

UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

DIPLOMOVÁ PRÁCE

2016

Bc. Marek Havelka

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Informační systém přepravní společnosti

Bc. Marek Havelka

Diplomová práce

2016

## ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Marek Havelka**  
Osobní číslo: **I13409**  
Studijní program: **N2646 Informační technologie**  
Studijní obor: **Informační technologie**  
Název tématu: **Informační systém přepravní společnosti**  
Zadávající katedra: **Katedra informačních technologií**

### Z á s a d y p r o v y p r a c o v á n í :

Cílem této práce je návrh systému a vytvoření funkční aplikace, 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. Přijímání 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 nezbytné provést rešerši systémů, které se zabývají touto problematikou. Rešerši je nutné doplnit o porovnání s nově navrhovaným systémem, který bude předmětem této práce. V práci bude provedena důkladná analýza procesů, které budou znázorněny v procesním BPMN diagramu. Diagram bude navržen s cílem dosažení maximální bezpečnosti informačního systému s využitím SW ARIS Express. Práce bude obsahovat analýzu navrhovaného řešení, popis použitých technologií, návrh databáze a aplikační řešení. V rámci návrhu databáze bude vytvořen ER diagram s využitím "Crow's Foot" notace entity-relationship. Pro vytvoření aplikace bude využit skriptovací jazyk PHP nebo JAVA a databáze MySQL nebo Oracle.

Rozsah grafických prací:

Rozsah pracovní zprávy: cca 60 stran

Forma zpracování diplomové práce: tištěná

Seznam odborné literatury:

ŘEPA, Václav. Analýza a návrh informačních systémů. 1. vyd. Praha :

Ekopress, 1999. 403 s. ISBN 80-86119-13-0.

LACKO, Luboslav. Oracle - Správa, programování a použití databázového systému. Brno: Computer Press a.s., 2007. 573 s. ISBN 978-80-251-1490-2.

GROFF, James R. a Paul N. WEINBERG. SQL kompletní průvodce. Brno: Computer Press a.s., 2005. 936 s. ISBN 80-251-0369-2.

BRYLA, Bob a Kevin LONEY. Mistrovství v Oracle Database 10g. Brno: Computer Press a.s., 2006. 700 s. ISBN 80-251-1277-2.

NARAMORE, Elizabeth, Jason GERNER, Scouarnec YANN LE and Timothy BORONCZYK. PHP

MySQL, Apache: Vytváříme webové aplikace. Brno: Computer Press a.s., 2009. 816 s. EAN:9788025127674.

Vedoucí diplomové práce:

Ing. Miloslav Macháček, Ph.D.

Katedra informačních technologií

Datum zadání diplomové práce:

31. října 2015

Termín odevzdání diplomové práce:

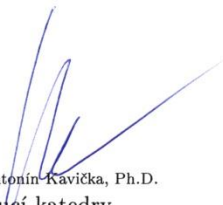
13. května 2016



prof. Ing. Simeon Karamazov, Dr.  
děkan



L.S.



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

V Pardubicích dne 15. listopadu 2015

## 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 3. 5. 2016

Bc. Marek Havelka

## **PODĚKOVÁNÍ**

Rád bych touto cestou vyjádřil poděkování Ing. Miloslavu Macháčkovi, Ph.D. za poskytnutí cenných rad, věcných připomínek, ochotu a vstřícný přístup při vedení mé diplomové práce. Dále bych chtěl poděkovat rodičům a kamarádům za trpělivost a podporu během studia.

## **ANOTACE**

Cílem této práce je návrh systému a vytvoření funkční aplikace, která bude obsahovat nástroje sloužící pro evidenci a logistické řízení přepravní společnosti.

V úvodní části je provedena rešerše systémů, které se zabývají touto problematikou a porovnání s nově navrhovaným systémem, který je předmětem této práce. Dále jsou popsány technologie a principy používané pro vývoj webových aplikací. Praktická část pojednává o vytvořeném informačním systému, funkcích a algoritmech použitých pro implementaci. Aplikace využívá skriptovací jazyk PHP a databázi MySQL.

## **KLÍČOVÁ SLOVA**

Informační systém, webová aplikace, přepravní společnost, problém obchodního cestujícího, Nette Framework, PHP, MySQL

## **TITLE**

Information System for the Transport Company

## **ANNOTATION**

The main object of this diploma thesis is to design a system and an application utility for registering and logistic management of a transfer company.

An opening part of this thesis includes an exploration of the facts about systems considering this problems and comparing these systems with the recently proposed system as the object of this thesis. Next part describes a technology and principles used for the web application design. The practical part of the thesis discusses recently designed information system, especially its functions and algorithms used for the implementation. An application is based on PHP script and MySQL database system.

## **KEYWORDS**

Information system, web application, transport company, the travelling salesman problem, Nette Framework, PHP, MySQL

# OBSAH

Úvod .....	15
1 Teoretická část.....	17
1.1 Elektronické informační systémy.....	17
1.2 Informační systém pro přepravní společnost .....	18
1.3 Dostupná řešení na trhu a jejich srovnání s navrhovanou aplikací .....	18
1.3.1 Dopravní a spediční informační systém SPEiS .....	18
1.3.2 Rinkai Routing .....	19
1.3.3 QI Informační systém – modul Doprava a spedice .....	20
1.3.4 Doprava 3K.....	20
1.3.5 Informační systém popisovaný v této práci.....	21
1.4 Technologie pro vývoj webových aplikací.....	22
1.4.1 World Wide Web a webová aplikace .....	22
1.4.2 HTTP (Hypertext Transfer Protocol) .....	22
1.4.3 HTML (HyperText Markup Language) .....	24
1.4.4 CSS (Cascading Style Sheets) .....	27
1.4.5 JavaScript.....	29
1.4.6 PHP .....	31
1.4.7 Architektura MVC .....	34
1.4.8 Architektura MVP.....	35
1.4.9 Nette Framework.....	36
1.4.10 MySQL databáze.....	37
1.5 Zabezpečení webových aplikací .....	37
1.5.1 Šifrované spojení mezi klientem a serverem .....	37
1.5.2 Autentizace a autorizace uživatele .....	38
1.5.3 Ošetření vstupů aplikace a SQL Injection .....	39
1.5.4 XSS (Cross Site Scripting) .....	40



1.6	Algoritmus pro výpočet trasy .....	41
1.6.1	Úloha obchodního cestujícího .....	41
1.6.2	Typy souřadných systémů .....	42
1.6.3	Navržený algoritmus pro řešení problému obchodního cestujícího.....	44
2	Praktická část.....	48
2.1	Návrh informačního systému.....	48
2.1.1	Informační systém pro přepravní společnost.....	48
2.1.2	Základní informace o aplikaci .....	48
2.1.3	Použité technologie .....	49
2.2	Architektura .....	49
2.2.1	Adresářová struktura projektu .....	49
2.2.2	Use case diagram – uživatelské role .....	50
2.2.3	UML Class diagram .....	51
2.3	Datový model.....	52
2.3.1	ER diagram .....	52
2.3.2	Tabulky v databázi – atributy, datové typy, vazby .....	52
2.4	Moduly aplikace.....	59
2.4.1	Evidence zaměstnanců .....	60
2.4.2	Evidence řidičů .....	61
2.4.3	Evidence vozidel.....	62
2.4.4	Evidence zásilek.....	63
2.4.5	Generování rozvozů .....	65
2.4.6	Statistiky .....	66
2.4.7	Nastavení .....	67
2.5	Algoritmus generování rozvozů – implementace .....	67
2.5.1	Příprava pro výpočet .....	68
2.5.2	Hlavní algoritmus výpočtu .....	69

2.6	Zdrojové kódy vybraných částí aplikace .....	70
2.6.1	Přihlášení do systému.....	70
2.6.2	Převod adresy na GPS souřadnice .....	71
2.6.3	Seznam řidičů pro rozvoz.....	72
2.7	Instalační příručka .....	74
2.7.1	Instalace informačního systému.....	74
2.7.2	První spuštění a konfigurace.....	75
2.8	Uživatelská příručka.....	75
2.8.1	Přihlášení do systému.....	75
2.8.2	Hlavní menu.....	76
2.8.3	Evidence řidičů .....	76
2.8.4	Evidence vozidel .....	77
2.8.5	Evidence zásilek a generování rozvozů.....	79
2.8.6	Modul statistik .....	80
	Závěr.....	82
3	Použitá literatura.....	84
4	Přílohy.....	87

## SEZNAM ILUSTRACÍ A TABULEK

Obrázek 1 – komunikace pomocí protokolu HTTP .....	22
Obrázek 2 – ukázka HTTP požadavku a odpovědi .....	24
Obrázek 3 – rozšířenost HTML verzí za posledních 5 let .....	25
Obrázek 4 – ukázka zdrojového kódu XHTML 1.0.....	26
Obrázek 5 – ukázka HTML5 formuláře a videa .....	27
Obrázek 6 – CSS styly zapsané v externím souboru.....	29
Obrázek 7 – funkce pro zobrazení mapy Google.....	31
Obrázek 8 – příklad PHP třídy.....	33
Obrázek 9 – vazby MVC .....	34
Obrázek 10 – diagram MVP, vzor Passive View.....	35
Obrázek 11 – graf závislosti trasy na velikosti úhlového sektoru.....	46
Obrázek 12 – znázornění cesty nalezené jednostranným algoritmem .....	47
Obrázek 13 – struktura projektu.....	49
Obrázek 14 – Use case diagram.....	51
Obrázek 15 – ER diagram.....	52
Obrázek 16 – UML Class diagram - evidence zaměstnanců .....	60
Obrázek 17 – UML Class diagram - evidence řidičů .....	61
Obrázek 18 – UML Class diagram - evidence vozidel .....	63
Obrázek 19 – UML Class diagram - evidence zásilek .....	64
Obrázek 20 – UML Class diagram - generování rozvozu .....	66
Obrázek 21 – kontrola přihlášení a uživatelské role .....	70
Obrázek 22 – funkce login.....	71
Obrázek 23 – funkce geocode.....	72
Obrázek 24 – výpis řidičů pro rozvoz .....	73
Obrázek 25 – konfigurační soubor databáze.....	75
Obrázek 26 – přihlašovací formulář.....	75
Obrázek 27 – hlavní menu aplikace pro roli operátor .....	76
Obrázek 28 - formulář pro úpravu řidiče .....	77
Obrázek 29 – evidence řidičů .....	77
Obrázek 30 – servis vozového parku .....	78
Obrázek 31 – formulář pro nastavení servisních údajů .....	78
Obrázek 32 – zásilky rozdělené podle stavu.....	79

Obrázek 33 – seznam rozvozů .....	80
Obrázek 34 – modul statistik – vozidlo .....	81

