obrázek.



Obrázek 13 – Vizualizace upraveného dotazu č. 2

Ačkoliv náročnost – počet získávaných řádků – zůstal díky indexům stejný, enormně se snížil počet tabulek, ke kterým je přistupováno. V původním dotazu byla data získávána pomocí osmi přístupů k daným tabulkám, v optimalizované verzi jsou to již pouze čtyři přístupy. Optimalizace je zde tedy razantní. Pro nový dotaz samozřejmě hovoří i časy pro zpracování. První verze dotazu byla serverem zpracována za 0,025 sekundy (25 ms), kdežto optimalizovaná verze za 0,010 sekundy (10 ms). Opět můžeme prohlásit, že jsme provedli úspěšnou optimalizaci dotazu.

6 Vlastní řešení IS přepravní společnosti

Pro potřeby této práce byla vytvořena konkrétní aplikace demonstrující vlastnosti a požadavky na IS přepravní společnosti. V aktuální kapitole je aplikace podrobně rozebrána, popsána a je vysvětleno, proč jsou dané úlohy řešeny právě uvedeným způsobem. Kompletní zdrojové kódy aplikace a obsah databáze jsou dostupné na CD, které je součástí práce. Popisovaná aplikace je zároveň dostupná na URL adrese `http://diplomka.youcan.cz`.

6.1 Použité technologie

6.1.1 PHP

PHP je zkratkou „Hypertext Preprocessor“, v češtině překládáno jako hypertextový preprocesor. Jedná se o skriptovací programovací jazyk, který je určený pro generování dynamických internetových stránek, obvykle ve formátu HTML a XHTML. Jazyk pracuje na straně serveru, uživateli (klientovi) je odesílán pouze výstup skriptu. Výhodou je tedy skrytý kód pro uživatele – na rozdíl třeba od JavaScriptu.

Syntaxe jazyka je inspirována několika programovacími jazyky, konkrétně je to Perl, C, Pascal a Java. Velikou výhodou jazyka PHP je platformní nezávislost, je možné jej tedy spouštět prakticky v libovolném operačním systému. Ačkoliv existuje několik málo funkcí a drobných odlišností v závislosti na operačním systému, většinou je migrace na jiný operační systém bezproblémová a nevyžaduje žádné zásadnější změny v kódu.

PHP patří mezi nejrozšířenější jazyky používané pro tvorbu webových stránek, je dostupný prakticky v každé nabídce hostingových společností. Velice oblíbené je použití tzv. LAMP serveru, tedy spojení Linuxu (operační systém), Apache (webserver), MySQL (databáze) a právě PHP jako skriptovacího jazyka. Výše zmíněná kombinace je masově rozšířená kvůli své jednoduchosti instalace, bezpečnosti a v neposlední řadě také díky své ceně – celé softwarové řešení je zcela zdarma.

K oblibě tohoto jazyka přispěla velikou měrou také jeho univerzálnost. Existuje celá řada knihoven rozšiřující základní „dovednosti“ jazyka PHP. Jmenovitě např. práce s grafikou – tzv. GD knihovna umožňující základní grafické operace jako rotace, ořez, zrcadlení atd. Velice oblíbenými knihovnami jsou dále třeba zlib umožňující práci s komprimovanými soubory. Knihovna PDFlib, jak již název napovídá, usnadní práci se soubory typu PDF apod. Dnes již neodmyslitelnou součástí jsou knihovny pro práci s databází. Je podporováno mnoho typů – běžně používané jako je MySQL, MSSQL, Oracle, nebo PostgreSQL. Ale také méně často používané databáze jako dBase, Sybase, Mongo, IBM DB2 nebo třeba SQLite.

PHP 5.3

Jedná se verzi, která byla vydána 30. června 2009 a která přinesla mnoho zásadnějších změn a také relativně mnoho problémů. Mezi hlavní novinky můžeme jmenovat podporu

jmenných prostorů, díky kterým můžeme aplikaci dělit na podprostory – jako je tomu u vyspělejších jazyků typu C# apod. Dále přibyla podpora anonymních funkcí, která výrazně přispívá přehlednosti a jednoduchosti kódu.

Jak již bylo zmíněno, s nástupem nové verze nastalo i množství problémů. Ve verzi 5.3 bylo mnoho funkcí označeno jako zastaralých (deprecated). Problém nastával, když hostingové společnosti prováděli update svých serverů a přešli na verzi PHP 5.3. Mnohé starší weby používaly právě ony zastaralé funkce, a proto docházelo k nefunkčnosti stránek.

6.1.2 MySQL

Jedná se o velmi populární multiplatformní databázový systém. V minulosti jej vyvíjela a vlastnila švédská společnost MySQL AB, v současnosti je produkt vlastněn a vyvíjen společností Oracle, která MySQL odkoupila. Svoji popularitu si tento databázový systém získal především díky své snadné implementovatelnosti – lze jej instalovat na většinu dnes běžně používaných operačních systémů. A přestože se jedná o produkt s bezplatnou licencí GPL, je databáze poměrně schopná a výkonná. Od počátku bylo MySQL optimalizováno především pro rychlost, což se projevilo na určitých zjednodušeních – např. podporuje pouze základní způsoby zálohování a ještě donedávna nebyly podporovány mnohé pokročilé funkcionality (triggery, pohledy atd.). Nicméně i tyto nedostatky jsou v posledních verzích programu řešeny, tudíž můžeme s klidným svědomím označit MySQL za moderní a plnohodnotný databázový systém. Jak již samotný název napovídá, komunikace probíhá prostřednictvím SQL jazyka (dotazů).

Pro ukládání dat využívá MySQL několik různých typů enginů. Mezi nejpoužívanější a nejznámější patří následující.

- MyISAM – jeden z nejstarších typů úložiště, fyzicky využívá dvou souborů na pevném disku (MYD – datový soubor a MYI – indexový soubor). Do verze 5.1 byl tento engine využíván jako výchozí typ.
- InnoDB – výchozí engine pro novější verze MySQL. Jeho návrh byl od počátku zaměřen pro zpracování transakcí. Na rozdíl od MyISAM podporuje cizí klíče, na druhou stranu je o něco málo pomalejší v porovnání s MyISAM. Nevýhodou je absence klíčů pro fulltextové vyhledávání[2].
- CSV – ukládání dat v textovém souboru formátu CSV (text oddělený čárkami).
- MEMORY – engine využívající pouze prostoru operační paměti počítače. Jedná se o vysoce výkonný typ, nicméně při restartu serveru jsou data ztracena.
- ARCHIVE – uzpůsoben pro uchovávání velikého množství neindexovaných dat.

[2] FULLTEXTový index byl přidán do nejnovější verze MySQL 5.6

Verze

Aktuální stabilní verzí – ke dni 27. 6. 2013 – je MySQL Community Server 5.6.12, přičemž veřejně dostupná vývojová (development) verze je 5.7.1 m11. Níže jsou uvedeny důležité verze MySQL s podstatnými a „revolučními“ novinkami.

- verze 3.23 – u enginu InnoDB je podpora pro transakce a cizí klíče,
- verze 4.0 – podpora příkazu UNION,
- verze 4.1 – podpora různých datových sad a porovnání na úrovni databáze, tabulky i sloupce, podpora pro vnořené dotazy, R-stromy pro engine MyISAM,
- verze 5.0 – možnost vytvořit procedury, triggery a pohledy, podpora pro kurzory,
- verze 5.1 – možnost fyzického dělení rozsáhlých tabulek do menších (partition), časování událostí (Event Scheduler), přidáno úložište IBM DB2,
- verze 5.5 – výchozím úložištěm je InnoDB, podpora rozšířeného Unicode kódování (utf16, utf32), značné vylepšení funkce partitioning, podpora IPv6,
- verze 5.6 – podpora FULLTEXT indexu pro InnoDB, vylepšení optimizéru (hlavně pro subdotazy), nový příkaz VISUAL EXPLAIN.

6.1.3 JavaScript

JavaScript je multiplatformní interpretovaný programovací jazyk. Původně byl stvořen pro implementaci do webových prohlížečů, kde dodnes slouží jako skriptovací jazyk na straně klienta pro interakci s uživatelem, ovládání okna prohlížeče, asynchronní komunikaci apod. V současné době je JavaScript využíván i mimo webové prostředí v klasických desktopových aplikacích. Jako příklad uveďme integraci v dokumentech PDF.

Jak již bylo nastíněno, jedná se o jazyk na straně klienta. V prvotní fázi je tedy zdrojový kód přenesen ze serveru k uživateli, následně je tento kód vykonán. Logicky to s sebou nese obrovskou nevýhodu pro vývojáře – jejich „výtvor“ (kód) je veřejně dostupný a snadno kopírovatelný. Pro koncového uživatele (konzumenta webové stránky) je JavaScript velmi bezpečný, neboť má poměrně přísné bezpečnostní limity. Zcela zásadním omezením je nemožnost manipulovat se soubory na pevném disku. Obecně se jedná o moderní programovací jazyk podporující objektově orientované programování, regulární výrazy nebo třeba výjimky.

6.1.4 Nette

Jedná se o framework pro tvorbu webových aplikací v PHP 5. Hlavní zaměření frameworku je eliminace chyb, jednoduchost, přehlednost a znovu použitelnost zdrojového kódu. Využívá událostmi řízené programování, z části je založen na používání komponent a je postaven na softwarové architektuře MVC (model-view-contoller). Celý projekt je dostupný pod licencí GNU GPL a zajímavou informací je fakt, že se jedná o framework

vytvořený Čechem. Ačkoliv je používání Nette podmíněno požadavky na hosting, jsou velmi nízké a dá se říct, že jej lze rozběhnout na každém dnes nabízeném hostingu.

Podobně jako u většiny frameworků i u Nette je nutné před samotným prvním vývojem nastudovat dokumentaci a seznámit se, jak to celé funguje. Start pro samotné využívání Nette je poměrně složitý a naučit se všechny jeho zákonitosti zabere dost času. Nicméně po delším zkoumání a proniknutí do tohoto frameworku zjistíme, že se jedná o velice mocný nástroj, který programátorovi v lecčem usnadní a hlavně urychlí vývoj. V následujících podkapitolách je uvedeno několik zajímavých funkcionalit frameworku Nette.

Zabezpečení

Prvotní věcí, na kterou by se měl každý vývojář zaměřit, je bezesporu bezpečnost aplikace. Zkrátka učinit opatření vedoucí k vniknutí do systému. Samotné Nette rozhodně nezabrání vniknutí do špatně postavené aplikace, nicméně pomáhá mnohými technikami k eleminaci bezpečnostních děr. Nette podporuje automatickou ochranu před následujícími útoky.

- XSS (Cross-Site Scripting) – metoda narušení webových stránek zneužívající neošetřených výstupů. Útočník touto technikou dokáže do stránky podstrčit svůj vlastní kód a tím může stránku pozměnit nebo dokonce získat citlivé údaje o návštěvnících.
- CSRF (Cross-Site Request Forgery) - útok spočívající v přimění uživatele navštívit stránku, která skrytě vykoná útok na webovou aplikaci, kde je aktuálně přihlášen. Lze takto například pozměnit nebo smazat článek, aniž by si toho uživatel všiml.
- URL attack – podstrčení aplikaci škodlivý vstup.
- Session hijacking – utočník může podstrčit uživateli svoje session ID a díky tomu získá do aplikace přístup.

Ladící nástroje

Obecně je ladění v programovacím jazyce PHP nepříliš dobře podporováno. Framework Nette obsahuje tzv. laděnku, což je ladící nástroj umožňující rychle odhalit a opravit chyby. Podporuje logování, vypisování proměnných a také měří čas (nejen) skriptu. Základem je tzv. Debugger Bar, což je malá lišta zobrazující rychlost načtení stránky, využití paměti a jiné informace, které je možné rozšířit pomocí dodatečných doplňků, popř. o vlastní kód.

Velice silnou stránkou je vizualizace chyb a výjimek. Samotné PHP se omezuje pouze na strohý výpis chyby typu `Parse error:  syntax error, unexpected T_STRING in soubor.php on line 15`. S laděnkou dostává vývojář přehledné barevné rozhraní s výpisem chyby. Zároveň na první pohled zjistí, kde chyba vznikla, neboť je zobrazen zdrojový kód skriptu. Programátor může aktivovat i tzv. Strict Mode hlídající existenci

proměnných apod. Lze se tak velice snadno vyhnout problémům s překlepem v názvu proměnné.

Framework Nette pracuje ve dvou režimech – produkční a vývojový. Pokud jsme ve vývoji, zobrazuje se laděnka s informacemi a vypisují se chyby. Pokud je ale již aplikace nasazena v ostrém provozu, laděnka se samozřejmě nezobrazí, stejně tak i chyby. Uživateli webových stránek se pouze zobrazí informace, že je něco špatně a chyba se uloží do logu. Zároveň je možné nastavit automatické odeslání emailu v případě výskytu takovéto chyby.

Šablonovací systém

Šablona je ve své podstatě HTML stránka rozšířená o podporu speciálních konstrukcí. Z pohledu MVC modelu je šablona písmeno V, tedy view. Šablonovací systém obsažený ve frameworku Nette, nese název Latte. Níže je uveden běžný HTML kód s částmi PHP kódu, který vypíše nečíslovaný seznam (HTML tag <li>) z proměnné $polozky.

```
<?php if ($polozky) { ?>
    <?php $pocitadlo = 1; ?>
    <ul>
    <?php foreach ($polozky as $polozka) { ?>
        <li id="item-<?php echo $pocitadlo++ ?>"><?php
        echo(htmlSpecialChars(strtoupper($polozka))); ?>
        </li>
    <?php } ?>
    </ul>
<?php } ?>
```

Stejného výsledku dosáhneme pomocí Latte následujícím kódem.

```
<ul n:if="$polozky">
{foreach $polozky as $polozka}
    <li id="item-{$iterator->counter}">{$polozka|upper}</li>
{/foreach}
</ul>
```

V rámci Latte šablon jsou definovány dva základní pojmy – makra a helpery. Makra jsou zjednodušeně řečeno vnořené kusy PHP kódu. V předchozím příkladu je makrem např. {foreach}, což je obdobou příkazu foreach v PHP. Maker existuje celá řada, můžou to být podmínky (if), cykly (for, foreach, while), definice proměnných nebo speciální makra pro definici tzv. bloků. Určitou část kódu můžeme pojmenovat a následně ji vložit zcela na jiném místě. Vkládání bloků je umožněno nejen v rámci jednoho souboru, nýbrž všemi šablonami. Dokonce lze uplatnit i funkce dědění šablon, což je funkcionalita známá z metod objektově orientovaného programování.

Druhým pojmem jsou helpery, což by se dalo do češtiny přeložit jako pomocníci. Pokud za proměnnou v Latte uvedeme znak | (roura) a následně název helperu, provede se funkce helperu s parametrem proměnné. Z výše uvedeného příkladu můžeme uvést kód

`{$polozka|upper}`, což je ekvivalent v PHP pro `strtoupper($polozka)`. Podobně jako u maker, helperů existuje celá řada a je možné definovat zcela vlastní.

V rámci šablon je možné uvádět makro pro překlad do jiného jazyka, např. `{_}Univerzita Pardubice{/_}`. Takto označený text se ve výchozím nastavení zcela standardně vypíše, nicméně lze použít nástroj pro překlad, tzv. translator. Tento doplněk najde všechny lokalizační makra a umožní texty přeložit do jiného jazyka. Pokud bude překlad k dispozici, může se např. pro angličtinu namísto `Univerzita Pardubice` vypsat přeložený text – `University of Pardubice`.

Formuláře

Při běžném programování formulářů musí vývojář napsat spousty HTML tagů popisujících požadovaný formulář. Následně by měl vyřešit validaci každého prvku pro případné nežádoucí hodnoty. Ba co víc, validace by měla být na straně serveru (PHP), tak i na straně klienta (JavaScript). Nette tuto zdlouhavou rutinu nesmírně usnadňuje. Stačí napsat pouze krátký kód s definicí formulářových prvků a následně se automaticky provede validace na straně serveru i na straně klienta, vyřeší se zabezpečení a prvek se sám vykreslí. Např. kód `$form->addText('jmeno', 'Jméno')->setRequired('Vyplňte prosím Vaše jméno');` zařídí přidání textového pole na stránku a zároveň definuje, že prvek musí být vyplněn, jinak nebude formulář odeslán.

Podpora pro validaci formulářových prvků je velice rozsáhlá. Pro každý prvek lze nastavit, zda má být číselného typu, měl požadovanou délku, jednalo se o URL adresu nebo email. Pokud ani základní pravidla nestačí, je možné použít pro validaci vlastní regulární výraz.

Vykreslení na stránce lze docílit pouhým zápisem makra `{control mujFormular}`. Ve standardním nastavení se formulář vykreslí jako tabulka. Toto rozložení lze samozřejmě změnit, popř. vykreslit formulář zcela individuálně pomocí příslušných maker. Velikou výhodou je možnost stejně snadno – pomocí jediného makra – vložit formulář na několik míst zároveň.

Routování

V žádné moderní webové aplikaci nesmí chybět optimalizace pro vyhledávače, tzv. SEO. Součástí této techniky je i podpora pro „hezké“ URL adresy, mnohdy nazýváno též jako user-friendly URL a nebo cool URL. Ve většině případů programátor vytváří ručně všechny adresy, které následně přepisuje do souboru `.htaccess` a ve výsledné fázi se adresy překládají modulem webserveru mod_rewrite. Díky Nette má vývojář celou situaci velice usnadněnou, nemusí vůbec upravovat žádná nastavení, vše se děje na úrovni frameworku. K tomuto účelu slouží právě routování.