Tabulka 1 – HTTP stavové kódy .....	23
Tabulka 2 – porovnání účinnosti optimalizací.....	45
Tabulka 3 – login .....	52
Tabulka 4 – uživatelské role .....	53
Tabulka 5 – adresy .....	53
Tabulka 6 – osoby .....	54
Tabulka 7 – zaměstnanci .....	54
Tabulka 8 – řidiči .....	55
Tabulka 9 – rozvozy .....	55
Tabulka 10 – balíky .....	56
Tabulka 11 – adresy balíků .....	57
Tabulka 12 – vozidla .....	58
Tabulka 13 – výrobce vozidla .....	58
Tabulka 14 – vozidlo servis .....	59
Tabulka 15 – nastavení aplikace .....	59

## **SEZNAM ZKRATEK A ZNAČEK**

API	Application Programming Interface
BPMN	Business Process Model and Notation
CSS	Cascading Style Sheets
DES	Data Encryption Standard
ERP	Enterprise Resource Planning
GPS	Global Positioning System
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IS	Informační systém
JS	JavaScript
JSON	JavaScript Object Notation
MVC	Model-View-Controller
MVP	Model-View-Presenter
OS	Operační systém
PC	Personal Computer
PHP	PHP: Hypertext Preprocessor
SHA-256	Secure Hash Algorithm 256
SQL	Structured Query Language
SSL	Secure Sockets Layer
STK	Stanice technické kontroly
TCP/IP	Transmission Control Protocol/Internet Protocol
UML	Unified Modeling Language
URL	Uniform Resource Locator

UTF-8	Unicode Transformation Format
VIN	Vehicle Identification Number
WGS-84	World Geodetic System 1984
WWW	World Wide Web
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language
XSS	Cross Side Scripting

# ÚVOD

Diplomová práce se zabývá problematikou návrhu a implementace informačního systému pro evidenci a logistické řízení přepravní společnosti. S neustálým rozvojem výpočetní techniky a softwarových technologií je stále snadnější využít pro řízení chodu společnosti nějakou formu informačního systému. Na trhu je dostupných mnoho informačních systémů, které lze nasadit do firmy působící v oboru dopravy a spedice. Cílem této práce je analyzovat existující řešení a na základě zjištěných informací navrhnout vlastní informační systém implementovaný formou webové aplikace.

První část diplomové práce začíná pohledem na informační systémy obecně – jejich vývoj, postupný přechod k elektronickým informačním systémům, popis silných a slabých stránek. Dále jsou nastíněny podmínky, které by měl navrhovaný systém splňovat – plánované využití počítá s nasazením do fiktivní firmy, která se zabývá doručováním drobných zásilek na malé a střední vzdálenosti. Následuje analýza existujících řešení zabývajících se touto problematikou. Zde je popsáno zaměření vybraných systémů, přehled podporovaných funkcí a je hodnocena i architektura aplikace. V závěru rešerše nechybí ani srovnání s nově navrhovaným informačním systémem. Další kapitola pojednává o technologiích použitých pro vývoj navrženého systému a webových aplikací obecně. Jsou zde popsány programovací jazyky HTML, PHP, JavaScript a CSS. U každého z nich je krátce naznačena historie, vývoj do současné podoby, pro jaké úlohy je ten který jazyk vhodný a nechybí ani stručné vysvětlení syntaxe a názorná ukázka zdrojového kódu. Kromě programovacích jazyků jsou součástí této kapitoly i obecné technologie používané webovými aplikacemi – World Wide Web, protokol HTTP, návrhové vzory MVC a MVP. Závěrečná část této kapitoly pojednává o Nette Frameworku a databázi MySQL. Další kapitola teoretické části se zabývá problematikou zabezpečení webových aplikací. Jsou zde popsány některé typy útoků a zásady, jak se jim bránit. Závěrečná kapitola teoretické části je věnována problematice hledání optimální cesty pro rozvoz zásilek.

Druhá část diplomové práce popisuje vytvořenou aplikaci navrženou podle předpokladů stanovených v úvodní části práce. V kapitole návrh informačního systému je vysvětleno prostředí, pro které je aplikace určena a stručně popsána funkcionalita. Další kapitola se už zabývá strukturou aplikace z hlediska implementace – je zde vysvětlena adresářová struktura projektu, znázorněny vazby jednotlivých částí aplikace pomocí diagramu tříd a také Use case diagram popisující přístup k modulům aplikace v závislosti na uživatelské roli. Kapitola

datový model obsahuje detailní popis tabulek databáze – jejich atributů a vzájemných vazeb. Celkový pohled na strukturu databáze je realizován pomocí ER diagramu. V následující kapitole – moduly aplikace – je podrobně vysvětlena funkce modulů, ze kterých se celý informační systém skládá. Každý modul má vlastní podkapitolu, ve které je kromě popisu podporovaných funkcí obsažen i diagram tříd příslušné části zdrojového kódu. Vlastní kapitola je věnována i algoritmu generování rozvozů a výpočtu tras. Následuje ukázka zdrojového kódu vybraných funkcí doplněná popisem a vysvětlením, jak tento kód pracuje. Poslední dvě kapitoly slouží jako instalační a uživatelská příručka. Zde je popsán postup instalace a zprovoznění aplikace a základy práce s ní.



# 1 TEORETICKÁ ČÁST

## 1.1 Elektronické informační systémy

Před nástupem informační techniky se veškeré dokumenty spojené s provozem firmy musely uchovávat v psané a později tištěné formě. Práce s papírovými dokumenty byla ve srovnání s elektronickou agendou výrazně pomalejší a méně efektivní. Uskladnění velkého množství papírových dokladů vyžadovalo dostatek prostoru a pečlivé uspořádání dokumentů podle data či jiných parametrů, aby se v nich dalo efektivně vyhledávat. V případě ztráty nebo zničení dokumentu bylo obtížné tuto chybu napravit.

S rozvojem výpočetní techniky v 80. A 90. letech minulého století se ukázalo, že mnoho činností spojených s provozem firmy je možné převést do digitální podoby. V této době byl vývoj opravdu razantní, což vedlo k výraznému navyšování výkonu a zdokonalování dostupných aplikací. Zároveň rychle klesaly ceny a počítače se tak staly pro mnoho firem snadno dostupnými. Začaly tak vznikat elektronické informační systémy, které měly usnadnit a zefektivnit práci. Ze začátku se jednalo o aplikace pracující pouze na lokálním počítači, bez možnosti komunikovat s okolím. Postupem času se stalo standardem grafické rozhraní aplikací, ovládání myši, později přibyla i síťová komunikace mezi více PC, nasazení serverů a databází. Stále se však jednalo o aplikace desktopového typu. S rozvojem Internetu a webových technologií se objevil trend vytvářet webové aplikace. Topologie klient-server se využívala už dříve, odlišnost v případě webových aplikací spočívá v tom, že klientská část vždy běží v okně webového prohlížeče. Popularita webových aplikací trvá až do dnešních dní, proto velká část informačních systémů je založena právě na tomto modelu.

Moderní informační systém přináší výrazně lepší efektivitu práce, ochranu dat proti ztrátě nebo zničení a v neposlední řadě i snížení nákladů. Výhodou elektronického informačního systému je snadný a rychlý přístup k informacím díky pokročilým možnostem vyhledávání a efektivnímu procházení dat. Samozřejmostí je automatické provádění výpočtů nad zadanými hodnotami, vizualizace informací pomocí grafů, propojení s jinými informačními systémy nebo export dat do různých formátů. Výrazně vyšší je bezpečnost dat díky zálohování a decentralizaci serverů, na kterých systémy běží. Slabší stránkou elektronických IS a ukládání digitálních dat obecně je dlouhodobá trvanlivost dat a metody jejich ukládání.

## **1.2 Informační systém pro přepravní společnost**

Přepravní společnost zcela jistě potřebuje pro efektivní fungování informační systém. Podle konkrétního zaměření společnosti jsou potřebné odlišné funkce a možnosti, které jsou od informačního systému požadovány. Přepravní společnost zaměřující se například na mezistátní kontejnerovou dopravu pro efektivní funkci potřebuje výrazně odlišný systém než jiná společnost, jejíž pracovní náplň je rozvážet malé a střední zásilky do vzdálenosti jednotek až desítek kilometrů. Právě tento druhý scénář slouží jako vzor pro návrh vlastního informačního systému, který je předmětem této práce. Ačkoliv existují různé univerzální a velmi rozsáhlé informační systémy, které lze použít i na méně náročnou činnost, ovládání takových systémů bývá zbytečně komplikované a některé detailněji zaměřené funkce chybí. Informační systém pro firmu specializující se na rozvoz zásilek po blízkém a středně vzdáleném okolí by měl obsahovat tyto moduly:

- Evidence řidičů a zaměstnanců. Součástí modulu je seznam všech zaměstnanců společnosti – evidují se osobní údaje a pracovní statistiky. Samozřejmostí je přidávání nových zaměstnanců, úprava údajů a případně mazání ze systému.
- Evidence vozidel. Přehled o vozovém parku je klíčovou funkcí – ukládají se technické údaje, servisní informace a vytížení jednotlivých vozidel. I zde musí být možné průběžně aktualizovat informace o vozidlech, aby odpovídaly reálnému stavu.
- Evidence zásilek a plánování rozvozů. Tady jsou uloženy všechny přijaté zásilky a jejich parametry (rozměry, hmotnost, pojištění, adresa pro doručení). Pro zásilky je potřeba navrhnout trasy rozvozů, spolupracuje se s evidencí vozidel a řidičů. Možnost optimalizovat rozvozy z hlediska minimalizace nákladů.
- Modul statistik. Umožňuje vedení firmy sledovat vytížení vozidel a zaměstnanců. Volitelné parametry pro zobrazení, přehledné grafické zpracování.
- Zabezpečení aplikace proti neoprávněnému vstupu. Uživatelské role pro rozdělení přístupových práv.

## **1.3 Dostupná řešení na trhu a jejich srovnání s navrhovanou aplikací**

### **1.3.1 Dopravní a spediční informační systém SPEiS**

SPEiS je vyvíjen českou firmou TH SOFT už od poloviny 90. let. Aplikace je rozdělena na tři moduly – doprava, spedice a společné funkce. Modul dopravy obsahuje řadu funkcí. Samozřejmostí je evidence řidičů a vozidel. U vozidel se zaznamenávají i informace o servisních prohlídkách, STK, opravách nebo dopravních nehodách. Pro řidiče je možno

vyplnit informace o zdravotních prohlídkách, na blížící se termín expirace pak aplikace automaticky upozorní. Mezi další nabízené funkce patří kniha jízd, předlohy pro výpočet mezd a záloh na plánované cesty.

Modul spedice se zabývá evidencí zásilek. Systém zaznamenává všechny potřebné informace o dopravovaném zboží. Patří mezi ně seznam zboží v zásilce, adresa a termín pro naložení, adresa a termín pro vyložení, cena zásilky a řada dalších parametrů. Systém SPEiS je navržen spíše pro dopravu velkých zásilek z bodu A do bodu B. Není zde implementován žádný algoritmus optimalizace trasy pro více výdejních míst.

Z hlediska koncepce aplikace se jedná o desktopovou aplikaci win32 založenou na architektuře klient/server. SPEiS umožňuje provoz na lokální síti s využitím databázového serveru, případně i synchronizaci více vzdálených poboček prostřednictvím sítě Internet. Synchronizace může probíhat v reálném čase (online) nebo jen jednou za čas s využitím lokální mezipaměti.

Vlastnosti:

- Rozsáhlá evidence řidičů a vozidel.
- Umožňuje sledovat a zaznamenávat informace o opravách a servisních prohlídkách vozidel.
- Podrobné informace o zásilkách.
- Sdílení dat mezi pobočkami přes Internet.
- Desktop aplikace, vyžaduje OS Windows. [1]

### **1.3.2 Rinkai Routing**

Aplikace od firmy Rinkai není tak detailní jako předchozí popisovaná, ale obsahuje všechny potřebné funkce pro naplánování rozvozu zásilek. Důraz je kladen na přehlednost aplikace, jednoduché zadávání zásilek do systému a výpočet tras rozvozu. Objednávky lze zadávat ručně nebo automaticky načítat z informačních systémů zákazníků pomocí ERP. Z přijatých zakázek se vygenerují trasy rozvozu a časový plán. Tyto informace se předají řidičům a zákazníkům. Rinkai Routing umožňuje i sledování vozidel pomocí GPS a následné vyhodnocení, jestli rozvoz probíhal podle plánu.

Vlastnosti:

- Přehledné a snadno ovladatelné grafické rozhraní aplikace.
- Ruční i automatické zadávání objednávek.
- Automatický výpočet optimálních tras pro rozvoz.

- Upozornění zákazníků na doručovanou zásilku.
- Sledování vozidel a analýza rozvozových tras s využitím GPS. [2]

### **1.3.3 QI Informační systém – modul Doprava a spedice**

Společnost DC CONCEPT a.s. se zabývá vývojem informačních systémů pro střední a velké firmy. K dispozici je velké množství modulů navržených pro různá odvětví průmyslu. Modul doprava a spedice je určen pro dopravní a speditérské firmy. Stejně jako všechny předchozí aplikace, i tato uchovává informace o vozidlech, řidičích a ujetých cestách. Součástí správy vozidel jsou i informace o servisních kontrolách.

Po přijetí objednávek systém automaticky rozdělí zásilky do vozidel a navrhne optimální trasy ve spolupráci s mapovým systémem. Kromě trasy jízdy se automaticky počítá i silniční daň a generují se doklady pro pojištění vozidel. Po dokončení cesty systém umožňuje kontrolovat efektivitu jízdy a na základě toho navrhnout penalizaci nebo odměnu pro řidiče. Efektivita se určí na základě porovnání skutečné trasy ujeté vozidlem a spotřebovaného paliva s hodnotami vypočtenými při návrhu.

Vlastnosti:

- Velmi rozsáhlý systém.
- Evidence řidičů, vozidel a objednávek s množstvím detailů.
- Automatický výpočet optimálních tras pro rozvoz.
- Sledování vozidel – porovnání skutečné trasy s navrženou.
- Výpočet plánovaného zisku a nákladů
- Desktop aplikace, vyžaduje OS Windows. [3]

### **1.3.4 Doprava 3K**

Informační systém Doprava 3K vyvíjený firmou KSH-Data je určen pro střední a velké firmy v oboru dopravy a spedice. Stejně jako všechny předchozí popisované aplikace, i Doprava 3K obsahuje množství funkcí potřebných pro chod dopravní společnosti. Systém se skládá z mnoha modulů, přičemž každý z nich obsahuje odlišnou sadu funkcí.

Modul dopravy umožňuje velmi podrobně vést agendu vozového parku firmy. Součástí je evidence vozidel – technické údaje, servisní kontroly, záznamy o provozu vozidla, opravy vozidel. Další částí modulu dopravy je evidence řidičů – kromě zaznamenání osobních údajů

aplikace obsahuje i funkce pro evidenci pracovní doby a výpočet mzdy. Na základě dat získaných z vykonaných jízd systém generuje rozsáhlé souhrnné informace.

Modul spedice uchovává všechny potřebné informace o zakázkách. Nejdůležitějšími částmi jsou databáze zákazníků, jejich objednávek na přepravu a výpočet trasy rozvozu. Systém využívá mapové podklady pro optimální plánování tras. Nechybí ani podrobné informace o stavu zakázek. V případě potřeby lze dělat ruční úpravy v přiřazení zásilek do zvolených vozidel.

Vlastnosti:

- Velmi rozsáhlý systém rozdělený do několika modulů.
- Evidence řidičů, vozidel a objednávek s množstvím detailů.
- Automatický výpočet optimálních tras pro rozvoz.
- Ruční nebo automatické přiřazení zásilek do rozvozu.
- Výpočet nákladů a výnosů pro vozidla.
- Aplikace typu klient / server – část klient je určena pro OS Windows. [4]

### **1.3.5 Informační systém popisovaný v této práci**

Cílem této práce je vyvinout vlastní informační systém pro zajištění provozu přepravní společnosti. Vzhledem k omezenému času a kapacitám pro vývoj není možné navrhnout a vytvořit systém, který by byl svým rozsahem a funkcemi plně srovnatelný s komerčními aplikacemi popsány v předchozích bodech. Navržená aplikace je určena pro firmy specializující se na rozvoz zásilek ve vzdálenosti jednotek až desítek kilometrů od centrály.

Aplikace je rozdělena na čtyři moduly – evidence vozidel, řidičů, práce se zásilkami a modul statistik. Evidence vozidel uchovává všechny potřebné informace o všech vozidlech používaných ve firmě. Součástí modulu jsou i informace o STK a servisních prohlídkách – v případě blížící se expirace je uživatel systému automaticky upozorněn. Nejdůležitější částí aplikace je modul zásilek – zde se vkládají do systému přijaté balíky, které se následně přiřadí do rozvozu. Plánování rozvozů a výpočet optimální trasy probíhá automaticky. Poslední částí je modul statistik, kde je možné prohlížet vytíženost jednotlivých řidičů a vozidel za zvolené období ve formě grafů a tabulek.

Vlastnosti:

- Systém navržený pro lokální rozvoz zásilek.
- Evidence řidičů, vozidel a balíků se všemi potřebnými informacemi.

- Upozornění na STK a servisní kontroly vozidel.
- Automatické generování rozvozů s ohledem na rovnoměrné vytížení řidičů a vozidel.
- Automatický výpočet optimálních tras pro rozvoz.
- Sledování vytížení vozidel a řidičů.
- Webová aplikace

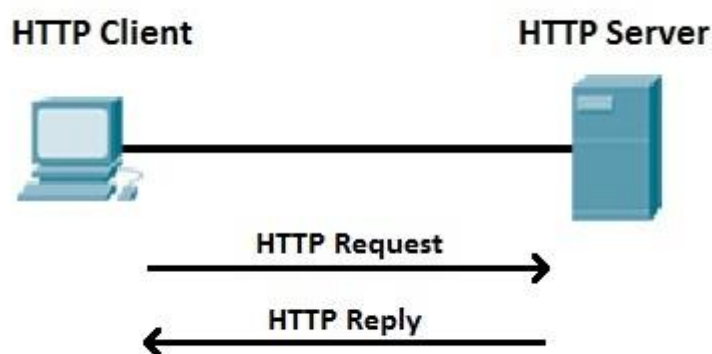
## 1.4 Technologie pro vývoj webových aplikací

### 1.4.1 World Wide Web a webová aplikace

World Wide Web, krátce nazývaný WWW je základem pro všechny webové aplikace. Princip WWW spočívá v mnoha dokumentech navzájem propojených mezi sebou hypertextovými odkazy. Web je založen na architektuře klient – server. Server zpracovává požadavky a v případě kladného vyřízení pošle klientovi požadovaný dokument, který se zobrazí ve webovém prohlížeči. Pro veřejnost byl projekt uvolněn v roce 1993 a od té doby rozšířenost webu stále stoupá. [35]

### 1.4.2 HTTP (Hypertext Transfer Protocol)

Protokol HTTP je neoddělitelnou součástí webových aplikací a vlastně celého Internetu. HTTP protokol je založen na principu request – response (požadavek - odpověď). Klient na server pošle požadavek, server vrátí požadovaný dokument. Klientem je obvykle internetový prohlížeč (například Mozilla Firefox, Google Chrome, Internet Explorer), na druhé straně spojení požadavky vyřizuje HTTP server (mezi nejpoužívanější patří Apache nebo Microsoft IIS Server).



Obrázek 1 – komunikace pomocí protokolu HTTP, zdroj: [5]

Existují tři verze protokolu – verze 0.9, verze 1.0 a verze 1.1. První verze byla velmi jednoduchá a podporovala pouze metodu GET. Verze 1.0 už dokázala pracovat s HTTP hlavičkami a k funkci GET přidala ještě HEAD a POST. Nejdůležitější novinkou v poslední verzi je použití perzistentního spojení pro přenos více dokumentů ze serveru v rámci jednoho spojení. To v předchozích verzích protokolu možné nebylo – pro každý objekt se muselo navázat nové spojení a v důsledku toho se přenos složitějších stránek složených z mnoha objektů zpomalil.

Důležitou součástí HTTP protokolu jsou stavové kódy. Jejich účel je informovat o událostech nebo chybách vzniklých při zpracování požadavku. Kódy jsou označeny třímístným číslem a dělí se do pěti skupin. Podle první číslice kódu se pozná příslušná skupina: 1 = informace, 2 = úspěch, 3 = přesměrování, 4 = chyba na straně klienta, 5 = chyba na straně serveru. Následující tabulka popisuje nejčastěji se vyskytující kódy.