Jedná se o obousměrné překládání mezi URL a akcí presenteru (presenter je v rámci MVC modelu controller). Při vývoji je tedy kompletně upuštěno od klasického odkazování na konkrétní adresu ve stylu `href="uzivatele/prihlaseni.php"`. Vše se řeší skrze

makro `{link}`. Odkaz vedoucí na akci přihlášení v modulu (presenteru) uživatelé bude v šabloně Latte zapsán jako `{link Uzivatele:prihlaseni}`. Obrovskou výhodou je pak skutečnost, že adresy je možné měnit během vývoje, resp. je možné adresy definovat úplně v poslední fázi projektu.

Problematika routování není úplně triviální, nicméně díky dokumentaci a návodům je snadno pochopitelná. Tato metoda tvorby adres umožňuje i mnohé pokročilé funkce. Je možné definovat slovník pro překlad. Máme-li aplikaci napsanou v českém jazyce, adresy budou pravděpodobně také v češtině. Problém nastane v migraci webu do jiného jazyka. I na toto bylo při návrhu frameworku myšleno, neboť lze definovat ekvivalenty pro dané akce. Např. můžeme určit, že modul (presenter) uživatelé je dostupný i jako users. Obdobně akce přihlášení = login apod. Routování nabízí rozsáhlé možnosti nastavení, podporovány jsou různé filtry, masky, jednosměrné adresy, volitelné sekvence atd. Pokud by ani toto náročné aplikaci nevyhovovalo, je možné si napsat zcela vlastní router (modul pro překlad adres). V takovém případě je možné vytvořit prakticky cokoliv, např. adresy určovat podle záznamů v databázi.

Přihlašování a oprávnění uživatelů

Práce s uživateli, tj. přihlašování, ověřování apod., se dnes nachází téměř v každé aplikaci. Každý programátor si správu uživatelů řeší po svém, je však třeba podotknout, že ne vždy je takové řešení zcela optimální a hlavně bezpečné. Proto Nette zavádí vlastní autentifikační modul. Programátorovi tak zcela odpadá nutnost řešit ukládání session a cookies, funkci trvalého přihlášení a mnohé další.

Pro používání modulu je třeba vytvořit pouze autentifikační metodu, což je většinou dotaz do databázového systému, zda uživatel s daným jménem a heslem existuje a zadané údaje jsou platné. O vše ostatní se postará framework. Po tomto kroku je možné využívat funkcionalit pro práci s uživatelem. Pro příklad uveďme velice snadné přihlášení uživatele - `$this->getUser()->login($jmeno, $heslo)`. Chceme-li ověřit, jestli je uživatel přihlášen, snad to provedeme zavoláním metody `$this->getUser()->isLoggedIn()`.

Jakmile je proces přihlášení úspěšně dokončen, získá uživatel identitu, která může obsahovat dodatečné informace jako je třeba uživatelská role. Podle ní lze snadno určovat, zda má dotyčný uživatel přístup do požadované sekce apod. Lze definovat i tzv. autorizátor, který definuje přístup pro daného uživatele (uživatelskou roli) k danému zdroji. Příkladem budiž zjištění, jestli uživatelská role manažer má oprávnění k přidání nového uživatele - `$this->getUser()->isAllowed('manazer', 'uzivatele', 'pridat')`. Mezi dalšími zajímavými funkcemi modulu můžeme jmenovat použití více aplikací v jedné session nebo třeba informaci o důvodu odhlášení uživatele – lze zjistit, zda se uživatel odhlásil sám nebo došlo k automatickému odhlášení či zavření prohlížeče.

Odesílání emailů

Téměř každá aplikace na webu potřebuje odesílat emaily, ať už jsou to různé novinky nebo třeba zapomenutá hesla. V klasickém PHP je pro tyto účely k dispozici funkce `mail()`, ta ale rozhodně neoplývá dokonalostí. Nette obsahuje třídu, které poskytuje poměrně pohodlné odesílání emailů. Níže uvedený příklad mluví za vše, jednotlivé metody není třeba nijak popisovat, podle názvu je ihned jasné, co který řádek dělá.

```
$email = new Nette\Mail\Message;
$email->setFrom('Univerzita Pardubice <info@upce.cz>')
      ->addTo('prijemce1@email.cz')
      ->addTo('prijemce2@email.cz')
      ->setSubject('Předmět emailu')
      ->setHTMLBody("Text zprávy, která je <b>formátována v HTML</b>.")
      ->send();
```

Dále je možné využívat vložení přílohy do emailu, a to snadno pomocí jediné metody `addAttachment()`. Pokud se někomu nelíbí výchozí používání funkce `mail()`, je možné definovat vlastní mailer. Můžeme tak snadno např. nadefinovat a použít vzdálený SMTP server. Skutečná síla se projeví při zkombinování se šablonovacím systémem Latte. Do emailu je vložena celá Latte šablona, tudíž lze velice snadno upravit vzhled celého emailu. Nic se nemusí řešit v kódu, vše je přehledně editovatelné v šabloně.

Ostatní funkcionality a výhody

Framework Nette obsahuje obrovské množství funkcí, doplňků a rozšíření nad standardní jazyk PHP. V seznamu níže je uvedeno několik dalších zajímavých funkcionalit, které stojí za zmínku.

- Cache – při výsledném generování stránky nejsou uživateli vždy sestavovány kompletně nové stránky, nýbrž je využíváno funkce cache. Jakmile se jednou požadovaná stránka sestaví, již není třeba provádět dotazy do databáze apod., odešle se pouze soubor z cache. Tato metoda významným způsobem šetří prostředky serveru a zvyšuje výkon celé aplikace. Třída obsluhující cache je velice „inteligentní“, neboť při změně zdrojového souboru je změna zjištěna a cachovací soubor je obnoven. Výše popsaná je automatická cache, o kterou se programátor nemusí starat a funguje v Nette standardně bez jakéhokoliv zásahu. Dále je k dispozici cache s manuálním ovládáním, kde lze definovat pokročilé vlastnosti typu nastavení expirace, podmínky pro znovuvybudování stránky, výběr úložiště apod.

- Auto-loading tříd – automatické vkládání (includování) potřebných souborů. Tato funkce pracuje inteligentně, tudíž načítá opravdu jen potřebné soubory, nikoliv všechny. Ušetří práci nejen programátorovi s přemýšlením, kde který soubor vložit, ale také uleví zátěži serveru.

- Práce s obrázky – PHP obsahuje knihovnu GD, která umožňuje pracovat s obrázky, nicméně její používání je dosti těžkopádné (např. různé funkce pro každý obrazový formát). Proto Nette přichází s vlastní třídou. Ta zjednodušuje nejčastější úkony, jako je změna velikosti, doostření nebo odeslání do prohlížeče.

- Stránkování – možnost automatického vytvoření „stránkovače“, tedy komponenty, která umožňuje procházet stovky a tisíce záznamů po stránkách. Stačí pouze nastavit data, která se mají stránkovat a počet záznamů na stránku, ostatní je vytvořeno automaticky.

Závěrem k Nette

V rámci projektu Nette existuje na internetu veliká komunita lidí, která diskutuje nad problémy frameworku. Jelikož se jedná o český výtvor, většina textů a diskuzí je také v češtině. Autor Nette se sám zapojuje do diskuzí, a pokud je nalezena chyba nebo navrhnuta nová funkce, je toto implementováno do nové verze.

Pozn.: následující odstavec vychází ze zkušeností autora této práce. Výše popisovaná komunita je velice aktivní, což je pro každý projekt rozhodně dobré, nicméně má to i svoje negativní strany. Díky tak „bouřlivému“ vývoji Nette nebyla dlouhou dobu k dispozici žádná dokumentace, vše se muselo tvořit víceméně metodou pokus-omyl, což není úplně ideální. Ještě větším problémem byla nekompatibilita verzí. Bylo běžnou praxí, že po upgradu na vyšší verzi (ač se jednalo o „setiny“ verze) přestal fungovat starý kód a bylo nutné aplikaci opravit. Dnes jsou již oba nedostatky odstraněny, komunita je stále aktivní, na čemž se podepsala i kvalitní dokumentace – dostupná v českém i anglickém jazyce. Po vydání verze Nette 2.0 je i vývoj ustálen a přechod na novější verzi (2.x) je bezproblémový.

6.1.5 Dibi

Dibi je knihovna pro skriptovací jazyk PHP, která usnadňuje a zjednodušuje práci s databází. Především pak zjednodušení zadávání SQL příkazů a ulehčení běžně používaných rutin – např. získání výsledku dotazu jako dvourozměrné pole, import/export SQL souboru atd. Mezi další přednosti patří nezávislost databázového systému, Dibi podporuje MySQL, PostgreSQL, SQLite, MS SQL a další. Programátor může díky této nezávislosti vytvořit svoji aplikaci pod jedním typem databáze a v produkčním prostředí je možné bez větších problémů využít zcela jiný databázový systém.

Dibi není jen databázová vrstva, je to velice dobrý pomocník pro práci s databází. Pro kompletní popis funkcí by bylo třeba samotné publikace. Nicméně níže je uvedeno několik – dle autora – zajímavých a praktických věcí, které Dibi poskytuje.

Zástupné znaky pro datové typy

Nespornou výhodou Dibi je možnost používání zástupných znaků pro různé datové typy. Blíže tuto funkcionalitu popíšeme na příkladu.

```
dibi::query('SELECT * FROM [table] WHERE [id] = %i', $id);
```

Metoda `dibi::query` vyvolá standardní dotaz do databáze, přičemž si všimněme dvou neobvyklých věcí. Název tabulky (`table`) je uzavřen do hranatých závorek. Stejně tak je to u sloupce `id`. Dibi automaticky upraví syntaxi SQL dotazu pro aktuálně používanou databázi. Pro MySQL vznikne z `[table]` text `` `table` `` a např. při použití ODB zůstane zachováno `[table]`. Dotaz se zkrátka „přizpůsobí" danému databázovému systému.

Velice obdobně funguje i další zástupný text, což je v našem případě `%i`. Tento zástupný text je nahrazen proměnnou `$id`, ve které je očekáván datový typ `integer`. Dibi opět inteligentně daný datový typ „upraví" tak, aby vyhovoval použité databázi. Např. u MySQL textové proměnné (`string`) přidá na začátek a konec apostrofy a případný apostrof uvnitř textu escapuje – přidá před něj zpětné lomítko. Zástupných textů existuje celá řada. Tabulka níže popisuje vybrané zástupné řetězce.

Tabulka 7 – Zástupné řetězce v Dibi

Zástupný řetězec	Očekávaný vstup
%s	string
%b	boolean
%i	integer
%f	float
%d	datum (očekává string nebo integer)
%t	datum & čas (také string či integer)
%sql	SQL – řetězec ponechá beze změny

Databázová vrstva Dibi obsahuje mnohé bezpečnostní praktiky, mezi které patří právě i výše zmíněný zástupný text. Chce-li útočník použít útok skrze SQL injection, Dibi mu to nedovolí. Pokud použijeme zástupný řetězec, je samotný dotaz vždy řádně escapován dle daného databázového typu, a proto nehrozí ona praktika SQL injection.

```
$text = "I'm fine";
dibi::query('UPDATE `table` SET `text`=%s', $text);
// MySQL: UPDATE `table` SET `text`='I\'m fine'
// ODBC:  UPDATE [table] SET [text]='I''m fine'
```

Vkládání celého pole jako parametr dotazu

Díky této vlastnosti databázové vrstvy Dibi si programátor velice usnadní a zrychlí práci. Dibi umoňuje vložit jako parametr metody celé PHP pole, které je následně automaticky převedeno na dvojici klíč – hodnota.

```
$arr = array(
    'pole' => 'hodnota',
    'bit'  => TRUE,
);
dibi::query('INSERT INTO [table]', $arr);
```

Pro MySQL bude finální dotaz vypadat následovně.

```
INSERT INTO `table` (`pole`, `bit`)
VALUES ('hodnota', 1)
```

Skládání dotazu

Pro mnoho programátorů je mnohdy problém řešit tzv. skládání dotazu. Pokud máme PHP kód přes několik desítek řádků a v různých částech kódů vyhodnocujeme podmínky, na základě kterých je dotaz upraven, je takový výsledný dotaz poměrně složité vytvořit. S Dibi je to jiné. Stačí použít obyčejnou proměnnou typu pole a přidávat do této proměnné segmenty složeného dotazu.

```
$dotaz[] = "SELECT * FROM [table]";

if (isset($objednavka)) {
    $dotaz[] = "WHERE [id_objednavky]=%i";
    $dotaz[] = $objednavka;
}

if ($seskupit == 1) {
     $dotaz[] = "GROUP BY [objednavky_id]";
}

if (isset($limit)) {
    $dotaz[] = "%lmt";
    $dotaz[] = $limit;
}
```

Z výše uvedeného kódu nám vznikne např. následující dotaz.

```
SELECT *
FROM `table`
GROUP BY `objednavky_id`
LIMIT 20
```

Získávání výsledků

Dibi obsahuje několik metod pro získání výsledků dotazu. Základní metodou je `dibi::query()`, nad kterou je následně možné použít klasickou iteraci. V každém průchodu dané iterace je vrácen jeden řádek tabulky.

Dále je možné získat jeden (první) řádek tabulky. To nám umožňuje metoda `dibi::fetchSingle()`. Užitečná je také metoda `dibi::fetchPairs()` pro získávání dat v podobě asociativního pole klíč => hodnota.

Velice často používanou metodou je `dibi::fetchAll()`, která vrací celou tabulku v indexovaném poli.

```
array(5) [
    0 => DibiRow(16) {
        uzivatele_ID => 1
        jmeno => "Marek"
    }
    1 => DibiRow(16) {
        uzivatele_ID => 2
        jmeno => "Milan"
    }
    2 => DibiRow(16) {
        uzivatele_ID => 3
        jmeno = "Adam"
    }
]
```

Všimněme si vráceného záznamu – objekt `DibiRow`. Tento objekt má vlastnost, že je možné k jeho datům přistupovat jako k poli, tedy např. `$uzivatel[0]['jmeno']`. Zároveň je možné použít objektový přístup, tedy `$uzivatel[0]->jmeno`.

Transakce

Transakcí můžeme zjednodušeně nazvat několik dotazů, které se jako celek chovají atomicky. Což znamená, že pokud se jeden z dotazů nepovede vykonat, ostatní jsou také neplatné. Dibi má přímou podporu pro transakce a jejich použití je velice snadné.

```
dibi::begin(); // nastartování transakce

try {
    $zjistene_id = dibi::fetchSingle("SELECT [id] FROM [uzivatele] WHERE
                                      [jmeno]=%s", "Adam");

    dibi::query("INSERT INTO [log]", array(
        "uzivatel" => $zjistene_id,
        "cas%sql" => "NOW()"
    ));

    dibi::query("UPDATE [nastaveni] SET [posledni_zmena]=%sql", "NOW()");

    dibi::commit(); // vše je v pořádku - commit

} catch (DibiDriverException $e) {
    dibi::rollback(); // něco se pokazilo - rollback
}
```

Další „vychytávky“

Mezi další „vychytávky“, které se sluší uvést, jsou podmíněné dotazy. Díky nim může programátor přímo ovlivnit dotaz skrze obvyklé podmínky if a else.

Mnohdy se pro daný projekt používají tzv. prefixy. Tedy název tabulky je uvozen prefixem, např. `mojeApp_`. Výsledný název tabulky je pak `mojeApp_table`. V klasické

aplikaci je to poměrně složité řešit – je nutné upravit všechny dotazy. V Dibi stačí použít metodu `dibi::addSubst()` a problém je vyřešen.

Velice užitečnou metodou je `dibi::test()`. Ta zobrazuje výsledný dotaz, který bude odeslán do databáze. Reálně není databáze vůbec použita, jen je zobrazen SQL dotaz. Díky této metodě je možné pohodlně testovat funkcionality Dibi.

6.1.6 jQuery

Jelikož dnes existuje mnoho různých webových prohlížečů s odlišným vývojem, jsou určité funkcionality a vlastnosti JavaScriptu implementovány v každém prohlížeči různě. Příkladem budiž zjištění šířky a výšky okna prohlížeče. V prohlížeči Internet Explorer je nutné použít odlišný kód než pro Mozilla Firefox. Nejen z tohoto důvodu vzniklo a vzniká mnoho rozšíření (knihoven) pro JavaScript – jQuery je jednou z nich. Jedná se o JavaScriptovou multiplatformní knihovnu s podporou napříč mnoha prohlížeči. „Instalace“, resp. implementace, je velice snadná, spočívá pouze v nalinkování jednoho souboru, následně je jQuery ihned připraveno k použití.

Knihovna poskytuje veliké pole použití – výběr dynamických objektů na stránce (DOM), včetně možnosti upravení. Přidává podporu pro efekty a animace na stránce, umožňuje manipulovat s kaskádovými styly (CSS) a sjednocuje metody pro všechny prohlížeče. V neposlední řádě usnadňuje práci programátorovi, neboť v porovnání s klasickým JavaScriptem není potřeba napsat tolik kódu pro stejnou funkcionalitu (srovnání níže). Základem pro práci je funkce `$`, což je v podstatě alias pro jmenný prostor jQuery. Níže je uvedeno několik klíčových vlastností.

- AJAX - díky technologii AJAX je možné načítat data ze serveru bez nutnosti obnovení celé stránky. Obnoví se pouze ta část, která je vyžádána. Součástí jQuery je bohatá podpora pro tuto technologii. Co je však zásadnější, implementace AJAXu se v různých prohlížečích liší, nicméně v jQuery je tento problém vyřešen – chování funkcí je zcela nezávislé na prohlížeči. Nejdůležitějšími funkcemi jsou `$.get()` načítající data skrze HTTP GET metodu, obdobou je `$.post()`, která načítá požadavkem HTTP POST. Zajímavá je též funkce `$.getJSON` načítající data ze vzdáleného serveru ve formátu JSON.