**Tabulka 1 – HTTP stavové kódy, zdroj: [6]**

**Nejběžnější HTTP stavové kódy**

<b>Stavový kód</b>	<b>Název</b>	<b>Popis</b>
200	OK	Požadavek úspěšně vyřízen
301	Moved permanently	Požadovaná stránka byla přesunuta na jinou adresu
400	Bad request	Nesprávně zadaný požadavek
403	Forbidden	Server odmítne odpovědět
404	Not found	Objekt specifikovaný v dotazu nenalezen
500	Internal server error	Vnitřní chyba serveru
503	Service unavailable	Server nedokáže zpracovat požadavek – obvykle v důsledku přetížení

Na obrázku 2 je ukázka syntaxe dotazu GET a získané odpovědi. První řádek musí ve stanoveném pořadí obsahovat metodu dotazu, adresu požadovaného objektu a verzi HTTP protokolu. Další řádky tvoří hlavičku dotazu, kde se přenáší dodatečné informace. Položka User-Agent předává informace o klientovi, který poslal dotaz. Host specifikuje adresu a port serveru, na který se dotaz posílá. Pokud port není uveden, použije se implicitně port 80. Accept-Language a Accept-Encoding stanovují jazyk a kódování souborů, které je klient schopný přijmout. Nastavení Connection na hodnotu Keep-Alive značí, že po přenesení požadovaného souboru se spojení neukončí a bude připraveno pro další přenosy.

Prostřední blok obrázku 2 zobrazuje odpověď zaslanou serverem. Odpověď se dělí na dvě části – první z nich je hlavička, druhá požadovaný dokument (soubor hello.html). Hlavička obsahuje stavový kód, datum a čas odeslání odpovědi, název aplikace, která zpracovala požadavek, datum a čas poslední modifikace požadovaného dokumentu, jeho délku a typ. [6], [7], [8]

```
GET /hello.htm HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.tutorialspoint.com
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed

<html>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

Obrázek 2 – ukázka HTTP požadavku a odpovědi, zdroj: [7]

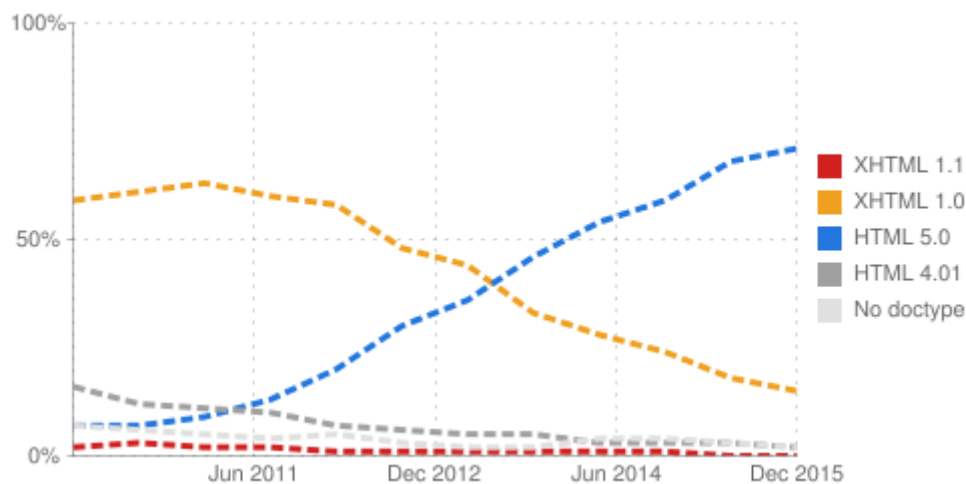
### 1.4.3 HTML (HyperText Markup Language)

HTML patří mezi klíčové technologie, bez které by webové aplikace v současné podobě nemohly existovat. HTML je značkovací jazyk určený pro tvorbu webových stránek. Obsahem je obvykle text doplněný obrázky, tabulkami, případně dalšími objekty. První verze vznikla v roce 1991 společně s konceptem webu. Ze začátku bylo podporováno 18 elementů (tagů) pro formátování a sestavení dokumentu. Některé z nich se používají dodnes. Během následujících let vývoj pokračoval a postupně byly vydány tyto hlavní verze:

- HTML 2.0 (rok 1995) – přidány grafické prvky a formuláře.
- HTML 3.2 (rok 1997) – přidány tabulky, zarovnání textu a další prvky pro grafickou úpravu.
- HTML 4.01 (rok 1999) – pokročilejší formátování tabulek, oddělení formátování od obsahu dokumentu, vznik CSS.



- HTML5 (rok 2014) – celá řada změn, vylepšené formuláře, nové sémantické elementy, přímá podpora pro multimédia.



**Obrázek 3 – rozšířenost HTML verzí za posledních 5 let, zdroj: [9]**

Dlouhé období mezi HTML 4.01 a 5 vyplnil standard XHTML, který přinesl do HTML některé aspekty formátu XML. Vznikly dvě verze – 1.0 a 1.1. V plánu byla i verze 2.0, která měla přinést velké změny a odklon od koncepce XHTML 1.1, ale vývoj byl zastaven ve prospěch HTML5.

- XHTML 1.0 (rok 2000) – stejné funkce jako HTML 4.01, nutno dodržovat normu XML.
- XHTML 1.1 (rok 2001) – odstranění některých elementů, které byly už v předchozí verzi označeny jako zastaralé.

Princip XHTML spočívá v důsledném dodržování pravidel jazyka XML, na kterém je XHTML založeno. V případě syntaktické chyby se zpracování stránky může zastavit (záleží na konkrétní implementaci v použitém prohlížeči), proto je nutné dbát na správnou syntaxi. Mezi základní pravidla patří:

- Správná posloupnost elementů – nesmí se křížit.
- Uzavírání párových i nepárových tagů
- Psát elementy malými písmeny – XHTML rozlišuje velikost písmen.
- Pokud element obsahuje atributy, ty musí mít vždy přiřazenou hodnotu.

Následující obrázek obsahuje krátký kus zdrojového kódu v jazyce XHTML. Jsou dodržena všechna výše popsaná pravidla – tento kód je validní.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>XHTML stránka</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  </head>
  <body>
    <div class="obsah">Obsah stránky</div>
    <br/>
  </body>
</html>
```

Obrázek 4 – ukázka zdrojového kódu XHTML 1.0

HTML5 přináší celou řadu novinek. Proti starším verzím jazyka přibyly sémantické elementy, širší možnosti pro tvorbu formulářů, přímá podpora zvuku nebo videa a nechybí ani implementace různých API pro pokročilé funkce webových aplikací. Sémantické elementy umožňují rozdělit stránku do logických bloků a zlepšit tím funkčnost vyhledávačů na dané stránce. Mezi nejpoužívanější elementy patří <header>, <footer>, <nav>, <section>, <article>, <aside>, <figure>. S HTML5 se snáze vytváří pokročilé formuláře, které mají přijímat jen vybraný typ dat a zároveň dokáží kontrolovat, jestli zadané hodnoty odpovídají tomuto typu a zadanému rozsahu dat. Nové typy vstupů jsou například: „color“, „date“, „datetime“, „email“, „number“, „tel“. U některých vstupů je možné nastavit omezující podmínky, které se při odeslání formuláře kontrolují. Ačkoliv HTML5 je už více než rok oficiálním standardem, některé prohlížeče přesto nepodporují pokročilé volby nastavení formulářů. Obrázek 5 ukazuje zdrojový kód HTML5 – formulář s využitím vstupního typu „date“ a omezením rozsahu vstupních hodnot. Ve spodní části je použit element <video>.

[10], [11], [12], [13], [14]

```

<!DOCTYPE html>
<html>
  <head>
    <title>HTML5 formulář a video</title>
  </head>
  <body>
    <form action="action_page.php">

      Enter a date before 1980-01-01:
      <input type="date" name="bday" max="1979-12-31"><br>

      Enter a date after 2000-01-01:
      <input type="date" name="bday" min="2000-01-02"><br>

      Quantity (between 1 and 5):
      <input type="number" name="quantity" min="1" max="5"><br>

      <input type="submit">
    </form>

    <video width="320" height="240" controls>
      <source src="movie.mp4" type="video/mp4">
      <source src="movie.ogg" type="video/ogg">
      Your browser does not support the video tag.
    </video>
  </body>
</html>

```

Obrázek 5 – ukázka HTML5 formuláře a videa, zdroj: [10], [11]

#### 1.4.4 CSS (Cascading Style Sheets)

Rané verze HTML upravovaly vzhled stránky výhradně pomocí k tomu určených elementů. Nástup HTML 4.01 a zejména XHTML přinesl velkou změnu – formátování stránky pomocí nově vytvořeného jazyka CSS. CSS se ve své první verzi objevilo v roce 1996, ale výrazněji se rozšířilo až později kvůli slabé podpoře ze strany webových prohlížečů. V tomto přechodném období se často kombinovalo stylování části stránky pomocí CSS a zbytek klasickými HTML elementy. S nástupem XHTML a HTML5 se od starých způsobů formátování upustilo úplně a dnes se používá výhradně CSS.

Princip funkce CSS spočívá v nastavení vzhledu každého prvku stránky stanoveným stylem. Protože objekty stránky jsou obvykle uspořádané hierarchicky, i CSS styly musí toto respektovat. U složitějších struktur může nastat problém s dědičností atributů nastavených rodičovským prvkům.

CSS styl lze zapsat třemi způsoby:

- Externí – styly jsou zapsány v jednom nebo více souborech typu \*.css. Tyto styly se připojí k HTML dokumentu pomocí elementu `<link>`, který se zapíše do sekce `<head>`.
- Interní – styly se zapíší do sekce `<style></style>` v daném HTML souboru.
- Inline – styl se napíše přímo ke konkrétnímu elementu.

Všechny tři způsoby docílí stejného vzhledu dokumentu. Z hlediska priority jsou všechny druhy zápisu rovnocenné. Pokud je pro jeden prvek definováno více stylů navzájem se přepisujících, aplikuje se ten, který je v kódu později. Například inline styl má vždy vyšší váhu než externí nebo interní styl, protože je zapsán bezprostředně u stylovaného elementu. Pro přehlednost zdrojového kódu je dobré používat variantu, která je nejvhodnější. Inline styl je vhodné používat jen ve výjimečných případech, kdy je nutné odlišit jeden konkrétní prvek dokumentu (například při vývoji a ladění aplikace), případně pro velmi jednoduché stránky, kde je jen málo objektů. Pro interní a externí zápis stylů platí, že definice vzhledu elementů je vždy oddělena od samotné informace, která je v nich obsažena. Interní zápis stylů se hodí zejména pro kratší dokumenty, externí najde uplatnění hlavně v rozsáhlejších projektech skládajících se z mnoha stránek, které používají stejné styly. Stačí ke každé z nich připojit externí CSS soubor obsahující definice stylů pro celý projekt nebo blok aplikace. Výhodou externího zápisu je snadná úprava vzhledu aplikace – změna stylu na jednom místě může ovlivnit mnoho elementů, kterým je přiřazen.

Syntaxe CSS je do jisté míry podobná moderním programovacím jazykům. Pro elementy vyskytující se na stránce lze nastavovat mnoho různých parametrů. Jaké konkrétní atributy jde nastavit, záleží na typu zvoleného elementu. Nastavení požadované vlastnosti je možné přiřadit buď jednotlivě, nebo hromadně pro více elementů jedním blokem CSS kódu. Velmi důležitý element pro nastavení společných stylů bloku HTML kódu je `<div>`. Tento tag sám o sobě nemá žádný vliv na formátování dokumentu, uplatnění najde jen ve spojení s CSS. Možnosti, jak v CSS stylu adresovat `<div>`, jsou celkem tři. Nejpoužívanější možností je využití selektoru class nebo id. Oboje využívá nastavení atributu „class“, respektive „id“ pro `<div>` element, ale totéž lze uplatnit i na jiné elementy. Class se stejným označením může v HTML být použito vícekrát, id pouze jednou. Ve stylpisu se na vybraný blok HTML kódu odkazuje přes název třídy nebo id. Další varianta je adresovat selektory přímo podle typu a jejich hierarchické posloupnosti. Všechny metody se v praxi používají, volba optimálního

způsobu adresace elementů záleží podle konkrétní situace. V případě využití inline stylu se kód stylu napíše přímo do těla elementu, jako je uvedeno v následujícím příkladu.

```
<p style="color: blue;">tento odstavec má nastavenou modrou barvu pomocí inline stylu</p>
```

Na obrázku 6 je ukázka syntaxe CSS stylpisu umístěného v externím souboru. [15], [16]

```
h1 { /*platí pro všechny nadpisy H1*/
    color: red;
}

h2, h3, p { /*platí pro vypsané elementy, kdekoliv v dokumentu*/

    border-style: solid;
    border-color: yellow;
    border-width: 1px;
}

.vrchni_blok { /*platí pro všechny elementy uvnitř třídy vrchni_blok*/
    width: 100%;
    text-align: center;
    margin: 25px;
}

.vrchni_blok > h1 { /*platí pro nadpisy H1 uvnitř třídy vrchni_blok*/
    color: green;
}

table > a { /*platí pro odkazy uvnitř tabulek*/
    font-size: 10px;
}
```

Obrázek 6 – CSS styly zapsané v externím souboru

### 1.4.5 JavaScript

Javascript je dalším programovacím jazykem, který se používá pro vývoj webových aplikací. Spolu s HTML a CSS tvoří základ pro tvorbu webových stránek. Obvykle se JavaScript využívá pro přidání interaktivních grafických prvků do webových aplikací, často se používá i pro validaci formulářů. Výhodou je asynchronní zpracování Javascriptových funkcí bez nutnosti znovu načíst stránku. Díky tomu je možné dynamicky zobrazovat různá vyskakovací okna, obrázky, nebo třeba přidávat nová pole do formulářů. JS kód se zapisuje přímo do

HTML – v elementu <script>. Druhá možnost je založit externí soubor (\*.js), kde je JS zdrojový kód uložen a k HTML se připojuje následujícím příkazem:

```
<script src="scripts/datetime/jquery-1.8.2.js"></script>
```

Stejně jako HTML, i JavaScript zdrojový kód se spouští přímo ve webovém prohlížeči (nikoliv na serveru). Bezchybná funkčnost i rychlost provádění záleží na kvalitě implementace JS v prohlížeči. Při návrhu aplikace je vhodné zvážit i možnost, že v závislosti na typu webového prohlížeče může být JavaScript nepřístupný nebo deaktivovaný.

Syntaxe se podobá vyšším programovacím jazykům jako c++ nebo java. JavaScript je objektově orientovaný skriptovací jazyk. Proměnné jsou netypové s nepovinnou deklarací, je tedy možné do proměnné vložit libovolná data. K prvkům pole je možné přistupovat klasicky přes indexy typu integer, ale dostupné je i asociativní indexování. Při novém načtení stránky se obsah standardních proměnných neuchová, pokud je taková funkce požadována, je nutné je označit jako statické. Mezi nejdůležitější funkce JavaScriptu patří schopnost přímo přistupovat k HTML objektům a pracovat s jejich obsahem. Obrázek 7 ukazuje funkci initMap() napsanou v jazyce JavaScript, která do připraveného elementu <div id="map"> vloží interaktivní mapu načtenou ze serveru Google. Tato funkce obsahuje práci s proměnnými, poli, cykly a připojuje se na element HTML označený identifikátorem „map“. [17], [18]

```

<script type="text/javascript">
    function initMap() {

        var locations = [
            ['Bod 1', 50.229248, 15.817081, 1],
            ['Bod 2', 50.224785, 15.706016, 2],
            ['Bod 3', 49.911214, 15.903426, 3],
            ['Bod 4', 50.266297, 16.385451, 4],
        ];

        var map = new google.maps.Map(document.getElementById('map'), {
            zoom: 9,
            center: new google.maps.LatLng(50.279462, 16.002303),
            mapTypeId: google.maps.MapTypeId.ROADMAP
        });

        var infowindow = new google.maps.InfoWindow();

        var marker, i;

        for (i = 0; i < locations.length; i++) {
            marker = new google.maps.Marker({
                position: new google.maps.LatLng(locations[i][1], locations[i][2]),
                map: map
            });

            google.maps.event.addListener(marker, 'click', (function (marker, i) {
                return function () {
                    infowindow.setContent(locations[i][0]);
                    infowindow.open(map, marker);
                }
            })(marker, i));
        }
    }
}
</script>

```

Obrázek 7 – funkce pro zobrazení mapy Google

### 1.4.6 PHP

Další skriptovací jazyk používaný pro tvorbu webových aplikací je PHP. I tento jazyk se používá ve spojení s HTML, ale na rozdíl od JavaScriptu se veškerý kód spouští na serveru a uživatel vidí ve svém prohlížeči pouze vygenerované HTML. Z pozice uživatele webové aplikace je tedy nemožné nahlédnout do zdrojového kódu PHP nebo ho měnit. První verze PHP vznikla v roce 1995 a od té doby probíhá vývoj až do současnosti. Aktuálně nejnovější verze je 7.0.4 vydaná 3. března 2016. Jazyk PHP je plnohodnotným objektově orientovaným jazykem – nechybí podpora práce se soubory, připojení k databázi, práce s řetězci, matematické funkce, jednoduché použití polí a mnoho dalších nástrojů.

Z hlediska syntaxe se PHP stejně jako celá řada ostatních programovacích jazyků podobá C++. Odlišná je ovšem práce s proměnnými. Stejně jako JavaScript, ani PHP u proměnných nerozlišuje datové typy. U lokálních proměnných rovněž není vyžadována deklarace – stačí dodržet několik syntaktických pravidel při pojmenování proměnné a přiřadit hodnotu. Tato pravidla stanovují, že proměnná musí vždy začínat symbolem \$ (dolar), nesmí obsahovat speciální znaky a za symbolem dolaru nesmí název začínat číslicí. Stejný princip platí i u polí, kde lze míchat mnoho datových typů. Indexování je obvykle realizováno standardně typem integer, často se využívá i asociativní indexování – řetězcem. Pole samozřejmě mohou být vícerozměrná a oba typy indexování jsou použitelné zároveň.

Složitější je práce s globálními proměnnými při použití objektového přístupu nebo sekvenčního s využitím vlastních funkcí. Globální proměnné v PHP třídě se obvykle deklarují na začátku, je nutné proměnné označit jedním z následujících klíčových slov:

- Public – objekt je dostupný v rámci celé aplikace.
- Protected – objekt je dostupný v rámci třídy ve které se nachází a ostatních tříd, které ji zdědily.
- Private – objekt je dostupný výhradně v rámci třídy, kde je deklarovaný.

Ve funkcích třídy se pak k proměnným přistupuje přes klíčové slovo následujícím způsobem:

```
$this->nazev_promenne
```

Globální proměnné v sekvenčním PHP kódu se řeší jiným způsobem. Proměnná se deklaruje na začátku bloku, v tomto případě bez označení public / protected / private. V každé funkci, kde má být globální proměnná přístupná, se tato deklaruje znovu s použitím klíčového slova global. Globální proměnné jsou vhodné pro výměnu dat mezi funkcemi v rámci třídy nebo bloku kódu. Nehodí se však pro komunikaci jednotlivých částí aplikace mezi sebou a data se přemažou i při obnovení stránky v prohlížeči. V tomto případě se používají superglobální proměnné. Těchto proměnných je celkem 9 a jsou napevno definovány PHP serverem, není možné založit vlastní superglobální proměnnou. Mezi předdefinované proměnné patří: \$GLOBALS, \$\_SERVER, \$\_REQUEST, \$\_POST, \$\_GET, \$\_FILES, \$\_ENV, \$\_COOKIE, \$\_SESSION. Pro uživatelské použití se nejvíce hodí první uvedená proměnná \$GLOBALS – stačí založit vlastní pole (například \$GLOBALS[„data“]=1.25) a uložená hodnota bude snadno přístupná ve všech částech aplikace.



Podle způsobu programování – sekvenční nebo objektově orientované – se v PHP zdrojový kód píše dvěma různými způsoby. Sekvenční programování je vhodné zejména pro koncové části aplikace – tedy ty, které do připraveného HTML kódu vypisují dynamicky generovaná data. Soubor s tímto kódem má vždy příponu \*.php a kromě PHP obsahuje zdrojový kód HTML, někdy spojený i s CSS nebo JavaScriptem. Do libovolného místa dokumentu se vloží PHP kód mezi speciální elementy:

- <?php – začátek PHP skriptu.
- ?> – konec skriptu.

Těchto skriptů lze vložit libovolně mnoho, počet není omezen. Proměnné definované v jednom skriptu jsou dostupné i pro všechny ostatní v rámci souboru.