- Selektory – snadný přístup k prvkům webové stránky. Selektor `$('div.moje-trida')` vybere všechny prvky na stránce, které mají nastavenou třídu `moje-tridy` (`class="moje-trida"`). Podporováno je mnoho metod přístupu – pomocí id, jména, typu prvku (`div`, `table`, `span`…). Nicméně selektorů existuje celá řada a nejsme omezeni pouze na základní výše jmenované. Např. selektor `$('td.bunka:last-child')` vybere všechny buňky tabulky (`td`), které mají třídu `bunka` (`.bunka`) a jsou posledním prvkem svého rodiče (`:last-child`).

- CSS – práce s kaskádovými styly. K dispozici je funkce `$.css()`, která získá/nastaví požadované vlastnosti prvku, např. `$('table').css('display',`

`'none');` skryje všechny tabulky. Užitečné jsou funkce `addClass()` a `removeClass()`, které přidají/odstraní třídu objektu.

- Události – vykonání akce při definované události (kliknutí myši apod.). Dostupné jsou funkce pro události známé z klasického JavaScriptu, tedy `mouseout()`, `mousemove()` apod. Nicméně silnou stránkou jQuery je tzv. bindování (česky svázání) s určitým objektem. V kombinaci s využitím selektorů se jedná o nadmíru efektivní a elegantní řešení. `$('p').on('click', function() {…});` – po kliknutí na libovolný odstavec (HTML tag `<p>`) v dokumentu se provede definovaný kód.

- Efekty – podpora pro grafické efekty. V podstatě se jedná „pouze“ o oku lahodící vizuální efekty. Tudíž místo prostého skrytí objektu můžeme snadno tento objekt animovat. V konečném důsledku bude objekt také skryt, nicméně před samotným skrytím však objekt např. zeslábne do ztracena. Podporováno je mnoho animací, ať se jedná o změnu průhlednosti nebo různé posuvy apod.

- Pomůcky pro usnadnění – existuje celá řada „nadstandardních“ funkcí, bez kterých bychom se obešli, každopádně jejich přítomnost nám velice usnadní práci. Příkladem je funkce `each()`, což je v podstatě obdoba `foreach` známého z jiných programovacích jazyků. Zápis `$('li').each(function () {…});` umožní projít každou nečíslovanou odrážku (HTML tag `<li>`) a vykonat nad ni určenou akci. Mezi další pomocné funkce můžeme zmínit `inArray()` nebo `isNumeric()`, což jsou funkce notoricky známé z programovacího jazyka PHP a které v klasickém JavaScriptu chybí.

Knihovna jQuery je velice rozsáhlá a poskytuje rozmanitou škálu funkcionalit. Rozhodně ale nepokrývá úplně vše. Proto je možné pro rozšíření použít tisíce doplňků dostupných na portálu `http://plugins.jquery.com`.

Ukázkový příklad

Níže je uveden komentovaný zdrojový kód napsaný v klasickém JavaScriptu a jeho ekvivalent pro jQuery. Skript provádí načtení souboru technologií AJAX a následné vypsání všech buněk tabulky, resp. HTML kódu uvnitř buněk.

Nejprve klasický JavaScript.

```
// prvek DIV, kam bude AJAXem načten obsah souboru
<div id="myDiv"></div>

<script type="text/javascript">
    var xmlhttp;
    if (window.XMLHttpRequest) {
        xmlhttp=new XMLHttpRequest(); // AJAX pro Firefox, Chrome a Operu
    } else {
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP"); // AJAX pro IE
    }

    // až bude dokončeno volání AJAXem (soubor se načte)
    xmlhttp.onreadystatechange=function() {
        if (xmlhttp.readyState==4 && xmlhttp.status==200) { // vše je OK
            document.getElementById('myDiv').innerHTML =
                xmlhttp.responseText;
            // nalezení všech <td>
            var elements = document.body.getElementsByTagName('td');
            for (var i=0; i<elements.length; i++) { // pro každý element
                alert(elements[i].innerHTML); // vypsání hodnoty
            }
        }
    }

    xmlhttp.open("GET","test.html", true); // nastavení AJAXového volání
    xmlhttp.send(); // odeslání AJAXového požadavku
</script>
```

Všimněme si problematického zápisu u definování AJAXové proměnné (`xmlhttp`). V prohlížeči Internet Explorer se inicializace provádí jinak než v ostatních prohlížečích. Zdlouhavý je také zápis samotného ověření výsledku – jestli je vše v pořádku. Následuje ekvivalentní kód v jQuery.

```
// prvek DIV, kam bude AJAXem načten obsah souboru
<div id="myDiv"></div>

// načtení knihovny jQuery
<script src="http://code.jquery.com/jquery-1.9.1.js"></script>

<script type="text/javascript">
    // AJAXový požadavek na soubor test.html
    $.get("test.html", function (data) {
        $('#myDiv').html(data); // naplnění DIVu načtenými daty

        $('td').each(function () { // pro každý element <td>
            alert($(this).text()); // vypsání hodnoty
        });

    });
</script>
```

Úsudek o obou zobrazených kódech si jistě udělá každý. Rozhodně ale všichni budeme souhlasit s tvrzením, že knihovna jQuery je velice mocný a užitečný pomocník.

6.1.7 jQuery UI

Již podle názvu lze odvodit, že se jedná o rozšíření jQuery. Konkrétně jde o rozšíření o další efekty a usnadnění implementace pokročilých elementů. Pro využívání tohoto doplňku je zapotřebí mít jQuery, samotná instalace spočívá v nalinkování dvou souborů (knihoven) – jQuery a jQuery UI.

JQuery UI se dělí do následujících skupin. U každé z nich budou popsané zajímavé funkcionality.

- Interactions – přidává pokročilé interakce mezi uživatelem a webovou stránkou. Obsahuje celkem pět metod – Draggable, Droppable, Resizable, Selectable, Sortable. Z názvů lze odvodit chování jednotlivých metod, např. díky metodě Sortable lze snadno přemisťovat pořadí prvků v rámci kolekce.

- Widgety – často používané mini-aplikace. Opět obsahuje několik různých widgetů, jako příklad lze uvést Progressbar, což je v podstatě ukazatel nějakého stavu (v procentech). Dále můžeme jmenovat Datepicker, což je kalendář, na kterém lze snadno uživatelem vybrat datum. Velmi populární je také Dialog, což může být jakási obdoba funkce `alert()` v klasickém JavaScriptu. Navíc však přidává možnost definice vlastních tlačítek a zcela libovolného HTML kódu.

- Efekty – podobně jako u klasického jQuery, i zde jde o animování elementů. V podstatě se jedná o další efekty nad rámec standardní nabídky jQuery. Najdeme zde efekty jako je exploze nebo třesení.

- Utility – podpůrné nástroje pro tvorbu vlastních widgetů.

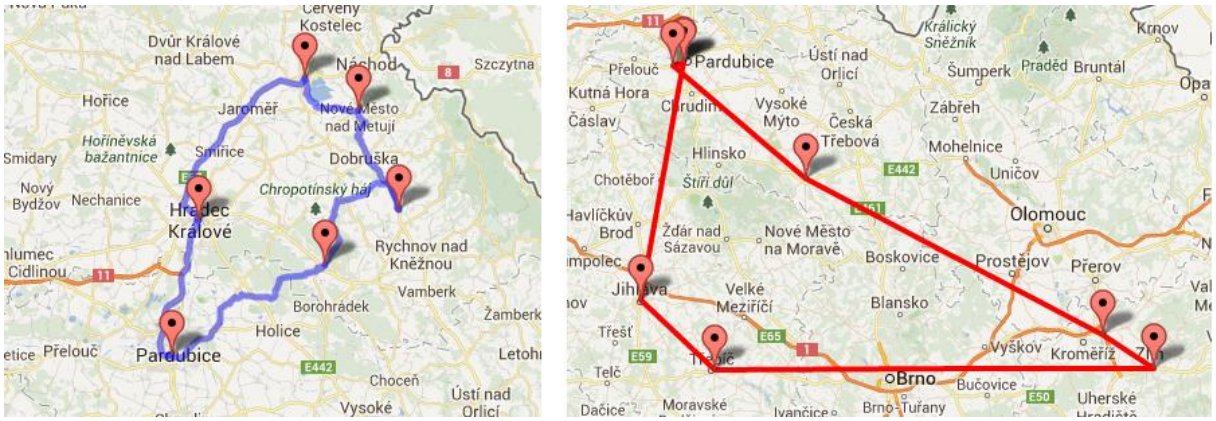
6.1.8 Google Maps API

V rámci aplikace je využíváno mapových podkladů Google Maps. Důvodem výběru právě této služby je několik. Hlavním důvodem je dostupnost řešení problému obchodního cestujícího přímo v mapě (google-maps-tsp-solver). Dalším důvodem jsou kvalitní a přesné mapy, velmi propracované API a bohatě zdokumentovaný manuál. V neposlední řadě hrála roli i zkušenost s produktem, neboť autor této práce s Google Maps API v minulosti již pracoval. Produkt, včetně ukázkového zdrojového kódu, byl popsaný v jiné kapitole, proto se detailním popisem služby nebudeme zabývat. Nicméně je nutné uvést, kde a v jakém rozsahu je tato služba v aplikaci využívána.

Google Geocoding API – na základě zadané adresy vrací GPS souřadnice místa. Využívá se při vkládání nové objednávky. V detailu objednávky je možné zobrazit adresu příjemce, pro tuto funkci je zobrazována mapa pomocí Google Maps API, přičemž příjemce je na mapě označen pomocí markeru.

Nejvýznamnější použití Google Maps API spočívá ve výpočtu optimální trasy pro jednotlivá vozidla (viz popis google-maps-tsp-solver) a následné vizualizace optimální trasy. Výsledná „optimální“ trasa je za pomoci Google Maps API vykreslena na webové stránce. Jednotlivé průjezdní body jsou na mapě označeny pomocí markerů, přičemž jejich

cesta (spojení) může být vykreslena dvěma způsoby. Prvním způsobem spojení je vypočítaná trasa mezi dvěma body. Takovéto spojení je možné provést za pomocí dodatečné služby Google Maps API, a to Directions Service. Po zadání dvou bodů – počáteční a cílový – je vypočítána nejkratší trasa mezi těmito dvěma body. Jedná se v podstatě o integrovaný plánovač tras. Bohužel má tato metoda dosti tvrdé omezení – je možné vypočítat trasu pouze pro maximálně 8 bodů. V opačném případě je nutné zvolit druhý způsob spojení – klasická spojnice mezi dvěma body (markery) vzdušnou čarou. Ačkoliv není touto metodou přesně znázorněna trasa, pro obecný přehled je metoda dostačující. Na obrázku níže je porovnání obou metod – vlevo metoda pomocí Directions Service, vpravo pak přímé spojení vzdušnou čarou.



Obrázek 14 – Porovnání metod spojení bodů na mapě (vlevo Directions Service, vpravo spojení vzdušnou čarou)

6.1.9 Google-maps-tsp-solver

Jedná se o třídu řešící problém obchodního cestujícího v prostředí Google Maps. Jelikož je třída vytvořená v jazyce JavaScript, výpočet probíhá na straně klienta, nikoliv přímo na serveru. V praxi celý postup výpočtu trasy vypadá následovně:

- uživatel webu vstoupí na stránku s výpočtem trasy,
- na stránce je načteno prostředí Google Maps,
- načte se knihovna google-maps-tsp-solver,
- vypočítá se optimální trasa,
- „nějak“ se zobrazí výstup – přímo v mapě, pomocí seznamu bodů, případně se data uloží do databáze apod.

Do algoritmu je vloženo n bodů (míst), které chceme navštívit. Následně si můžeme vybrat mezi dvěma módy výpočtu, a to (i) tzv. „Fastest Roundtrip“, česky přeloženo jako nejrychlejší okružní jízda. V tomto módu se vypočítá cesta z počátečního bodu A skrze všechny definované body a trasa skončí opět v bodě A. Tedy – jak již název napovídá – okružní jízda s návratem na začátek. Druhá možnost (ii) je tzv. „Fastest A-Z Trip“, který

funguje podobně jako výše popsané, nicméně trasa končí v posledním navštíveném bodě, nevrací se zpět na začátek. Po ukončení výpočtu můžeme výsledek algoritmu získat několika rozdílnými způsoby. Pomocí metody `getGDirections()` dostáváme k dispozici objekt `GDirections`, který je přímo podporován samotnými Google Maps. Stačí tento objekt vložit do mapy a okamžitě se vykreslí trasa do definované mapy. Pokud chceme data zobrazit po svém a nevizualizovat je na mapě, lze použít metodu `getOrder()`, která vrací klasické pole s pořadím jednotlivých bodů v optimálním pořadí, např. `[5, 3, 4, 0, 2, 1]`. Na první pohled je zřejmé, že jako první bod je vypočítáno stanoviště s indexem 5, následuje pozice 3, 4, 0 atd. Velmi užitečnou metodou je pak `getDurations()`, díky níž lze získat odhadovaný čas průjezdu vypočítané trasy. Níže je uveden komentovaný elementární kód pro výpočet okružní jízdy.

```
// element DIV, do kterého bude vykreslena mapa
<div id="map-canvas" style="width: 800px; height: 400px"></div>

// načtení API
<script type="text/javascript"
src="http://maps.googleapis.com/maps/api/js?key=AIzaSyAi1dDUK6UwPv0xKF_8E
QACPXwfgNtFjVg&sensor=false"></script>
<script type="text/javascript"
src="http://optimap.net/BpTspSolver.js"></script>

<script type="text/javascript">
    // klasické načtení Google Maps
    var mapOptions = {mapTypeId: google.maps.MapTypeId.ROADMAP}
    var map = new google.maps.Map(document.getElementById("map-canvas"),
mapOptions);

    // vytvoření objektu TspSolver
    tsp = new BpTspSolver(map);

    // nastavení (vyhnout se dálnicím a silniční mód = jízda autem)
    tsp.setAvoidHighways(true);
    tsp.setTravelMode(google.maps.DirectionsTravelMode.DRIVING);

    // vložení průjezdných míst – nejprve pomocí GPS souřadnic (Univerzita
    // Pardubice) a následně pomocí adresy
    tsp.addWaypoint(new google.maps.LatLng(50.033330, 15.767395));
    tsp.addAddress('ČVUT Praha');
    tsp.addAddress('VUT Brno');
    tsp.addAddress('Univerzita Palackého Olomouc');

    // spuštění výpočtu okružní jízdy
    tsp.solveRoundTrip();

    // vypsání pořadí jednotlivých bodů do konzole prohlížeče
    console.log(tsp.getOrder());
</script>
```

Samotný výpočet optimální trasy probíhá pomocí více variant. Lze zvolit buď výpočet hrubou silou (brute-force), tedy porovnávání každý s každým. Tato metoda je však velice náročná a již při několika málo bodech trvá neúnosně dlouho. Druhou metodou pro výpočet optimální trasy – avšak aproximované – je výpočet založený na optimalizaci mravenčí kolonie (Ant Colony Optimization – ACO). Pomocí konstanty `maxTspDynamic`

přímo v těle třídy lze nastavit maximální počet bodů, které budou počítány metodou hrubé síly. Pokud bude tato hodnota překročena, použije se druhá popisovaná (aproximační) metoda.

Google-maps-tsp-solver je šířen pod svobodnou MIT licencí, která dovoluje použít skript komukoliv a bez poplatku. Autor vytvořil ukázkový web, kde demonstruje sílu jeho projektu. Adresa demo stránky je `http://optimap.net`.

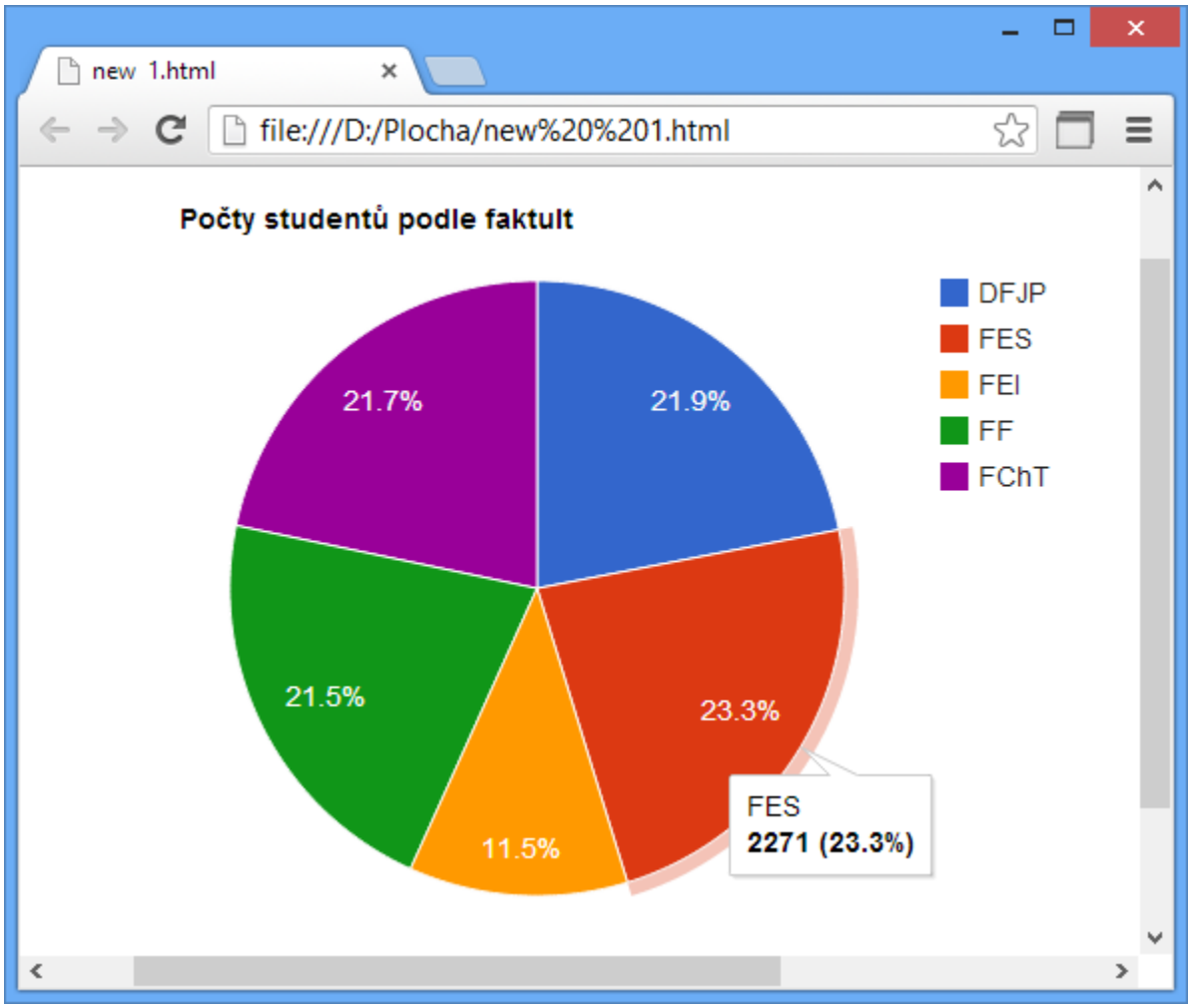
6.1.10 Google Charts

Do češtiny přeloženo jako Google grafy, je nástroj pro snadnou vizualizaci dat v aplikacích. Google Charts nabízí velké množství typů grafů – od jednoduchých grafů až po komplexní hierarchické mapy. Nejsnazší cestou, jak vykreslit data, je použití JavaScriptových knihoven. Do webové stránky stačí načíst poskytované knihovny, pomocí krátkého kódu popsat typ a vlastnosti grafu a dodat data. Následně stačí již pouze specifikovat cílový element DIV, do kterého se graf zobrazí. Samotné vykreslování grafu probíhá pomocí technologií HTML5/SVG, které podporuje převážná většina webových prohlížečů – včetně těch mobilních. Pro starší prohlížeče (Internet Explorer 6) je použita technologie VML. Obrovskou výhodou celého produktu je využití zdarma pro kohokoliv – včetně organizací. Odpadá tak nutnost zdlouhavé registrace a generování klíčů pro jednotlivé domény jako je tomu např. u Google Maps.

Následuje komentovaný elementární kód pro demonstraci použití Google Charts.

```
// element DIV, do kterého bude graf vykreslen
<div id="chart_div" style="width: 900px; height: 500px;"></div>

<script type="text/javascript"
src="https://www.google.com/jsapi"></script>
<script type="text/javascript">
    // inicializace
    google.load("visualization", "1", {packages:["corechart"]});
    google.setOnLoadCallback(drawChart);

    function drawChart() {
        // definice zdrojových dat
        var data = google.visualization.arrayToDataTable([
            ['Fakulta', 'Počet studentů'],
            ['DFJP', 2139],
            ['FES', 2271],
            ['FEI', 1121],
            ['FF', 2098],
            ['FChT', 2117]
        ]);

        // titulek (nadpis) grafu
        var options = {title: 'Počty studentů podle faktult'};

        // vykreslení grafu
        var chart = new
google.visualization.PieChart(document.getElementById('chart_div'));
        chart.draw(data, options);
    }
</script>
```

Výše uvedený zdrojový kód vypíše koláčový graf (Pie Chart), kde budou zobrazeny fakulty Univerzity Pardubice a počty studentů jednotlivých fakult. Po umístění kurzoru myši nad libovolnou část grafu se zobrazí informace s popiskem. Na obrázku níže je výstup uvedeného kódu. Pozn.: počty studentů jsou převzaty z oficiálních stránek Univerzity Pardubice a aktuální ke dni 31. 12. 2012.



Obrázek 15 – Ukázka použití služby Google Charts

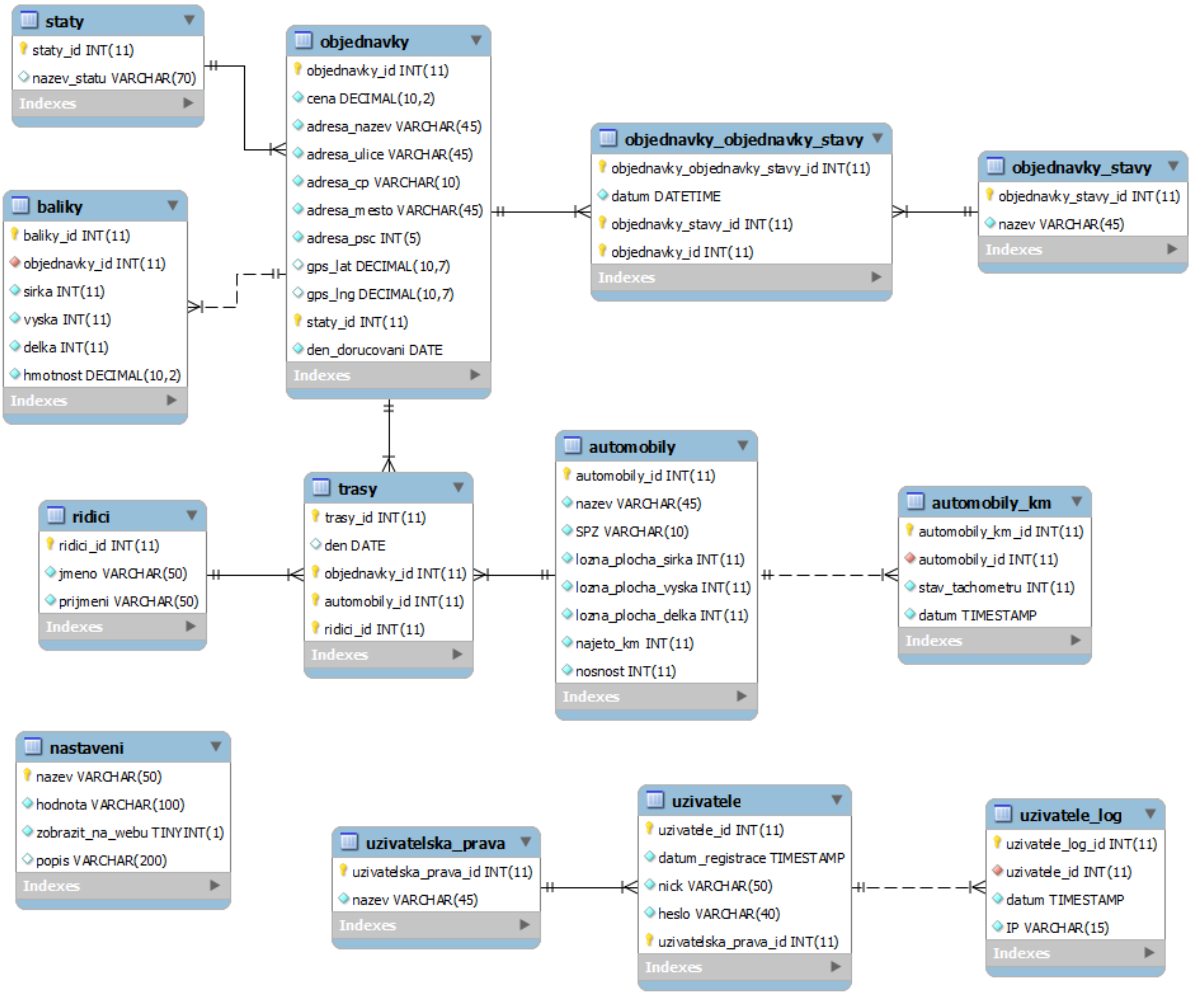
Google Charts podporuje mnoho typů grafů, jmenovitě jsou to tyto:

- Pie Chart – koláčový graf,
- Scatter Chart – rozptylový graf zobrazující data z číselných hodnot na obou osách,
- Gauge – měřič vhodný pro zobrazení rychlosti, teploty apod.,
- Geo Chart – pokročilý geografický „graf“, vykresluje mapu světa a pro jednotlivé oblasti (státy, regiony…) zobrazuje četnosti – např. četnost objednávek do jednotlivých zemí,
- Table – klasická tabulka s možností řazení,

- Treemap – zobrazení datového stromu,
- Combo Chart – kombinace více typů grafu v jednom,
- Line Chart – spojnicový graf,
- Bar Chart – pruhový graf,
- Column Chart – sloupcový graf,
- Area Chart – podobný jako spojnicový graf, navíc přidává výplň oblastí,
- Candlestick Chart – burzovní grafy.

6.2 Databázový návrh a popis tabulek

Databázový model obsahuje celkem 13 tabulek. Databáze je navržena dle platných pravidel pro návrh databáze (3NF), což umožňuje případné pohodlné rozšíření funkčnosti do budoucna.



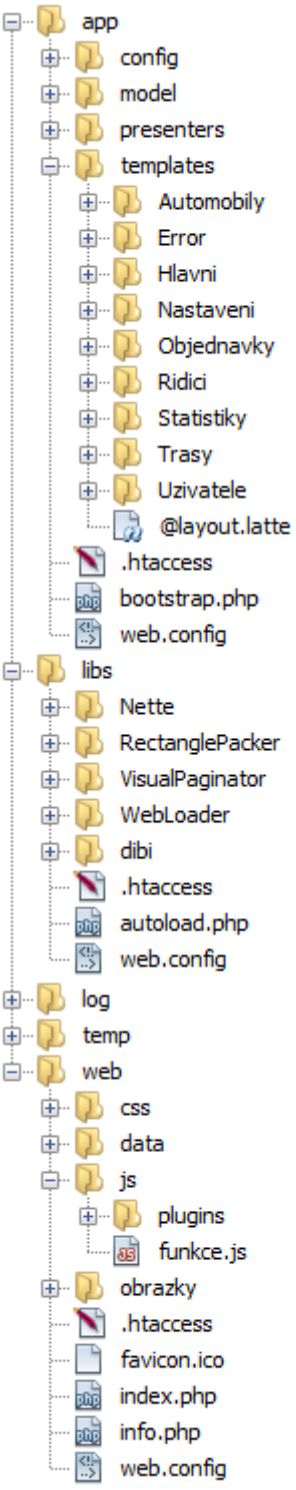
Obrázek 16 – Databázový návrh aplikace

Následuje stručný popis tabulek:

- `staty` – jednoduchá tabulka obsahující seznam států.
- `baliky` – seznam doručovaných balíků. Ukládány jsou informace o rozměrech (šířka, výška, délka) a hmotnosti. Cizím klíčem je sloupec `objednavky_id`, který ukazuje na objednávku, jež balík patří.
- `objednavky` – jedna z největších a nejdůležitějších tabulek. Nese informace o jednotlivých objednávkách, tj. cena, doručovací adresa, GPS souřadnice místa ve formátu WGS84 a den, kdy se má objednávka doručovat.
- `objednavky_objednavky_stavy` – spojovací tabulka pro vztah typu M:N. Obsahuje seznam objednávek a jejich stavů. Současně je uchováváno datum, kdy ke změně objednávky došlo.
- `objednavky_stavy` – názvy stavů objednávek, např. doručená, expedovaná apod.
- `ridici` – seznam řidičů.
- `trasy` – další důležitá tabulka obsahující data o jednotlivých trasách. Sdružuje informace (cizí klíče) z několika tabulek – `objednavky`, `ridici` a `automobily`. Pro každý den je uchováváno, kterou objednávku poveze který řidič, plus jaké auto pro rozvoz použije.
- `automobily` – seznam dostupných automobilů, včetně technických informací o vozidle, tj. nosnost a ložná plocha.
- `automobily_km` – nese informace o stavu tachometru pro každé vozidlo. Důležitá tabulka pro sledování vytíženosti automobilů, případně pro další akce – např. doplňování oleje po x najetých kilometrech.
- `nastaveni` – systémová tabulka, která obsahuje důležitá nastavení celé aplikace.
- `uzivatelska_prava` – definice jednotlivých uživatelských práv aplikace, např. admin, dispečer atd.
- `uzivatele` – seznam uživatelů aplikace. Mimo jiné obsahuje také datum registrace a heslo v hashované podobě (sha1).
- `uzivatele_log` – tabulka sloužící k logovacím účelům. Sleduje aktivitu přihlášení uživatelů aplikace.

6.3 Adresářová struktura

Díky využití frameworku Nette je adresářová struktura poměrně pevně daná. Ačkoliv je možné strukturu od výchozího uspořádání Nette změnit, není to nezbytně nutné. Výchozí struktura zároveň poskytuje vyšší míru zabezpečení z důvodu skrytí zdrojových souborů mimo viditelný adresář z webu (document_root).



Obrázek 17 – Adresářová struktura aplikace

V hlavním adresáři se nachází celkem pět adresářů – `app`, `libs`, `log`, `temp` a `web`. Postupně bude každý z adresářů podrobněji popsán. Je nutné zmínit, že tyto adresáře se nenacházejí v document_root serveru, nejsou tedy zvenčí (skrze http protokol) viditelné. V rámci celé adresářové struktury se hojně vyskytují soubory `.htaccess` a `web.config`. Jedná se o konfigurační soubory pro webserver apache, resp. Microsoft IIS. Díky těmto konfiguračním souborům lze snadno zakázat/povolit přístup do daného adresáře. Existence těchto souborů je tedy z hlediska bezpečnosti kritická.

Prvním adresářem je `app`, je také asi nejdůležitější, neboť obsahuje všechny zdrojové soubory aplikace. Podle obrázku stromové struktury je na první pohled zřetelné rozdělení na architekturu MVC, resp. MVP (model-view-presenter). Adresář `app` obsahuje následující podadresáře:

- `config` – obsahuje jediný soubor, a to `config.neon`. Ten v sobě nese nastavení celé aplikace, např. nastavené časové pásmo, dobu expirace sessions nebo přihlašovací údaje k připojení k databázi.

- `model` – písmeno „M“ ze zkratky MVC, obsahuje tedy php soubory s modely celé aplikace. Jednotlivé třídy nebudou na tomto místě dále popisovány, neboť jejich seznam včetně metod je zobrazen v kapitole popisující UML diagram tříd.

- `presenters` – podobně jako u modelů, i presentery jsou znázorněny v kapitole zabývající se UML diagramy.

- `templates` – šablony aplikace. V podstatě se jedná o rozšířené HTML soubory, přesněji Latte soubory. Adresář obsahuje další podadresáře, které jsou adekvátní k seznamu presenterů. V každém takovém podadresáři (`Automobily`, `Error`, `Hlavni` atd.) nalezneme soubory s příponou

`.latte`, což jsou výsledné šablony pro danou akci. Např. stránka s adresou `/automobily/seznam` bude frameworkem načítána z adresáře `/app/templates/Automobily/seznam.latte`. Za zmínku stojí také soubor `@layout.latte`, což je hlavní šablona, do které je vkládána požadovaná dílčí šablona (např. onen zmiňovaný `seznam.latte`).

- soubor `bootstrap.php` – zaváděcí soubor frameworku. Načítá konfiguraci a knihovny Nette. Dále je v něm možné definovat mnoho dalších funkcí a vlastností – routování (překlad adres), zapnutí/vypnutí logování atd.

Další položkou ve stromové struktuře hlavního adresáře je `libs`. Jak již samotný název napovídá, jedná se o adresář s knihovnami (z anglického library – knihovna). Podadresář `Nette` obsahuje soubory stejnojmenného frameworku, v adresáři `RectanglePacker` nalezneme jediný soubor – `RectanglePacker.php`, což je třída pro řešení Container loading problem (vysvětlení viz jiná kapitola). `VisualPaginator` je pomocná komponenta frameworku, která zjednodušuje stránkování v aplikaci, podobně je tomu i v případě adresáře `WebLoader`. Tato knihovna umožňuje spojování více souborů JavaScriptu (*.js) a stylopisů (*.css) do jednoho, podporována je také minifikace. Díky spojení do jednoho souboru a následné minifikaci je ušetřeno několik požadavků na server – stahuje se pouze jeden css a jeden js soubor namísto desítek menších souborů. V adresáři `dibi` nalezneme stejnojmennou knihovnu pro práci s databází. Konečně poslední položka – soubor `autoload.php` – se stará o správné načtení frameworku Nette.

Adresáře `log` a `temp` jsou pouze pomocné a jsou využívány samotným frameworkem. V adresáři `log` nalezneme soubory logu, tedy zprávy o chybách, varování a upozornění. Adresář `temp` obsahuje cache frameworku Nette a soubory pro uchování sessions. Oba adresáře musí mít nastavená práva pro zápis, jinak skončí načítání stránky s chybou.

Posledním – a velice důležitým – adresářem je `web`. Tento adresář je jako jediný z výše jmenovaných veřejně dostupný skrze http protokol. Jedná se o tzv. document_root. Obsah adresáře je následující:

- `css` – obsahuje soubory kaskádových stylů (*.css),
- `js` – skripty v jazyce JavaScript, soubor `funkce.js` je hlavním skriptem, který inicializuje různé funkcionality v aplikaci. Podadresář `plugins` obsahuje knihovny pro JavaScript, např. jQuery, jQuery UI, google-maps-tsp-solver apod.,
- `data` – adresář pro případné uživatelské soubory, aktuálně obsahuje pouze výsledné spojené soubory knihovny WebLoader (viz výše),
- `obrazky` – zdrojové obrázky pro webovou aplikaci,

- soubor `index.php` – „spouštěcí“ soubor, bude načten po vstupu na stránky aplikace. Má za úkol jediné – spustit soubor `bootstrap.php` (viz popis adresáře `app`), který následně zavede framework Nette,
- soubor `info.php` – pomocný soubor, který spouští příkaz phpinfo(), jenž zobrazí konfiguraci PHP na serveru,
- soubor `favicon.ico` – ikona stránky.