Objektový přístup se obvykle používá pro ostatní části aplikace, které se přímo nepodílí na zobrazení koncových dat. Typicky se jedná o moduly starající se o práci s databází, ukládání souborů na server, nebo provádění složitějších výpočtů. V tomto případě je vhodné vytvořit PHP třídu. Třída neobsahuje žádné HTML, pracuje se zde pouze s PHP kódem.

```
class newPHPClass {  
  
    private $text = "";  
  
    public function __construct($textCon) {  
        $this->text = $textCon;  
        $this->transformace();  
    }  
  
    private function transformace() {  
        $this->text = strtoupper($this->text);  
    }  
  
    public function vypisText() {  
        return $this->text;  
    }  
  
}
```

Obrázek 8 – příklad PHP třídy

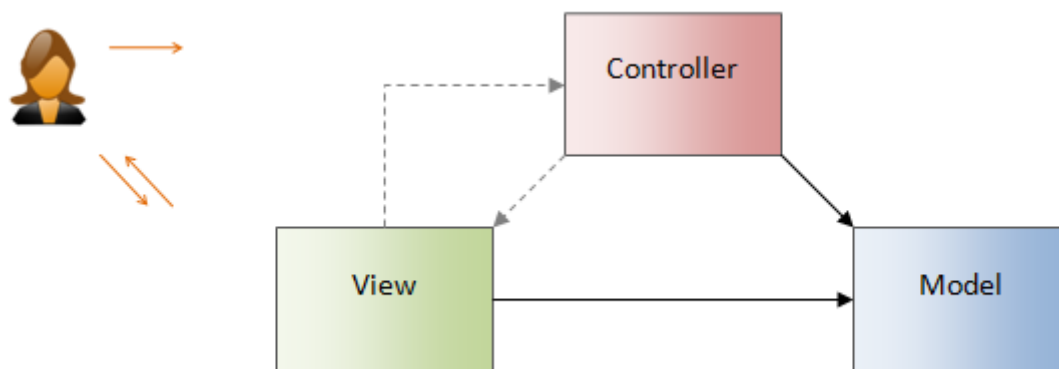
Jednoduchá PHP třída je zobrazena na obrázku 8. Je vidět, že PHP třída vypadá velmi podobně jako v ostatních objektově orientovaných jazycích. Ukázka obsahuje parametrický konstrukt, soukromou lokální proměnnou, soukromou funkci a veřejnou funkci, která vrací obsah proměnné text. Ke globálním proměnným třídy i funkcím je možné přistoupit jen

s použitím klíčového slova `$this`. Funkce jsou stejně jako proměnné netypové, nerozlišuje se ani, zda funkce vrací data nebo ne. [19], [20], [21]

### 1.4.7 Architektura MVC

Při vývoji webových aplikací se zvětšujícím se rozsahem výrazně narůstá množství zdrojového kódu a současně klesá jeho přehlednost. Úpravy nebo rozšiřování aplikace vytvořené živelně bez předem stanovené koncepce se tak může stát velmi obtížnou nebo téměř nemožnou prací. Jeden ze způsobů, jak lépe organizovat strukturu aplikace je architektura MVC.

Hlavní myšlenkou MVC je oddělení vnitřní aplikační logiky od výstupu (obvykle grafického rozhraní). Ideální stav je pokud možno nekombinovat dohromady HTML a PHP. Architektura MVC zavádí tři druhy objektů – Model, View, Controller.



Obrázek 9 – vazby MVC, zdroj: [22]

Model se vůbec nestará o interakci s uživatelem nebo zobrazení informací. Jeho náplní je uchovávat data, zapisovat nebo aktualizovat je (obvykle tyto úkoly znamenají práci s databází). Dále model obsahuje aplikační logiku – tedy různé výpočty a operace nad daty. Aplikace se většinou dělí na oddělené funkční bloky, kde každý z nich spravuje určitou část funkcí. Obvykle každý takovýto blok je zastoupen samostatným modelem.

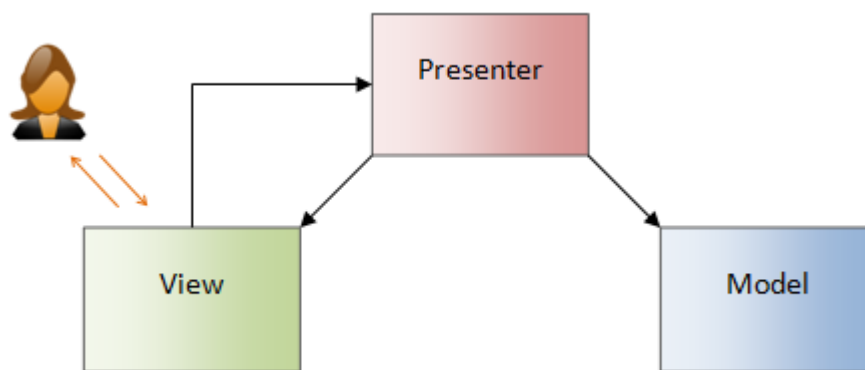
View (česky označovaný jako pohled) má za úkol zobrazovat data obsažená v modelech. Pohled obsahuje HTML kód doplněný výpisem proměnných PHP (často v cyklu), které dodají potřebná dynamická data. Aplikační logika se v pohledu nevyskytuje buď vůbec, nebo jen v minimálním potřebném množství. V praxi je často k jednomu modelu a controlleru připojeno více pohledů.

Controller přijímá požadavky od uživatele. Po vyhodnocení požadavku v případě nutnosti controller vyvolá aktualizaci modelu (spustí funkci pro vložení nových dat nebo úpravu stávajících) a vyvolá překreslení souvisejících pohledů.

Na obrázku 9 jsou naznačeny vazby mezi třemi výše popsány komponentami. Důležitá je vazba controlleru na model – ta umožňuje po zpracování požadavku od uživatele spouštět funkce modelu pro aktualizaci, vložení dat nebo jiné funkce, které jsou součástí aplikační logiky. Vazba pohledu na model slouží k načítání dat pro zobrazení. Je důležité si všimnout směru vazby – pohled může získávat data z modelu, ale model samotný nemá žádné reference na pohledy, je na nich naprosto nezávislý. Totéž platí o směru vazby controller => model. Některé varianty MVC spojují i controller s pohledem, tato vazba může být i oboustranná (na obrázku naznačeno čárkovaně). Vazba z modelu na controller nebo pohled nesmí existovat, porušil by se tím princip MVC. [22], [23]

#### 1.4.8 Architektura MVP

Existuje několik upravených variant MVC, jednou z nich je MVP – Model, View, Presenter. Tento upravený návrhový model odstraňuje nedostatky MVC. První změnou je ještě silnější oddělení modelu od pohledu – vazba mezi nimi je zrušena. Druhá věc je navázání oboustranné komunikace mezi pohledem (view) a presenterem, který nahrazuje původní controller.



Obrázek 10 – diagram MVP, vzor Passive View, zdroj: [22]

Jednou z často používaných variant MVP je vzor Passive View. Cílem této varianty je co nejvíce zjednodušit pohled. Ten obsahuje pouze šablonu HTML, do které presenter vloží požadovaná data. Pohled nově zajišťuje také příjem požadavků od uživatele. V rámci zachování jednoduchosti se při zachycení požadavku (například klepnutí na odkaz) zavolá příslušný presenter, který obsahuje funkci schopnou požadavek vyřídit. Presenter získá

potřebná data aktivací funkce modelu a pošle je zpět k vykreslení pohledem. Z těchto důvodů je nutná obousměrná vazba mezi pohledem a presenterem. [22]

### 1.4.9 Nette Framework

Pro tvorbu moderních webových aplikací se často používá některý z dostupných frameworků. Ve srovnání s programováním v čistém PHP přináší frameworky znatelné ulehčení práce, eliminaci bezpečnostních rizik a snadnější rozšiřitelnost aplikace. Jedním z nejrozšířenějších frameworků pro jazyk PHP je Nette Framework. Jedná se o open source framework využívající PHP 5. Původním autorem je David Grudl, v současnosti se o další vývoj stará organizace Nette Foundation. Aktuálně nejnovější verze je 2.3.9 vydaná v únoru 2016.

Základem Nette je MVP architektura s využitím passive view. Tvorba webové aplikace je relativně snadná, stačí si osvojit základní principy fungování Nette. Základem jsou dva konfigurační soubory – `config.neon` a `config.local.neon`. První jmenovaný obsahuje základní nastavení prostředí – časové pásmo, dobu expirace session a registraci modelů obsažených v aplikaci. Každý model, který má být použit, je nutné připsat do tohoto konfiguračního souboru. Druhý konfigurační soubor obsahuje informace o připojení k databázi. Další důležitou součástí je router, který podle předdefinovaných pravidel skládá URL adresy vedoucí k jednotlivým komponentám aplikace. Vlastní výkonný kód je zapsán v modelech a presenterech. Zejména presenter musí dodržovat správnou sémantiku v názvech funkcí a pohledů, aby správně fungovalo propojení mezi nimi. V Nette je mnohdy pro jeden presenter používá více pohledů. Každý z nich je obsluhován správně pojmenovanou funkcí presenteru. Název funkce pro komunikaci s pohledem vždy začíná slovem „render“, potom následuje pojmenování konkrétní funkce a stejný název musí nést i pohled. Pohledy obsahují kombinaci HTML a značkovacího jazyka, který umožňuje vkládat dynamicky generovaný obsah (ten pohledu pošle presenter).

Mezi hlavní výhody Nette patří:

- Jednoduché vytváření HTML formulářů s podporou validace vyplněných údajů.
- Podrobné sledování chyb při vývoji a ladění aplikace. Součástí je i analýza SQL dotazů.
- Převod URL adres na uživatelsky přívětivé.
- Zabezpečení vytvářené aplikace. [24]

#### **1.4.10 MySQL databáze**

MySQL patří mezi nejrozšířenější databázový systém používaný pro webové aplikace. Jedná se o multiplatformní software, který je kompatibilní s většinou v současnosti používaných operačních systémů. MySQL patří do kategorie open source produktů s licencí GPL. Původním tvůrcem systému je firma MySQL AB, v současnosti je majitelem společnost Oracle. Existují instalační balíky, které obsahují kompletní řešení pro provoz webových aplikací – obvykle je součástí MySQL, PHP a webový server Apache. Součástí bývá i grafický administrační nástroj pro správu databáze – phpMyAdmin. Tato nadstavba umožňuje pohodlně vytvářet strukturu tabulek, nastavovat vazby, přístupová práva a další věci. Samozřejmostí je i práce s daty. K tomu je možné použít nástroje grafického rozhraní, nebo spouštět přímo SQL dotazy. [36]

### **1.5 Zabezpečení webových aplikací**

V dnešní době webové aplikace dosáhly velkého rozšíření, a proto je nutné velmi zodpovědně přistupovat k otázce zabezpečení. Cílem útoku je obvykle krádež citlivých dat, podstrčení škodlivého kódu do aplikace, nebo snaha o vyřazení z provozu.

#### **1.5.1 Šifrované spojení mezi klientem a serverem**

Pokud webová aplikace pracuje s citlivými daty, je vhodné celou komunikaci zašifrovat. Často se používá protokol SSL, který pracuje mezi transportní vrstvou (TCP/IP) a vrstvou aplikační (HTTP). Webové aplikace využívající šifrované spojení se poznají podle URL adres začínajících na https://, běžné spojení pomocí http nesmí být povoleno.

Navázání šifrovaného spojení začíná fází handshake. První krok spočívá v poslání požadavku na SSL spojení od klienta na server. Server odpoví a pošle svůj certifikát. Klient podle certifikátu ověří totožnost serveru a vygeneruje základ šifrovacího klíče, který zakóduje veřejným klíčem serveru (ten je součástí certifikátu). Server svým soukromým klíčem dešifruje obdržený základ šifrovacího klíče a dohodne se s klientem na vytvoření session klíčů a šifrovacím algoritmu, který se bude používat pro další komunikaci. Tímto krokem fáze handshake končí a obě strany dále posílají zprávy zašifrované session klíčem s využitím zvoleného algoritmu. Fáze handshake využívá asymetrickou kryptografii (algoritmus RSA), pozdější komunikace pak symetrickou kryptografii (některý z algoritmů 3DES, RC4, RC2, DES). [25], [26]

### 1.5.2 Autentizace a autorizace uživatele

Pro zabezpečenou aplikaci je nutnost nějakým způsobem řídit přístup uživatelů do systému. Jedním ze základních typů je HTTP autentizace. Jedná se o omezení vstupu do aplikace jen vybraným uživatelům (identifikace pomocí jména a hesla), které je implementováno v HTTP protokolu – nastavuje se v konfiguraci webového serveru. Výhodou je, že tento způsob přihlašování je velmi jednoduchý a podporovaný širokým spektrem prohlížečů. Jednoduchost je ale vykoupena relativně závažnými nedostatky. Přihlášení probíhá formou standardního dialogového okna, které je napevno definováno webovým prohlížečem. Okno nelze žádným způsobem upravit ani integrovat do grafického rozhraní aplikace. Další problém je posílání přihlašovacích údajů na server – heslo není kryptograficky zabezpečeno. Uživatelské jméno i heslo je zakódováno pomocí algoritmu base64, což není bezpečné. V případě odposlechnutí komunikace lze snadno zachycený řetězec převést zpátky na jméno a heslo v čitelné podobě. Třetí nedostatek spočívá v nemožnosti odhlásit se z aplikace manuálně nebo automaticky po uplynutí stanovené doby. Uživatel zůstává přihlášen až do zavření okna prohlížeče.

Druhá možnost je naprogramovat si vlastní funkci, která bude řešit autentizaci a autorizaci. Pro tento účel se obvykle používá skriptovací jazyk na straně serveru – například PHP. PHP skript přijme uživatelské jméno a heslo poslané ze standardního HTML formuláře a zkontroluje, zda vyplněné údaje odpovídají záznamu v databázi. Heslo se z formuláře na server přenáší opět jen v textové formě a může být odposlechnuto. Řešením je použití šifrování v klientské části aplikace pomocí JavaScriptu. I tento způsob je napadnutelný – útočník může pozměnit JavaScriptovou část aplikace a šifrování obejít. Bezpečnější je připojovat se k serveru zabezpečeným připojením SSL, v tomto případě veškerá komunikace probíhá kódovaně. SSL je možné použít i ve spojení s HTTP autentizací. Výhodou vlastní přihlašovací funkce je snadná integrace přihlašovacího formuláře do aplikace. Identita přihlášeného uživatele se ukládá do session, což výrazně ulehčuje autorizaci uživatelů pro různé části aplikace. Při pokusu vstoupit do části aplikace jde snadno zkontrolovat, jestli právě přihlášená osoba má dostatečná práva a případně vstup zakázat. S použitím vlastní přihlašovací funkce a session není problém implementovat automatické odhlášení po uplynutí stanoveného časového limitu, nebo ruční odhlášení. Obojí se snadno provede přepsáním hodnot v session, nebo její úplným zrušením. [26]

### 1.5.3 Ošetření vstupů aplikace a SQL Injection

Další krok v zabezpečení aplikace proti napadení je důsledné ošetření vstupů. Všechna místa, kde uživatel má možnost vkládat data nebo měnit některé parametry, jsou potenciálně nebezpečná a mohou sloužit k vniknutí útočníka do systému. Mezi tato místa patří:

- HTML Formuláře.
- Upload souborů na server pomocí funkcí aplikace.
- URL adresa požadavku.

Zásadní pravidlo pro kontrolu příchozích dat je provádět validaci vždy na serverové straně aplikace. Validace na straně klienta (JavaScript) se hodí pro kontrolu správnosti vložených údajů (rozsah hodnot, formát dat) z hlediska uživatelského – lze snadno informovat o chybách ve vkládaných hodnotách. Bezpečnostní kontrola vstupu ale musí být provedena na serveru, protože skripty na straně uživatele může útočník pozměnit. Toto se týká zejména HTML formulářů. Ošetření vstupu standardně má za úkol zabránit vložení škodlivých skriptů. Jednou z možností je analyzovat vložená data na přítomnost speciálních znaků. Nepovolené znaky jsou z textu buď odstraněny, nebo převedeny na HTML entity, což případný útočný skript učiní nefunkčním.

Možnost nahrávat na server soubory také patří mezi rizikové operace. V případě nedostatečné kontroly by se mohlo stát, že útočník dokáže na server dostat vlastní soubor obsahující škodlivý kód a ten pomocí dalších metod spustit. Pokud aplikace pro svoji činnost potřebuje uživatelům povolit upload externích souborů, funkce zpracovávající soubory by měla kontrolovat typ nahraných souborů a umožnit zapsat na server jen vybrané typy.

Většina webových aplikací používá pro spojení jednotlivých částí HTML odkazy. URL adresu obsaženou v odkazu aplikace dekóduje a podle následně zobrazí požadovanou stránku. Jednou z možností je připojit pomocí PHP funkce include HTML nebo jiný objekt, jehož název je specifikován v adrese odkazu. Nezabezpečený kód použije název z URL jako argument funkce include, která objekt načte a zobrazí ve stránce. Útočník se může jednoduše pokusit získat libovolný soubor ze serveru tím, že místo názvu objektu napíše přímo cestu k souboru, u kterého předpokládá, že se na serveru nachází. V případě platné cesty soubor získá. Ochrana proti tomuto typu napadení spočívá v důkladné kontrole parametrů URL a poskytnutí pouze povolených souborů.

SQL Injection je druh útoku, jehož cílem je napadnutí databáze za účelem získání dat nebo naopak jejich poškození. Naprostá většina webových aplikací používá ke své činnosti databázi. Funkce pracující s databází jsou obvykle nějakým způsobem spojené se vstupy od uživatele – HTML formuláře nebo parametry URL. Do nezabezpečené aplikace se dá snadno vložit škodlivý SQL kód. Nejlépe přístupným místem, kde se útočník může pokusit napadnout databázi, je přihlašovací formulář. Následující kód na straně aplikace je snadno napadnutelný:

```
$sql = "SELECT * FROM users WHERE name = '". $_POST['jmeno'] ."'";"
```

Pro přístup do aplikace bez znalosti správných přihlašovacích údajů stačí zadat místo uživatelského jména řetězec ' or '1'='1. Druhá možnost, jak zaútočit na stejný nezabezpečený přihlašovací formulář, je vložení řetězce 123; DROP TABLE users. Většina databázových systémů podporuje vykonání více SQL dotazů ve formě dávky. Toho může útočník využít k poškození databáze a vyřazení aplikace z provozu. Výše uvedený řetězec vloží uživatelské jméno 123, následně středníkem ukončí původní dotaz a vloží další, který smaže tabulku uživatelů a tím zabráni všem uživatelům v přístupu do aplikace.

Proti SQL Injection se lze bránit více způsoby. Jedna možnost je kontrolovat text na vstupu a zakázat veškeré speciální znaky. Často se používá principu „prepared statements“, kde se parametry ze vstupu nekládají přímo do dotazu, ale posílají se zvlášť. Kromě zabezpečení proti SQL Injection je tento přístup výhodný i z hlediska rychlosti provádění dotazů. Příprava dotazu proběhne pouze jednou a následně se do předkompilovaného dotazu pouze vkládají odlišné parametry. [26], [27]

#### **1.5.4 XSS (Cross Site Scripting)**

Se zabezpečením vstupů aplikace úzce souvisí XSS. Jestliže útočník zjistí nezabezpečený vstup do aplikace, může se pokusit použít tento typ útoku. XSS využívá skripty spouštěné na straně klienta ve webovém prohlížeči. Nejrozšířenější jsou tedy útočné skripty psané v JavaScriptu. Principem XSS je spuštění škodlivého skriptu na počítači oběti a získání citlivých dat (například získání session ID z cookies). Existují dva hlavní typy XSS útoku.

Okamžitý útok. Tento druh útoku neovlivňuje samotnou aplikaci, ani se proti němu nedá v rámci aplikace zavést protiopatření. Nejčastěji se takto útočí na aplikace, které obsahují funkci pro vyhledávání nebo jinou funkci, která využívá parametry napsané přímo v URL adrese. Útočník vytvoří podvržený odkaz obsahující škodlivý skript a přesvědčí uživatele, aby na odkaz kliknul. Webový prohlížeč kód spustí a útočník získá přístup k citlivým datům.



Perzistentní útok. V tomto případě se už využívají vstupní místa aplikace. Častým příkladem jsou diskuzní fóra nebo návštěvní knihy. Pokud taková aplikace nekontroluje vložená data, útočník má možnost vložit kód obsahující škodlivý skript a uložit ho do databáze napadené aplikace. Návštěvník takové stránky zobrazí vložený obsah, automaticky se spustí útočnickův skript a ten získá přístup k cookies oběti. Z cookies může útočník zjistit session ID, které lze použít k přihlášení na server pod identitou oběti bez znalosti přihlašovacích údajů.

Pro obranu proti XSS útokům je velmi důležité ošetřit všechny vstupy aplikace proti vložení škodlivého kódu, tak jak je popsáno v předchozí kapitole. V případě, že pro správnou funkci aplikace je potřeba, aby uživatelé mohli vkládat formátovaný text (různé písmo, obrázky, odkazy), pak je vhodné implementovat vlastní systém značek a ty pak před zobrazením konvertovat na standardní HTML značky. Tento princip používá například systém phpBB. [28]

## **1.6 Algoritmus pro výpočet trasy**

Poslední částí rozvozu balíků je jejich doručení adresátovi. Jednotlivé cílové adresy jsou tedy vlastně uzly grafu a cílem je nalézt jejich optimální spojení. Každá spojnice je určitým způsobem ohodnocena a na základě toho je nutné určit optimální propojení uzlů. Existují různé způsoby ohodnocení – například podle času, silniční vzdálenosti nebo spotřeby paliva. Problém je tedy celkem jednoduše definován. Jedná se o velmi starou "Úlohu obchodního cestujícího" a je to složitý diskrétní optimalizační problém.