6.4 Moduly a funkčnost

Aplikace je rozdělena do několika funkčních celků, tzv. modulů. Jednotlivé moduly budou níže popsány a bude představena funkčnost celé aplikace.

6.4.1 Řidiči

Jednoduchý modul, který poskytuje správu řidičů systému. Je možné přidat nové řidiče, upravit stávajícího nebo řidiče kompletně odstranit. Řidič je definován pomocí jména a příjmení.

6.4.2 Automobily

Funkčně velice podobný modul jako výše popisovaní řidiči. Opět je umožněna správa automobilů – vytváření, editace a mazání. Navíc je zde umožněno upravovat stav tachometru. Po každé jízdě (dni) by měl být stav tachometru aktualizován. Tento stav je spolu s datem úpravy uložen a slouží k pozdějšímu přehledu využití vozidla (modul statistiky). Každý automobil je charakterizován svým názvem a státní poznávací značkou (SPZ), dále je nutné definovat nosnost vozidla a také rozměry přepravního prostoru.

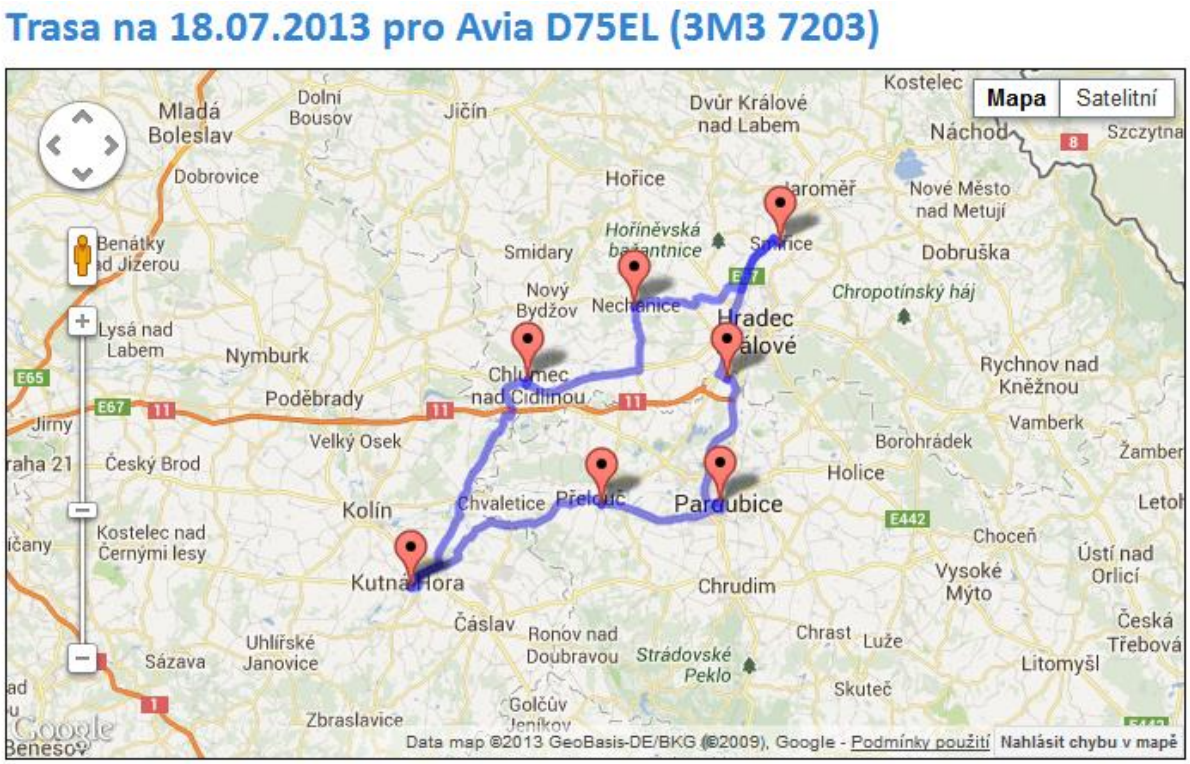
6.4.3 Objednávky

Modul objednávek umožňuje uživateli vkládat nové objednávky do systému, přidávat k nim určené balíky a měnit stavy objednávek – expedovaná, doručená apod. Samozřejmostí je možnost úpravy nebo smazání objednávky. Dostupné je také vyhledávání – lze dohledat objednávku dle zadaných slov, přičemž se hledá v adrese a názvu společnosti. Pro upřesnění hledání je možné zvolit den doručování. Pro každou novou objednávku je nutné zadat příjemce, jeho adresu, den doručování a také případnou cenu dobírky. Následně je zapotřebí pro každou objednávku vložit přepravované balíky, ty jsou definovány svými rozměry (šířka, výška, délka) a hmotností. Systém automaticky zjistí dle zadané adresy GPS souřadnice nového místa. Toto místo je možné v detailu objednávky zobrazit na mapě.

6.4.4 Trasy

Stěžejní modul, ve kterém se odehrává nejdůležitější část aplikace – generování tras. Na základě zadaných objednávek pro daný den se vygeneruje optimální trasa pro daná vozidla a řidiče. Samozřejmostí je možnost jednotlivé trasy editovat, popř. znovu vygenerovat s jinými vstupy. V detailu trasy je na mapě graficky znázorněna cesta, kterou má daný automobil projet. Seznam průjezdných bodů je zobrazen i v textové podobě – spolu

s informacemi o předpokládané vzdálenosti a času průjezdu. Na stejném místě lze také exportovat soubory s trasou do navigačních zařízení (TomTom, Garmin, iGO). Příklad vygenerované optimální mapy je zobrazen na následujícím obrázku.



Obrázek 18 – Příklad vygenerované optimální trasy

6.4.5 Statistiky

Modul určený především pro vedení společnosti, resp. pro manažery. Součástí je několik různých přehledových tabulek a grafů, které by měly zobrazit mnohé aspekty přepravní společnosti. Tato sekce je rozdělena na následující části:

- Trasy – statistika tras ukazuje, který den byly rozváženy objednávky. Zároveň je zobrazeno, kolik objednávek se v daný den rozvezlo a kolik automobilů zásilky rozváželo.

- Objednávky – prvotně jsou zobrazeny průměrné a statistické informace, tj. průměrná cena zásilek, průměrná velikost přepravovaných balíků a jejich hmotnost. Sekce je dále doplněna o graf zobrazující vývoj počtu objednávek v závislosti na aktuálním datu. Nakonec je zobrazena mapa Evropy, která přehledně zobrazuje počty objednávek v závislosti na cílové zemi.

- Automobily – zobrazuje graf využití automobilů vůči objednávkám (kolik objednávek daný automobil rozvezl). Dále pak graf pro každé vozidlo s vývojem najetých kilometrů – na první pohled je jasné, jak rychle se mění stav tachometru.

- Řidiči – obsahuje jednoduchý graf zobrazující, kolik dní daný řidič odjezdil a zároveň počet objednávek, které rozvezl.

6.4.6 Uživatelé

Modul určený pro řízení uživatelů a jejich oprávnění. Každý uživatel je identifikován pomocí svého nicku (přezdívky) a hesla. Je možné přidávat nové uživatele, upravovat stávající, měnit uživatelskou roli, popř. uživatele odstranit. Současně lze prohlížet přihlašovací log.

6.4.7 Nastavení

Jednoduchý modul umožňující měnit systémová nastavení. Aktuálně obsahuje pouze změnu výchozí polohy DEPA – místa, odkud automobily začínají svoji trasu.

6.5 Uživatelské role

V systému je definováno několik uživatelských rolí. V aktuální podkapitole jsou jednotlivé role popsány a je znázorněno, která role má jaká oprávnění. V aplikaci jsou k dispozici tyto uživatelské role:

- Admin – účet s nejvyšším oprávněním, může v systému dělat úplně vše.
- Dispečer – účet primárně určený k zadávání objednávek, generování denních tras apod. Mělo by se jednat o nejčastěji využívaný účet.
- Manažer – uživatelská role, která může přistupovat pouze k modulu statistiky.

Níže uvedená tabulka shrnuje uživatelské role a jejich jednotlivé přístupy v rámci modulů systému.

Tabulka 8 – Přehled oprávnění uživatelských rolí

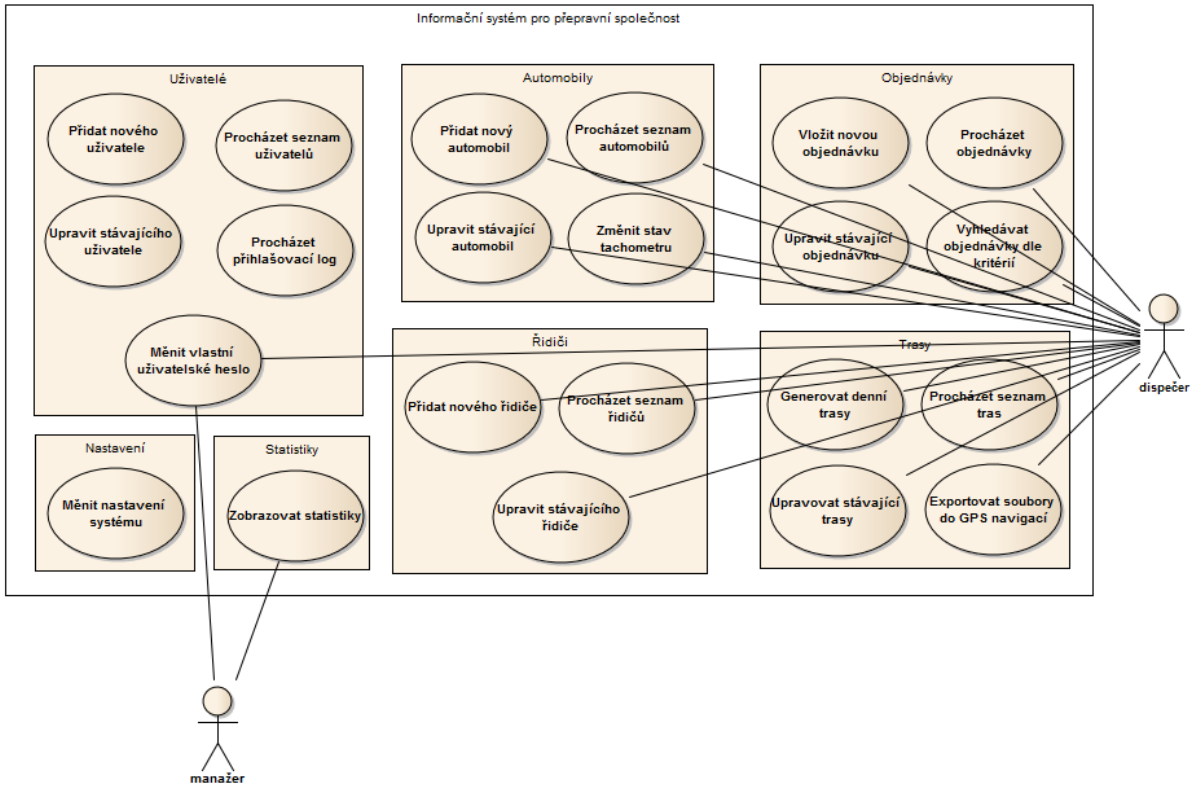
modul / role	admin	dispečer	manažer
řidiči	X	X	
automobily	X	X	
objednávky	X	X	
trasy	X	X	
statistiky	X		X
uživatelé	X		
nastavení	X		

V naprogramované aplikaci jsou součástí zdrojových kódů na CD v databázi definovány tři účty. Díky nim je možné aplikaci otestovat pod všemi dostupnými uživatelskými rolemi.

- přihlašovací jméno: `admin`, heslo: `admin`, role: admin,
- přihlašovací jméno: `dispecer`, heslo: `dispecer`, role: dispečer,
- přihlašovací jméno: `manazer`, heslo: `manazer`, role: manažer.

6.5.1 UseCase diagram

Na obrázku níže je zobrazen UseCase diagram, který zobrazuje případy užití pro danou uživatelskou roli. V diagramu jsou naznačeny hranice jednotlivých modulů a řádně pojmenovány. Z důvodu přehlednosti chybí v diagramu uživatelská role administrátora, ta má přístup ke všem případům užití.

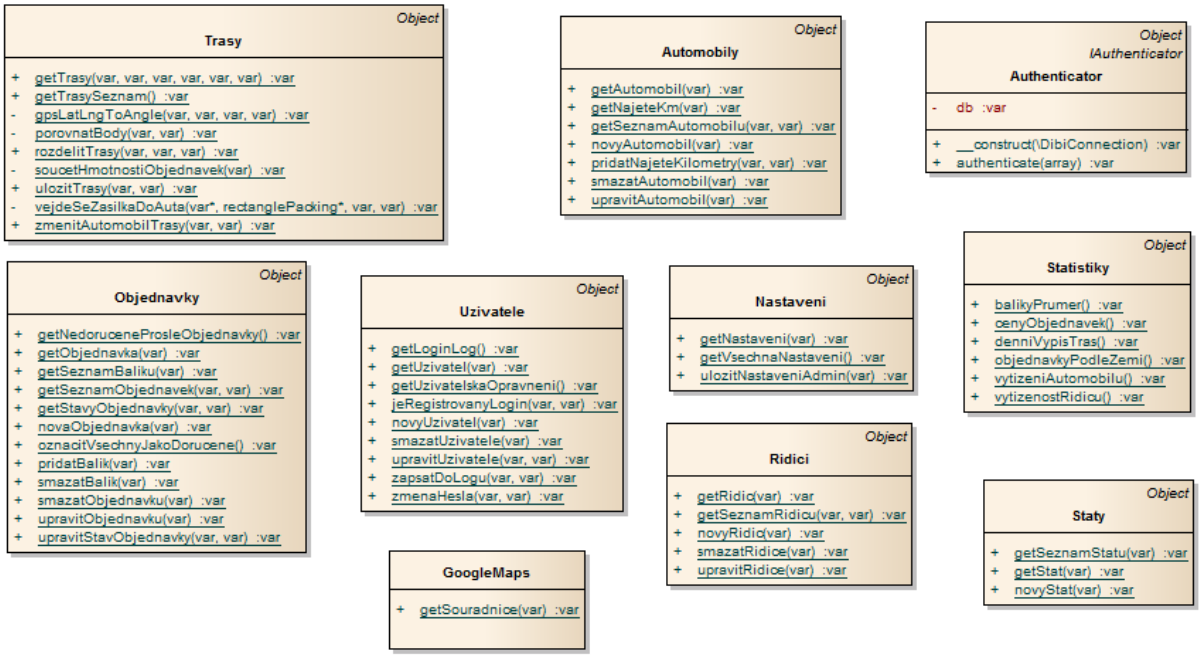


Obrázek 19 – UseCase diagram

6.6 UML diagram tříd

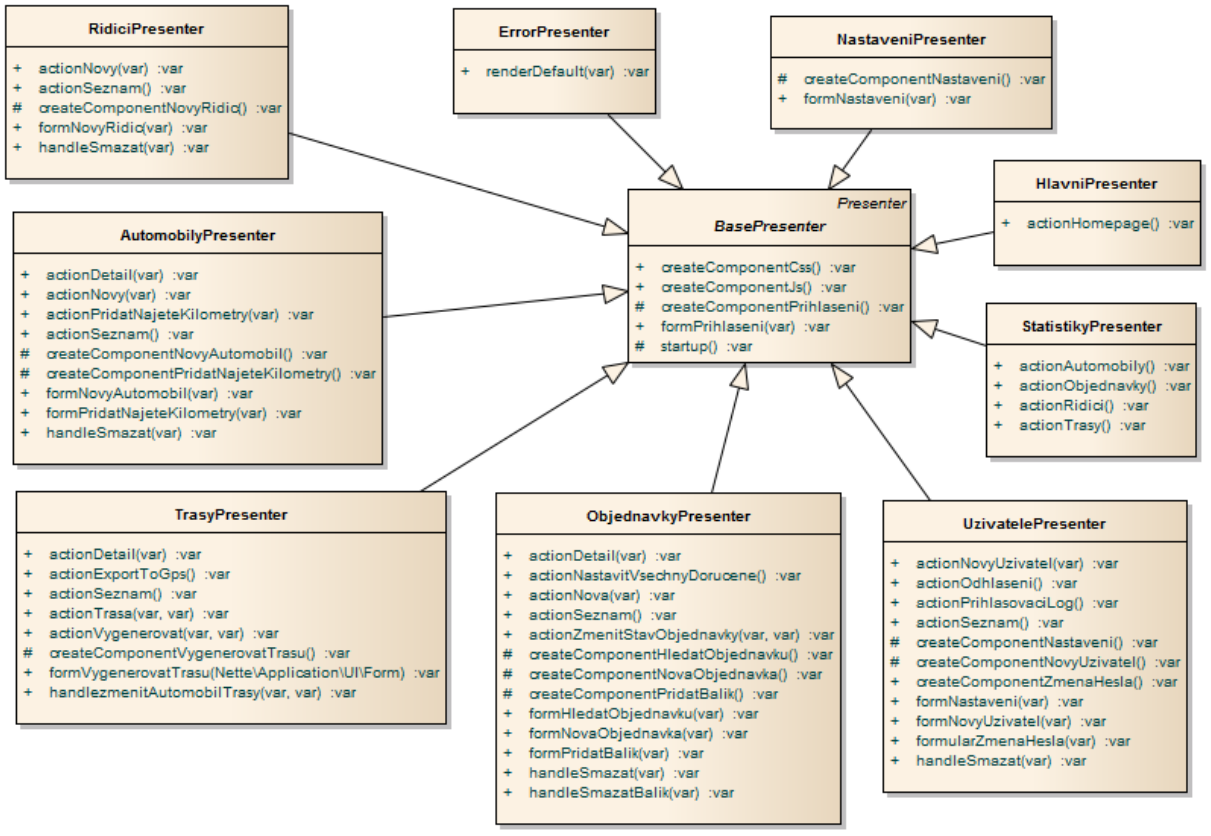
Pro přehled, jak je celá aplikace postavená a jaké třídy obsahuje, jsou níže uvedeny UML diagramy tříd. Jednotlivé třídy nebudou jednotlivě popisovány, pro pochopení struktury plně postačují dané diagramy tříd.

Jako první je zobrazen modelový diagram tříd. Pozn.: zobrazeny jsou pouze třídy samotné aplikace, nikoliv třídy frameworku a pomocných knihoven.



Obrázek 20 – UML diagram modelů

Dále následuje diagram tříd presenterů – v podstatě kopíruje jednotlivé modely. Všimněme si základní třídy „BasePresenter“, ze které všechny další presentery dědí. Tato třída obsahuje úkony, které je nutné provádět vždy – např. ověření přihlášeného uživatele.



Obrázek 21 – UML diagram presenterů

6.7 Předpokládaný scénář použití

Přestože je aplikace zcela volně použitelná a přihlášenému uživateli poskytuje neomezeně všechny funkce hodné jeho uživatelské roli, autor aplikace předpokládá s jistým scénářem používání. Na základě tohoto scénáře je také aplikace stavěna a přizpůsobena rychlému použití.

Předpokládejme, že v systému bude každý pracovní den přihlášen jeden uživatel s uživatelskou rolí „dispečer“. Tento dispečer má na starosti kompletní správu objednávek, tj. vkládání nových objednávek, generování tras pro řidiče a zapisování stavu tachometru pro jednotlivá vozidla. Předpokládá se, že během pracovní doby dispečer přijímá objednávky (telefonicky, emailem, atd.) a zadává je do systému. Na začátku pracovního dne je však nutné udělat několik úkonů – (i) změnit stav objednávek z předchozího dne jako doručené, (ii) upravit stavy tachometrů automobilů v systému, (iii) vygenerovat trasy pro aktuální den. Jelikož je výše uvedený scénář předpokládán, aplikace po přihlášení na úvodní stránce umožňuje hromadně změnit stavy „včerejších“ objednávek. Automaticky jsou vybrány objednávky, které byly expedovány v minulých dnech, a pokud nemají stav „doručeno“ nebo „stornováno“, vypíší se a je možné je hromadně označit za doručené. Zároveň se zobrazí upozornění, že by bylo vhodné aktualizovat stavy tachometrů vozidel. V bodovém vyjádření by předpokládaný scénář vypadal následovně:

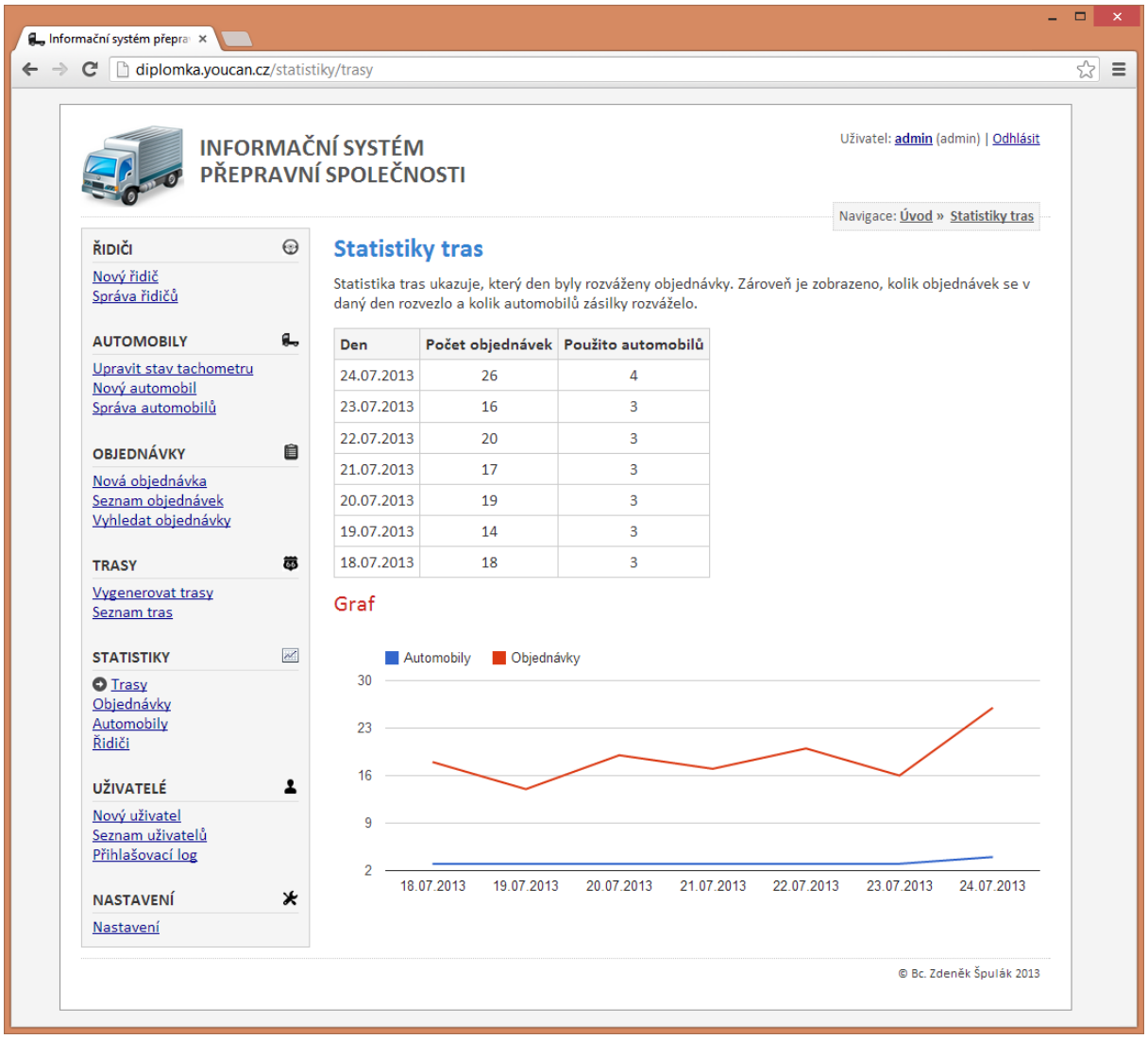
- ráno (po příchodu do práce):
    - nastavit doručeným objednávkám stav „doručené“,
    - upravit stavy tachometrů vozidel,
    - vygenerovat trasy pro aktuální den,
    - předat trasy řidičům, popř. nahrát trasy do GPS navigací,
- během dne:
    - zaznamenávat do systému objednávky od zákazníků.

Výše uvedený scénář je předpokládaný pro uživatelskou roli „dispečer“. Další role – „admin“ a „manažer“ žádný předpokládaný scénář nemají, neboť se nepředpokládá jejich časté přihlašování do systému. Administrátorem se myslí osoba z IT oddělení, která do systému nepotřebuje za běhu nijak zásadně zasahovat – maximálně přidat nového uživatele systému. U manažerů se počítá s nahodilým přístupem – několikrát do měsíce. Ti si budou pro interní potřeby zobrazovat pouze statistiky a nebudou žádným způsobem do systému zasahovat.

6.8 Rozvržení a vzhled aplikace

Rozložení webové stránky a celý layout byl volen s ohledem na maximální jednoduchost a především přehlednost. Jelikož bude aplikace využívána denně, je zásadní prioritou rychlá

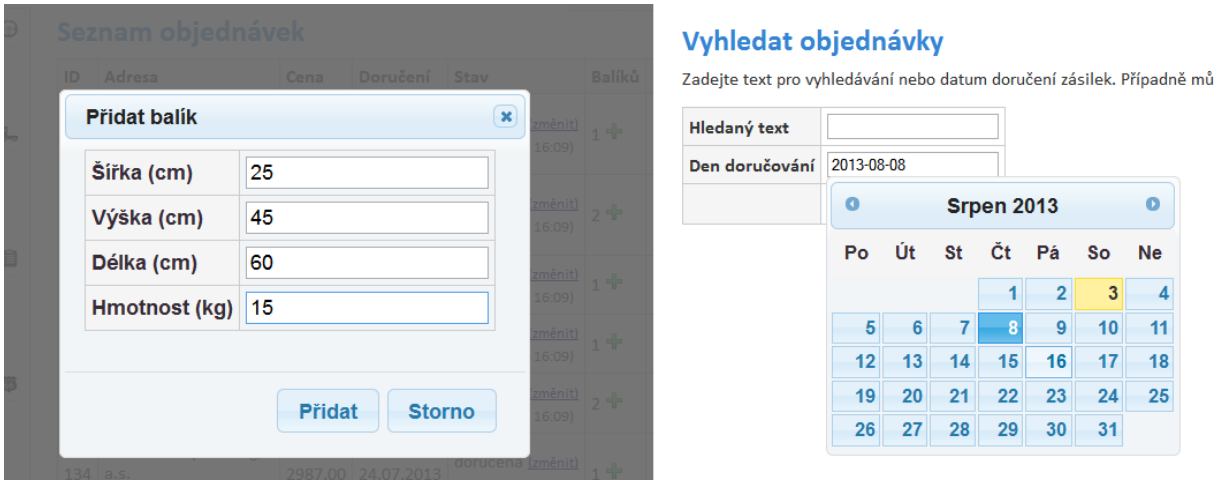
orientace na stránce a zobrazení maxima informací bez nutnosti skrolování. Po grafické stránce byly voleny především neutrální odstíny šedé, tedy nijak křiklavé nebo oči unavující barvy. Na obrázku níže je náhled celé aplikace.



Obrázek 22 – Náhled vytvořené aplikace

Layout stránky je rozdělen do několika funkčních bloků. V horní části se nachází hlavička, kde nalezneme logo a název aplikace. Současně je uživatel v této části informován o přihlášení a aktuálním umístění – to díky navigaci. Levý postranní pruh zaujímá menu. To je logicky rozděleno podle modulů a akcí. Menu se dynamicky mění v závislosti na uživateli, resp. jeho uživatelské roli. Pro každý modul je vyobrazena malá ikona, která usnadní orientaci v menu. Podobnou funkci má také malá šipka, která upozorňuje na aktuálně zvolenou stránku (na obrázku výše je u Statistiky – Trasy). Střední část, a také největší, zaujímá hlavní obsah. Ten je měněn v závislosti na aktuálně vybrané stránce. V náhledu výše jsou zobrazeny statistiky tras. V dolní části – patičce – se nachází již pouze podpis a copyright.

Při práci s aplikací je využíváno několik doplňků, které uživateli velice usnadní orientaci a obecně zrychlí práci, a tím pádem zvýší i produktivitu. Pro některé funkcionality je využíváno modální dialogové okno. Například u přidání nového balíku v seznamu objednávek nemusí uživatel složitě přecházet do detailu objednávky, ale přímo v seznamu klikne na „přidat“ a vyskočí formulář s přidáním nového balíku. Dalším použitým doplňkem je tzv. DatePicker, česky výběr data. Aby nemusel uživatel složitě vypisovat datum ve formátu RRRR-MM-DD, je po kliknutí na požadované pole zobrazen kalendář. Výběr data je následně pohodlně zadán myší bez nutnosti použít klávesnici. Reálný náhled obou popsaných doplňků lze zhlédnout na následujícím obrázku.



Obrázek 23 – Ukázka použitých doplňků (vlevo modální dialog, vpravo DatePicker)

Při vývoji šablony byl kladen důraz na totožné zobrazení napříč všemi prohlížeči. Nesmí nastat případ, kdy uživatel na starším počítači se starším internetovým prohlížečem nebude schopen aplikaci obsluhovat. Proto byl vzhled aplikace důkladně testován v mnoha prohlížečích napříč operačními systémy. Testování probíhalo v operačních systémech Windows (XP, Vista, 7, 8, 2008 server), Linux (Debian, Ubuntu, SuSe, FreeBSD) a MacOS X (10.8). Přičemž byly testovány nejpoužívanější prohlížeče v mnoha verzích – Opera (10.63 až 15.0), Chrome (10 až 28), Mozilla Firefox (1.5 až 23.0), Microsoft Internet Explorer (5.5 až 10) a Safari (5.0.3 až 6.0.5). Otestováno bylo i několik méně známých prohlížečů, např. IceWeasel, SeaMonkey, Arrora, Dillo, Kazehakase, Epiphany nebo Konqueror. Závěrem testování lze prohlásit, že ačkoliv se webové stránky nezobrazují úplně stejně, odchylky jsou minimální. Co je však hlavní – ve všech výše zmíněných prohlížečích lze plnohodnotně aplikaci obsluhovat.

6.9 Řešení nakládky zboží

Při výpočtu rozvážky jednotlivých objednávek (balíků) je třeba brát v úvahu několik faktorů. Primárními informacemi je počet objednávek a počet dostupných vozidel, resp. počet řidičů. V případě, že chceme optimalizovat trasu, jsou důležité i další informace – vzdálenost mezi jednotlivými zastávkami a směr od výchozího místa (depa). Při výpočtu je nutné uvažovat hmotnost přepravovaných zásilek a také jejich rozměry. V případě, že je

nosnost, resp. kapacita ložné plochy, vozidla přesažena, je nutné takovou objednávku vložit do nového (následujícího) vozidla.

Výpočet tras pro aktuální den je velice komplexní a složitý problém, ve kterém je třeba uvažovat mnoho různých faktorů, omezení apod. Pro účely této práce byl navržen algoritmus, který vše výše popsané splňuje – uvažuje rozměry a hmotnosti balíků, uvažuje počet automobilů a řidičů a zároveň se snaží pro každý automobil vypočítat optimální trasu. Stručný popis a fungování algoritmu je popsáno v následujících bodech, přičemž vstupem do algoritmu je seznam automobilů, řidičů a objednávek.

- kontrola zadaných údajů – počet řidičů se musí rovnat počtu automobilů a zároveň musí být definován alespoň jeden automobil a jeden řidič,
- výběr platných objednávek – nesmí být ve stavu doručená nebo stornovaná,
- pro místo doručení každé z objednávek se vypočítá úhel (směr) vůči výchozímu bodu (depa),
- objednávky jsou setříděny podle vypočítaného úhlu,
- výpočet počtu objednávek na jeden automobil (počet objednávek / počet automobilů),
- cyklus procházející seznam objednávek,
    - zkontroluje se seznam odložených objednávek, které se v předchozích průchodech cyklu nevešly do automobilu – pokud se změnil automobil, vloží se do seznamu objednávek,
    - pokud je počet objednávek pro daný automobil překročen, změní se aktuální automobil,
    - kontrola, zda se daná objednávka (balíky) vejde do aktuálního vozidla – kontrola rozměrů a hmotnosti. Pokud se nevejde, přidá se objednávka do seznamu odložených objednávek,
    - vše je v pořádku – aktuálnímu automobilu se přiřadí daná objednávka,
- algoritmus vrací proměnnou `$automobily`, která nese informaci o jednotlivých automobilech a především o tom, která místa (objednávky), má daný automobil navštívit,
- pokud zadávající uživatel povolí, každé z objednávek se nastaví stav na „expedovaná“,
- pro každý automobil je možné následně vygenerovat optimální trasu jízdy – problém obchodního cestujícího.

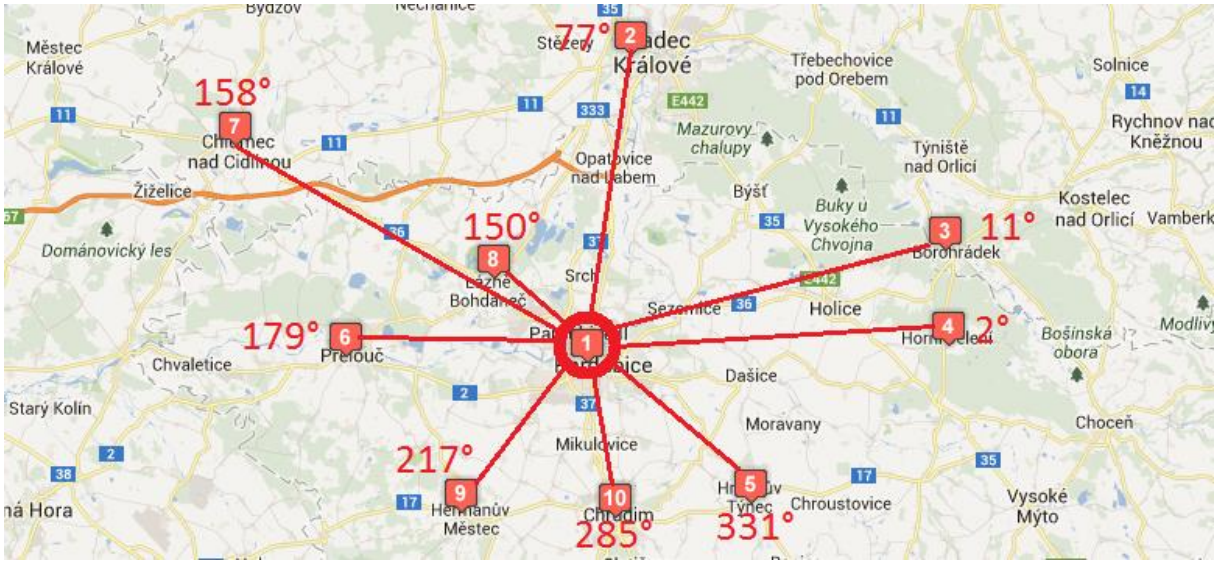
Detailnější popis dílčích algoritmů je uveden v následujících podkapitolách. Je však třeba uvést, že celý výše popisovaný algoritmus není úplně dokonalý a má drobné nedostatky. Hlavním problémem je stejný (podobný) počet objednávek pro každý automobil. V případě hodně rozdílného umístění jednotlivých lokací může algoritmus vypočítat nepříliš ideální rozdělení pro jednotlivá vozidla. Nicméně pro demonstraci funkčnosti je tento algoritmus zcela dostačující a vrací uspokojivé výsledky. V případě nutnosti není problém algoritmus snadno nahradit jiným – komplexnějším – řešením.

6.9.1 Rozdělení objednávek na základě směru od depa

Jak již bylo v základním algoritmu naznačeno, dílčí objednávky, resp. jejich lokace, jsou vyhodnoceny na základě polohy vůči výchozímu bodu – depu. Všechny souřadnice jsou udávány ve formátu WGS 84 (World Geodetic System 1984), souřadnice Fakulty elektrotechniky a informatiky jsou v tomto formátu následující: `50.033451`, `15.76714`. Jelikož se jedná o klasické `X;Y` souřadnice, lze s nimi velice snadno pracovat. Víme-li souřadnice depa a požadované lokality, snadno dopočítáme, kterým směrem je daná lokalita (objednávka) situována. Na níže uvedeném zdrojovém kódu je znázorněna metoda pro výpočet.

```
private static function gpsLatLngToAngle($stred_lat, $stred_lng,
$bod_lat, $bod_lng) {
    $x = $bod_lng - $stred_lng;
    $y = $bod_lat - $stred_lat;

    $deg = rad2deg(atan2($y, $x));
    return $deg < 0 ? (360 + $deg) : $deg;
}
```

Argumenty metody jsou souřadnice středu a souřadnice vypočítávaného bodu. Metoda vrací úhel směru, kterým je požadovaný bod situován. Následující obrázek znázorňuje demonstrativní příklad. Středem (depo) jsou Pardubice (bod č. 1) a následně jsou pro každý bod na mapě (2-10) spočítány úhly od středu. Pozn.: úhly jsou na obrázku zaokrouhleny na celé jednotky.



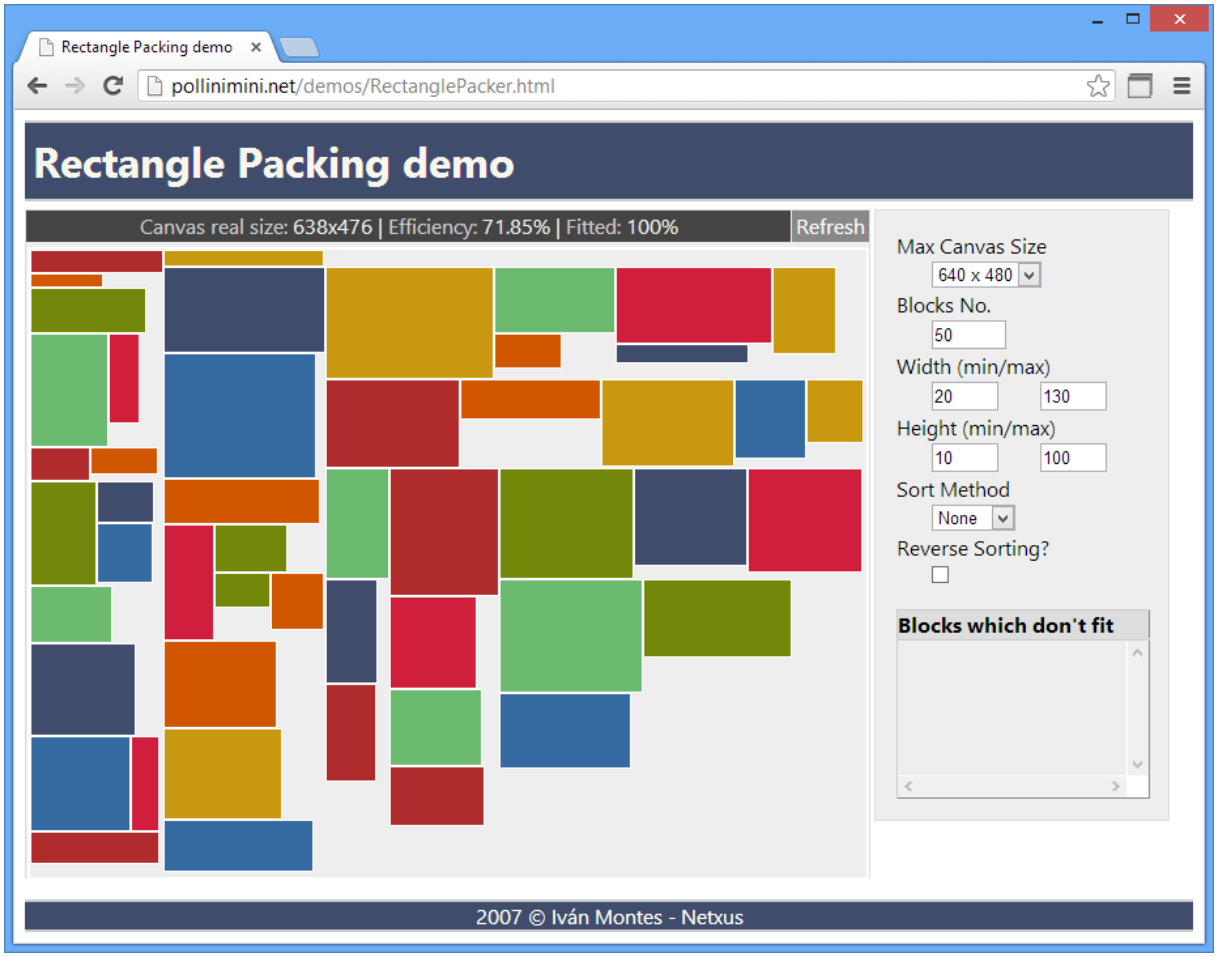
Obrázek 24 – Demonstrace ohodnocování bodů na mapě podle úhlu

6.9.2 Testování hmotnosti a rozměru zásilek

Při průchodu jednotlivých iterací hlavního algoritmu (viz výše) se pro každý automobil uchovává aktuální naložení zásilkami. Jelikož nosnost a rozměry automobilů nejsou neomezené, je nutné kontrolovat, zda je možné další zásilku do automobilu naložit – zda se nepřekročí nosnost automobilu nebo rozměry. Sledovat omezení nosnosti je poměrně snadné. Pro každý automobil je sčítána hmotnost jednotlivých balíků, a pokud dojde na případ, kdy součet hmotností překročí nosnost vozidla, nelze zásilku do automobilu naložit.

Naopak testování rozměrů balíků a vozidla je již značný problém. Obecně je tento problém popsán v jiné kapitole – Container loading problem. Jelikož je řešení problému ve trojdimenzionálním prostoru (3D) velice složité, je v této práci řešeno pouze nakládání zásilek ve dvou rozměrech, tedy 2D. Jednoduše řečeno – atribut výšky je ignorován. Konkrétně se v této praktické části řeší Container loading problem typu balení obdélníků v obdélníku, tedy zásilky (obdélníky) v prostoru vozidla (obdélník). Algoritmus pro řešení problému je převzat od autora Ivána Montese, který na svém webu (`pollinimini.net`) uveřejnil kód pro tzv. „Rectangle Packing (2D packing)“. Zdrojový kód je psaný v jazyce JavaScript a je dostupný pod volnou licencí – Creative Commons 2.5. Pro potřeby této práce byl původní kód přepsán do programovacího jazyka PHP a nasazen do aplikace. Princip fungování algoritmu je následující – pro každý vkládaný obdélník se volá metoda `findCoords($w, $h)` s parametrem šířky a délky. Tato metoda vytváří binární strom z již vložených prvků, přičemž jednotlivé prvky stromu nesou informaci o šířce, resp. délce, vložených obdélníků. Při vkládání nového prvku do stromu se strom prochází a vyhledává se volné místo, které svými rozměry vyhoví novému prvku. Pokud takové volné místo není nalezeno, vrací metoda hodnotu NULL. V opačném případě je prvek zařazen do struktury stromu a návratovou hodnotou je pole se souřadnicemi `X` a `Y`, na kterých je vkládaný prvek umístěn.

Na následujícím obrázku je demonstrován popisovaný algoritmus. Jedná se o původní JavaScriptovou verzi vytvořenou autorem Iványem Montesem.



Obrázek 25 – Ukázka funkčnosti Rectangle Packing

6.10 Export dat do GPS navigací

Vytvořená aplikace podporuje export optimalizované trasy do navigačního zařízení. Aktuálně je podporováno několik typů navigací, které jsou blíže popsány.

6.10.1 Soubor ITN

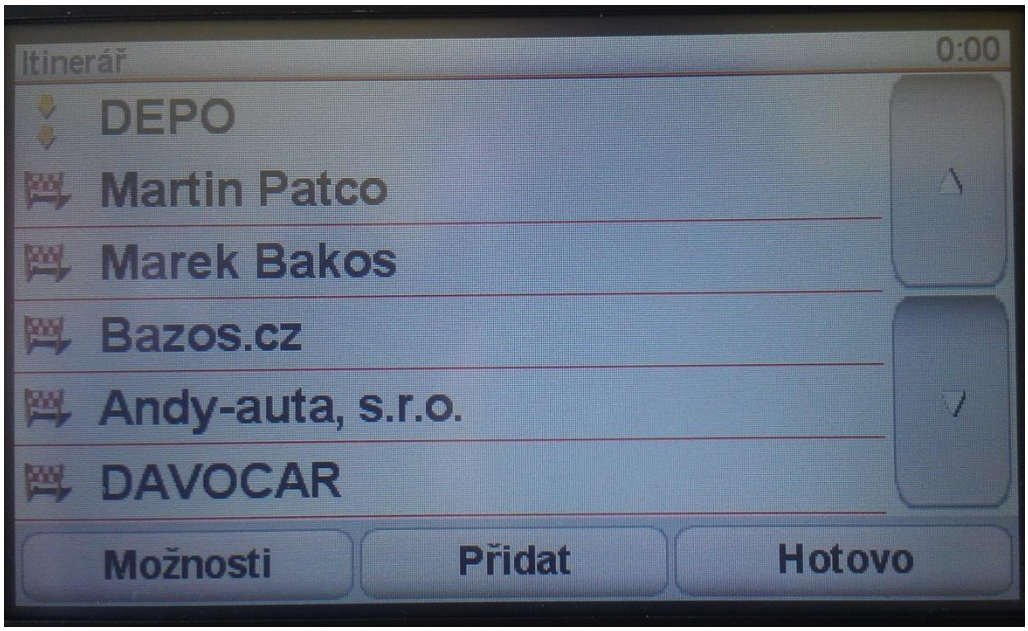
Soubor s příponou *.itn je používán v navigacích firmy TomTom. Jedná se o soubor itineráře, tedy seznamu bodů, které mají být projety. Jedná se o prostý textový soubor, který obsahuje:

- zeměpisnou délku ve formátu WGS84 (na 100.000 stupňů),
- zeměpisnou šířku ve formátu WGS84 (na 100.000 stupňů),
- název bodu,
- značka – může nabývat několika hodnot, např. počáteční bod, průjezdný bod apod.

Jednotlivé hodnoty jsou od sebe odděleny znakem | (roura), přičemž každý nový bod je oddělen novým řádkem. Příklad souboru je uveden níže.

```
1576731|5003341|DEPO|4|
1567104|4972061|Martin Patco|2|
1652207|4917419|Marek Bakos|2|
1719085|4951532|Bazos.cz|2|
1468574|4941334|Andy-auta, s.r.o.|2|
1463877|4967348|DAVOCAR|2|
1444281|5005164|HYPERINZERCE|2|
1446785|5010221|AC DODAVKY, s.r.o.|2|
1444811|5013719|1 1 Nejlepsi autopujcka s.r.o.|2|
1576731|5003341|DEPO|2|
```

Po nahrání do samotného navigačního zařízení a načtení souboru vypadá soubor itineráře následovně.



Obrázek 26 – Zobrazení itineráře v navigaci TomTom

6.10.2 Soubor UPOI

Formát souboru *.upoi je využíván navigační aplikací iGO. Jedná se o seznam bodů zájmu, tzv. POI (point of interest). Podobně jako u souboru ITN, i u formátu UPOI se jedná o prostý text. Bohužel autor práce nenalezl oficiální dokumentaci k tomuto formátu, proto byla tvorba výstupního souboru závislá na hledání z neoficiálních zdrojů a různých komunit. Stejný seznam bodů jako v příkladu souboru ITN je pro formát UPOI následující.

```
0|ITN Converter|DEPO||50.033410|15.767310|||||||||
1|ITN Converter|Martin Patco||49.720610|15.671040|||||||||
2|ITN Converter|Marek Bakos||49.174190|16.522070|||||||||
3|ITN Converter|Bazos.cz||49.515320|17.190850|||||||||
4|ITN Converter|Andy-auta, s.r.o.||49.413340|14.685740|||||||||
5|ITN Converter|DAVOCAR||49.673480|14.638770|||||||||
6|ITN Converter|HYPERINZERCE||50.051640|14.442810|||||||||
```

```
7|ITN Converter|AC DODAVKY, s.r.o.||50.102210|14.467850|||||||||
8|ITN Converter|1 1 Nejlepsi autopujcka
s.r.o.||50.137190|14.448110|||||||||
9|ITN Converter|DEPO||50.033410|15.767310|||||||||
```

Oddělení bodů je realizováno opět pomocí nových řádků, stejně tak oddělení jednotlivých hodnot pomocí roury. Uvedený příklad formátu souboru je dosti výmluvný a není dále jej nijak extra komentovat. Za zmínku stojí pouze fakt, že nevyplněné informace – několik znaků roura vedle sebe – se autorovi nepodařilo dohledat. Nicméně demonstrovaný příklad bezproblémově v navigaci funguje.

6.10.3 Soubor GPX

Poměrně univerzální formát souborů, který je založen na formátu XML. Název je odvozen od „GPS eXchange Format“ a mj. jej dokáží přečíst navigace od společnosti Garmin. Na rozdíl od výše popisovaných formátu, je GPX mnohem univerzálnější a existuje k němu rozsáhlý manuál. Díky formátování v XML existuje více různých verzí a „mutací“ s dodatečnými informacemi. Příklad, který kopíruje průjezdní body z předchozích typů, je uveden níže. Pozn.: z důvodu přehlednosti je zdrojový soubor zkrácen, nicméně pro demonstraci formátu by měl plně dostačovat.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<gpx version="1.0" creator="ITN Converter"xmlns:xsi="…">

    <time>2013-07-18T11:09:36Z</time>
    <bounds minlat="49.174190" minlon="14.442810" maxlat="50.137190"
maxlon="17.190850"/>

    <rte>
        <name>4e5-4510-2013-07-12-1</name>

        <rtept lat="50.033410" lon="15.767310">
            <name>DEPO</name>
        </rtept>

        <rtept lat="49.720610" lon="15.671040">
            <name>Martin Patco</name>
        </rtept>

        <rtept lat="49.174190" lon="16.522070">
            <name>Marek Bakos</name>
        </rtept>

        <rtept lat="49.515320" lon="17.190850">
            <name>Bazos.cz</name>
        </rtept>

        <rtept lat="49.413340" lon="14.685740">
            <name>Andy-auta, s.r.o.</name>
        </rtept>

</rte>
</gpx>
```

6.11 Licence použitých komponent

V rámci aplikace je použito několik komponent a doplňků třetích stran. Je proto nutné uvést jejich licence. Seznam je uveden v následující tabulce.

Tabulka 9 – Seznam licencí využívaných doplňků a komponent

Komponenta / doplněk	Licence
Nette	BSD
Dibi	BSD
WebLoader	MIT
RectanglePacker	Creative Commons Attribution 2.5 Generic
JssMin	MIT
CssMin	MIT
Google Maps API	Free licence s limitací požadavků registrovaná pro doménu diplomka.youcan.cz (https://developers.google.com/maps/licensing?hl=cs)
FancyBox	Creative Commons Attribution-NonCommercial 3.0
Google maps TSP Solver	MIT
jQuery	MIT
jQuery UI	MIT
jQuery JSON	MIT
jQuery Redirect	MIT
Obrázky použité v aplikaci	Obrázky jsou stažené z fotobanky ikon – iconfinder.com. Přičemž všechny jsou pod volnou licencí, např. Free for personal use nebo LGPL.

Závěr

Cílem této diplomové práce bylo vytvoření funkční aplikace pro přepravní společnost. Přičemž součástí byl teoretický rozbor dostupných řešení na trhu a různých variant řešení IS pro přepravní společnost. Ačkoliv se zpočátku zdál úkol jako nepříliš složitý, opak byl pravdou. Během tvorby aplikace a hledání teoretických podkladů se vyskytlo nemálo problémů. Hlavní překážkou byl výpočet optimální trasy, kterou nebylo možné vypočítávat pomocí algoritmů, s nimiž jsme se seznámili v rámci studia. Po dohledání algoritmu obchodního cestujícího, který zadaný problém řeší, však nastaly další komplikace – jak tento algoritmus implementovat v rámci webového prostředí. Po zdárném vyřešení však nastalo další hledání, a to algoritmu pro optimální nakládku balíků do automobilu. Jelikož byl i tento problém vyřešen – včetně teoretického rozboru – lze prohlásit, že všechny cíle diplomové práce byly splněny. Podobně je tomu i v praktické části, v rámci které byl vytvořen zcela nový unikátní projekt splňující veškeré náležitosti informačního systému pro přepravní společnost.

Nad rámec zadání bylo do aplikace doprogramováno několik funkcí navíc. Jedná se především o podporu exportu vypočítané trasy do několika typů navigačních zařízení. A také statistický modul aplikace, díky kterému může management společnosti snadno vidět přehledové grafy a tabulky vytíženosti automobilů, řidičů aj. Navíc je také logovací modul, který monitoruje seznam přihlášení jednotlivých uživatelů aplikace. Samozřejmostí je vysoká bezpečnost celého řešení, které je odolné vůči útokům apod. Uživatelská hesla jsou navíc uložena v podobě hashe, nikoliv jako prostý text, což zásadně snižuje dopad útoku v případě odcizení databáze.

Od počátku byla tvorba aplikace směřována na jednoduchost – a to jak po stránce ovladatelnosti, tak i po stránce architektury a databáze. Lze prohlásit, že aplikace je velice snadno rozšiřitelná a upravitelná pro konkrétní požadavky zadavatele. Možných příkladů rozšíření aplikace je hned několik. Velice užitečnou funkcí by mohla být integrace online sledování vozidel pro přesnější představu vytíženosti automobilů a řidičů. Navržená aplikace počítá pouze s jedním výchozím bodem (depem), proto by mohlo být vhodným návrhem na rozšíření aplikace toto „omezení“ odstranit. Posledním tipem možné rozšiřitelnosti aplikace je avízo pro zákazníka – např. napojit k aplikaci službu pro zasílání informativních SMS.

V současné době (srpen 2013) probíhá testování aplikace v menší rodinné firmě, která rozváží svoje výrobky po mnoha prodejnách v rámci České republiky. Tato firma má o vytvořenou aplikaci veliký zájem a s největší pravděpodobností bude v budoucnu reálně využívána, neboť předběžné výsledky ukazují na značnou úsporu nákladů na dopravu – cca až 20 %.

Literatura

Jmenné prostory a další novinky v PHP 5.3. *Root.cz* [online]. 2009 [cit. 2013-08-13]. Dostupné z: http://www.root.cz/clanky/php-5-3/

Quick Start. *Dibi* [online]. 2013 [cit. 2013-08-13]. Dostupné z: http://dibiphp.com/cs/quick-start

Obecný mapový podklad. *Seznam nápověda* [online]. 2013 [cit. 2013-08-13]. Dostupné z: http://napoveda.seznam.cz/cz/mapy/mapove-podklady/obecna/

Mapy API verze 4.8 – Hanzelka a Zikmund. *API Mapy.cz* [online]. 2013 [cit. 2013-08-13]. Dostupné z: http://api4.mapy.cz/

Nokia Mapy a Bing Mapy se sbližují: co bude nového?. *Mobilenet.cz* [online]. 2012 [cit. 2013-08-13]. Dostupné z: http://mobilenet.cz/clanky/nokia-mapy-a-bing-mapy-se-sblizuji-co-bude-noveho-8713

Google Maps JavaScript API v3. *Google Developers* [online]. 2013 [cit. 2013-08-13]. Dostupné z: https://developers.google.com/maps/documentation/javascript/

Yahoo! Maps Web Services. *Yahoo! Maps Developer APIs* [online]. 2013 [cit. 2013-08-13]. Dostupné z: http://developer.yahoo.com/maps/

Nokia Here vs Google Maps – feature comparison. *Softonic* [online]. 2012 [cit. 2013-08-13]. Dostupné z: http://features.en.softonic.com/nokia-here-vs-google-maps-feature-comparison

Nokia zapojila rozšířenou realitu přímo do Here map. *MobilMania.cz* [online]. 2013 [cit. 2013-08-13]. Dostupné z: http://www.mobilmania.cz/nokia-zapojila-rozsirenou-realitu-primo-do-here-map/a-1323787/default.aspx

Main Page. *OpenStreetMap Wiki* [online]. 2013 [cit. 2013-08-13]. Dostupné z: http://wiki.openstreetmap.org/wiki/Main_Page

JQuery API. *JQuery API* [online]. 2013 [cit. 2013-08-13]. Dostupné z: http://api.jquery.com/

JQuery UI 1.10 API Documentation. *JQuery UI API Documentation* [online]. 2013 [cit. 2013-08-13]. Dostupné z: http://api.jqueryui.com/

Rychlý a pohodlný vývoj webových aplikací v PHP. *Nette Framework* [online]. 2013 [cit. 2013-08-13]. Dostupné z: http://nette.org/cs/

Dokumentace. *Nette Framework* [online]. 2013 [cit. 2013-08-13]. Dostupné z: http://doc.nette.org/cs/

Software pro dopravní firmy. *PRŮMYSLOVÝ KATALOG Česká republika* [online]. 2013 [cit. 2013-08-13]. Dostupné z: http://www.logismarket.cz/software-dopravni-firmy/947645470-cp.html

Using EXPLAIN PLAN. *Oracle® Database Performance Tuning Guide* [online]. 2008 [cit. 2013-08-13]. Dostupné z: http://docs.oracle.com/cd/B19306_01/server.102/b14211/ex_plan.htm

10 sql tips to speed up your database. *Cats who Code* [online]. 2010 [cit. 2013-08-13]. Dostupné z: http://www.catswhocode.com/blog/10-sql-tips-to-speed-up-your-database

MySQL Performance Tuning: Top 10 Tips. *Slideshare* [online]. 2013 [cit. 2013-08-13]. Dostupné z: http://www.slideshare.net/osscube/mysql-performance-tuning-top-10-tips

Optimalizátor MySQL. *Administrace databázových systémů* [online]. 2009 [cit. 2013-08-13]. Dostupné z: http://www.cecak.cz/fel/dba/referaty/mysql/oprimalizator_mysql

Itinerary files. *TomTom Navigator SDK* [online]. 2004 [cit. 2013-08-13]. Dostupné z: http://www.tomtom.com/lib/doc/ttnavsdk3_manual.pdf

HYNEK, Josef. *Genetické algoritmy a genetické programování*. 1. vyd. Praha: Grada, 2008, 182 s. ISBN 978-80-247-2695-3.

ŽÁK, David. *Architektury a techniky databázových systému*. Pardubice, 2011. Přednášky. Univerzita Pardubice.

PHP. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-08-13]. Dostupné z: http://cs.wikipedia.org/wiki/PHP

Bing Maps. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-08-13]. Dostupné z: http://en.wikipedia.org/wiki/Bing_Maps

Google Maps. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-08-13]. Dostupné z: https://en.wikipedia.org/wiki/Google_Maps

Here (Nokia). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-08-13]. Dostupné z: https://en.wikipedia.org/wiki/Here_%28Nokia%29

MySQL. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-08-13]. Dostupné z: https://cs.wikipedia.org/wiki/MySQL

JavaScript. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-08-13]. Dostupné z: http://en.wikipedia.org/wiki/JavaScript

JQuery. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-08-13]. Dostupné z: http://en.wikipedia.org/wiki/JQuery

JQuery UI. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-08-13]. Dostupné z: http://en.wikipedia.org/wiki/JQuery_UI

Nette Framework. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-08-13]. Dostupné z: http://cs.wikipedia.org/wiki/Nette_Framework

Dijkstra's algorithm. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-08-13]. Dostupné z: http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

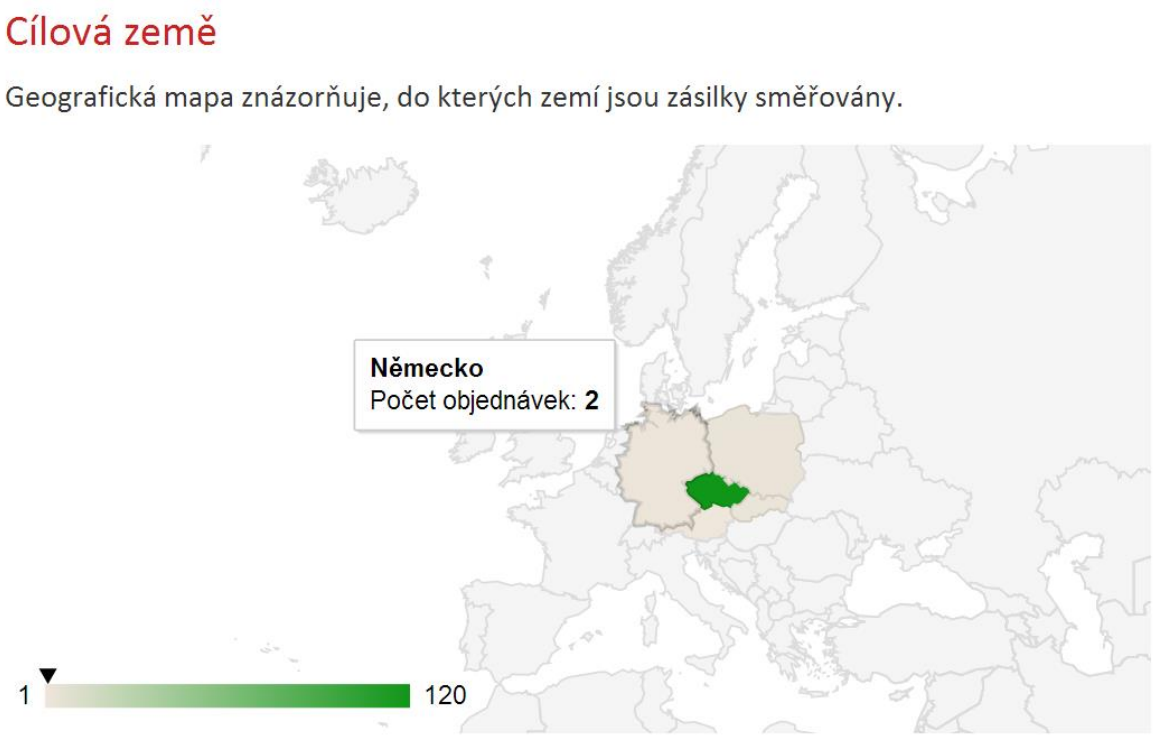
A*. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-08-13]. Dostupné z: http://cs.wikipedia.org/wiki/A*

Bellmanův-Fordův algoritmus. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-08-13]. Dostupné z: http://cs.wikipedia.org/wiki/Bellman%C5%AFv-Ford%C5%AFv_algoritmus

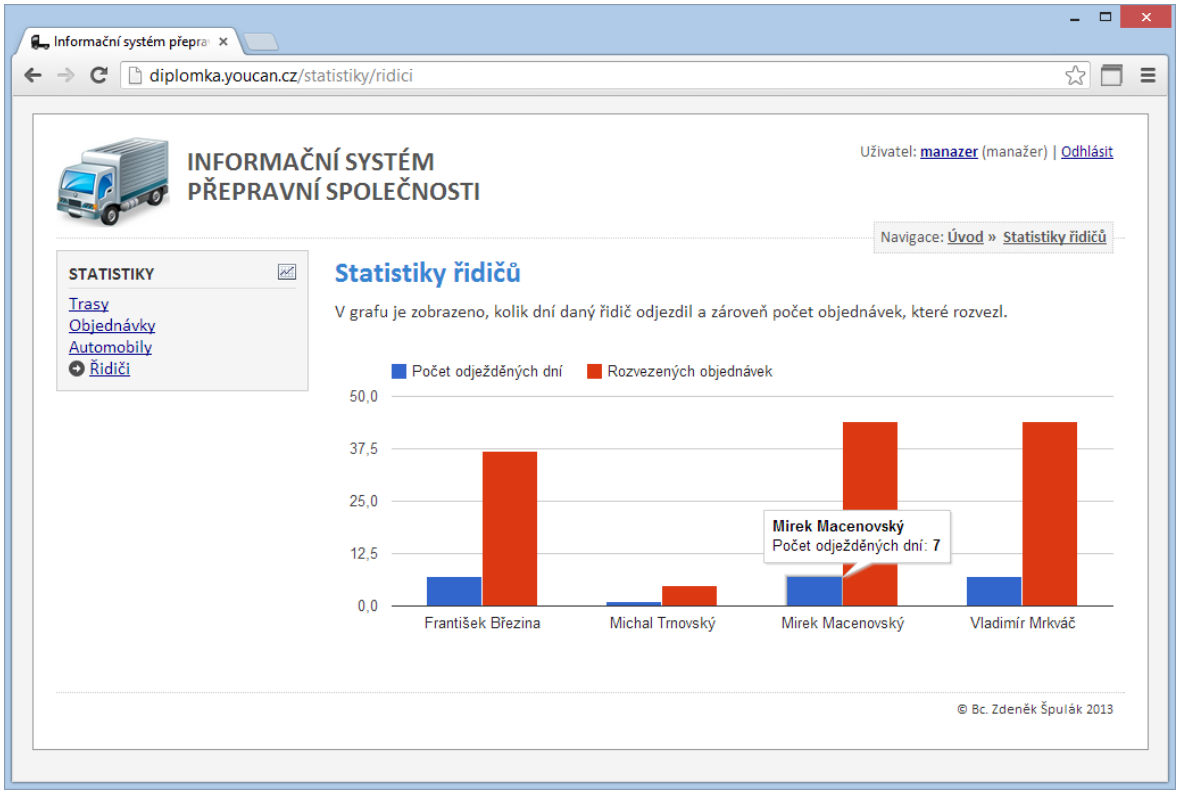
Floydův-Warshallův algoritmus. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-08-13]. Dostupné z: http://cs.wikipedia.org/wiki/Floyd%C5%AFv-Warshall%C5%AFv_algoritmus

Packing problem. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-08-13]. Dostupné z: http://en.wikipedia.org/wiki/Packing_problem

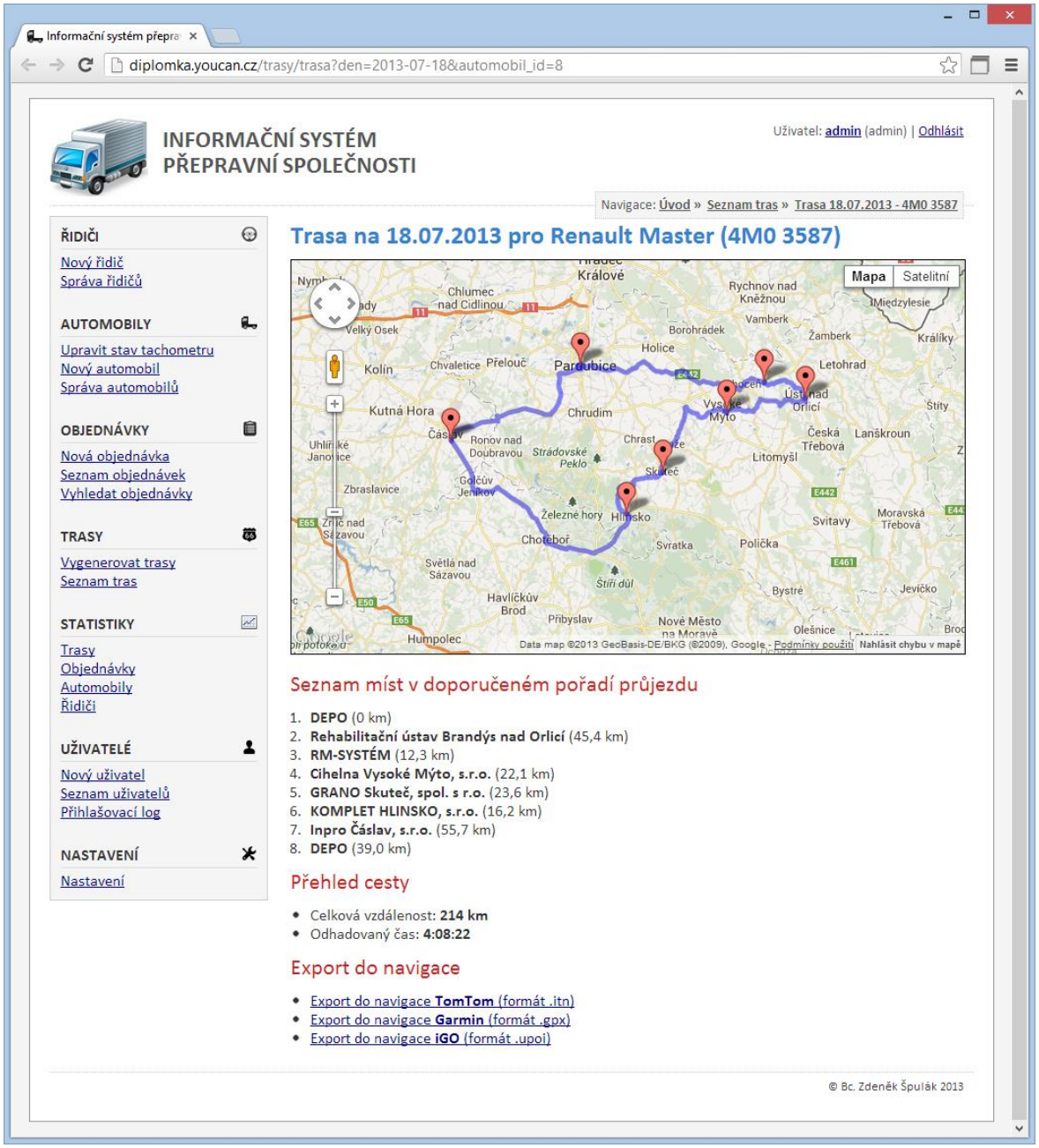
Příloha A – Screenshoty z vyvinuté aplikace



Obrázek 27 – Použití interaktivní mapy pro sledování cílových zemí objednávek



Obrázek 28 – Statistika vytíženosti řidičů



Obrázek 29 – Příklad vygenerované optimální trasy pro vozidlo spolu s možností exportu do GPS navigací a odhadovaným časem a vzdáleností trasy