### **1.6.1 Úloha obchodního cestujícího**

Nejjednodušším algoritmem by bylo vytvořit cesty mezi všemi uzly, vypočítat jejich ohodnocení a najít optimální řešení. Takto lze však úlohu řešit pouze do počtu několika desítek uzlů. Problémem pro větší počet uzlů se stane časová náročnost výpočtu. Pro počet všech možných cest totiž platí, že je roven faktoriálu z počtu bodů, takže náročnost je faktoriální. Pro praktické použití je vhodné najít časově efektivní algoritmus se suboptimálním výsledkem. Pro zjednodušení úlohy se také někdy může použít pravidlo, že ohodnocení spojnice bodů splňuje trojúhelníkovou nerovnost. Další možné zjednodušení je, zda ohodnocení spojnice z uzlu A do uzlu B je shodné jako z B do A. Pokud ano, matice vzdáleností je symetrická.

Pro řešení této úlohy bylo vypracováno mnoho více či méně efektivních algoritmů různé složitosti. Cílem této práce však není optimální řešení této pod-úlohy, takže byl navržen a použit relativně jednoduchý algoritmus. Tento algoritmus nemusí být nejlepším řešením této úlohy.

Přepavní společnost využívá pro určení místa doručení poštovní adresy. Existují systémy, které na základě zadané posloupnosti poštovních adres určí optimální propojení těchto bodů. Jedním z poskytovatelů této služby je OptiMap. Služba je dostupná na adrese [<http://gebweb.net/optimap/>]. Adresy se zadávají do online formuláře. Možné je i dávkové zpracování adres. Tyto systémy však mají významné omezení na počet bodů. Další nevýhoda je, že se jedná o „černou skříňku“ – implementace vyhledávacího algoritmu není známá. Z těchto důvodů je výhodnější použít jiný způsob. Určení bodu pro doručení zásilky ve formě poštovní adresy je vhodné převést na souřadnicové určení polohy místa. [30], [31]

### **1.6.2 Typy souřadných systémů**

Země je fyzikální těleso a jeho tvar je výslednicí dvou sil, síly přitažlivé a odstředivé. Výslednicí obou sil je tíhová síla  $G$ . V geodetickém smyslu je Zemí nulová hladinová plocha ve všech místech kolmá na směr gravitace. Tato plocha se nazývá geoid. Povrch Země i geoid jsou však nepravidelná, členitá a velmi složitá tělesa, proto jsou pro potřeby mapování aproximována referenčními plochami. Těmito plochami mohou být referenční elipsoid, referenční koule a referenční rovina. Nejblíží skutečnému tvaru je elipsoid. U nás se v civilním sektoru využívá Besselův elipsoid a ve vojenském sektoru elipsoid Krasovského. Velmi užívaným je také elipsoid Hayfordův, který byl v roce 1924 přijat za mezinárodní elipsoid. Pro metody měření pomocí GPS je používán elipsoid WGS-84. Geometrické výpočty na elipsoidu však také nejsou jednoduché. O něco jednodušší jsou výpočty na kouli, zabývá se jimi sférická trigonometrie. Elipsoid se koulí nahrazuje pro část povrchu. V případě menších požadavků na přesnost se nahrazuje celý povrch. Pokud se nahrazuje část elipsoidu, používá se pro území o poloměru do 200 km. Pro nahrazení celého elipsoidu se poloměr náhradní koule určuje více způsoby:

- Koule by měla mít stejný objem jako elipsoid.
- Koule by měla mít stejný obsah jako elipsoid.
- Poloměr koule by měl být rovný aritmetickému průměru všech tří poloos elipsoidu.
- Délky kvadrantů by měly být stejné.

I na kouli jsou výpočty poměrně složité. Proto se pro menší území přechází na rovinné zobrazení.

Pro Českou republiku se používá několik rovinných systémů souřadnic. Například souřadnicový systém jednotné trigonometrické sítě katastrální (S-JTSK) je definován Besselovým elipsoidem s referenčním bodem Hermannskogel a Křovákovým zobrazením. Souřadnicový systém S-42 používá Krasovského elipsoid s referenčním bodem v Pulkavu. Souřadnice bodů jsou vyjádřené v  $6^\circ$  a  $3^\circ$  pásech Gaussova zobrazení. V poslední době je velmi používaný souřadnicový systém WGS 84. Jeho referenční plochou je elipsoid WGS 84 (World Geodetic System).

V současné době je asi nejvíce rozšířen systém určení souřadnic bodu pomocí systému GPS. Tento systém je založen na skupině družic, které obíhají kolem Země ve výšce asi 20000 km. Tyto družice nepřetržitě vysílají signál. Signály jsou přijímány přijímačem a vyhodnoceny. K určení polohy je nutný příjem signálu aspoň ze tří družic, pro výpočet polohy i s výškou jsou potřeba čtyři družice. Přesnost při dobrém příjmu se pohybuje v jednotkách metrů, což pro běžné použití vyhovuje. Pro vyšší nároky na přesnost je možné i přesnější určení polohy.

Pro převod poštovní adresy do souřadného systému je výhodné použít Google Maps Geocoding API. Tuto službu je možno bezplatně použít do počtu 2500 dotazů za den s maximální frekvencí 10 dotazů za sekundu. Tato omezení jsou pro potřeby navržené aplikace akceptovatelná. Výsledkem je přiřazení GPS souřadnic k cílové adrese zásilky. Jedná se o zeměpisné souřadnice, bod je určen zeměpisnou šířkou a délkou. Šířka se počítá od rovníku na sever (severní šířka) a na jih (jižní šířka) – nabývá hodnot  $0^\circ$  až  $90^\circ$ . Zeměpisná délka se počítá od nultého poledníku a rozlišuje se východní a západní délka – nabývá hodnot  $0^\circ$  až  $180^\circ$ .

Pro potřeby přepravní společnosti je potřeba pracovat s velkým množstvím bodů trasy – v řádu desítek. Pro optimální rozdělení zásilek mezi jednotlivá vozidla je třeba úlohu optimální trasy řešit mnohokrát. Navržená aplikace je určena pro rozvoz balíků v relativně malé oblasti, což snižuje nároky na přesnost. Proto je možné body určené GPS souřadnicemi převést do systému bodů v rovině určených v pravoúhlém souřadném systému.

U převodu ze zeměpisných souřadnic do rovinných je podstatná požadovaná přesnost. Pro rostoucí přesnost se výpočet výrazně komplikuje. Existuje více metod převodu souřadnic s různou úrovní náročnosti výpočtu a přesnosti – v případě složitých algoritmů řádově centimetry. Plánování trasy rozvozu zásilek nevyžaduje tak velkou přesnost, proto je

výhodnější použít jednodušší metodu převodu. Takovou metodou je Ekvidistantní válcová projekce (někdy nazývaná Marinovo zobrazení). Výsledkem je zobrazení Země ve formě pravidelné pravoúhlé sítě úseček, které reprezentují poledníky a rovnoběžky. Tato metoda je jednoduše realizovatelná a pro malé oblasti poskytuje dostatečnou přesnost. [32], [33]

### 1.6.3 Navržený algoritmus pro řešení problému obchodního cestujícího

Pro hledání trasy je vhodné použít výše zmíněný pravoúhlý systém souřadnic upravený tak, aby depo bylo na souřadnicích  $[0;0]$ . Tato poloha depa je výhodná pro rozdělování zásilek do úhlových sektorů. V tomto rovinném systému je nejjednodušší hledat přímková spojení jednotlivých bodů. Takovéto zjednodušení může citelně zhoršit nalezený výsledek. Pro oblasti s poměrně hustou silniční sítí však výsledek bude použitelný. Pro výpočet optimální trasy je použita symetrická matice vzdáleností všech bodů pro doručení zásilek. Matice je vytvořena ze souřadnic bodů před začátkem výpočtu trasy, poloha samotných bodů se v algoritmu nepoužívá. Pokud by distanční matice byla namísto přímých vzdáleností vyplněna silničními vzdálenostmi, výsledek hledání by byl přesnější.

Úloha, pro kterou je potřeba najít řešení spočívá v nalezení pokud možno nejkratší trasy mezi všemi body (adresy pro doručení zásilek), přičemž začátek a konec cesty je vždy v depu. Rozdělení bodů kolem depa se předpokládá náhodné. Pro všechny body (včetně depa) je vypočítána distanční matice, pro zjednodušení se uvažují přímé vzdálenosti. Matice je symetrická, vzdálenost mezi dvěma body je pro oba směry shodná. Algoritmus hledání optimální trasy spočívá v postupném hledání nejbližšího volného uzlu. Tento nejbližší uzel se stává dalším bodem trasy a z něj se hledá další, zatím nepoužitý nejbližší uzel. Po spotřebování všech bodů se trasa dokončí návratem do depa. Tento algoritmus lze nazvat „jednostranný“. Je zřejmé, že tento algoritmus se často může dostat do nejvzdálenějšího bodu od depa a odtamtud následuje dlouhý návrat. Proto je vhodné provést modifikaci tohoto algoritmu. Z depa (počáteční a zároveň koncový bod) se střídavě tvoří cesta na obě strany.

- Z počátečního bodu se najde nejbližší bod a zaznamená se do subtrasy č. 1 (první segment)
- Z koncového bodu se najde další volný nejbližší bod a zaznamená se do subtrasy č. 2 (poslední segment).

Takto se střídavě postupuje až do vyčerpání všech bodů. Na závěr se k subtrase č. 1 připojí subtrasa č. 2 v obráceném pořadí. Tento algoritmus lze tedy nazvat „dvoustranný“. Oba tyto algoritmy byly odzkoušeny na mnoha příkladech a trasy graficky znázorněny pomocí

spojnicového grafu. Grafické znázornění odhalilo i některé nedostatky. Někdy na trase vznikla lokální překřížení nebo bylo zřejmé, že prohození pořadí dvou bodů trasu zkrátí. Pro odstranění těchto jevů byla přidána ještě optimalizace nalezené trasy.

Optimalizace spočívá v tom, že pro pět po sobě jdoucích bodů trasy se pořadí třech vnitřních bodů zvolí tak, že tato trasa z pěti bodů je nejkratší. Znamená to určit všechny možnosti a vybrat nejkratší trasu. Existuje tedy šest možností pro tři měněné body. Po nalezení optimální posloupnosti se hledání posune o jeden bod dále. Tímto způsobem se projde celá trasa od začátku do konce. Pokud dojde k prohození bodů, je toto zaznamenáno. Dále se zkusí optimalizovat celá trasa v opačném směru – od konce k začátku. Pokud žádné prohození bodů při optimalizaci celé trasy neproběhlo, pokusy o optimalizaci se ukončí. Počet i úspěšných optimalizací je maximálně 4. Na grafickém znázornění je vliv optimalizace jasně patrný. Je zřejmé, že tato optimalizace nemůže zabezpečit globálnější optimalizaci trasy.

Oba algoritmy spolu s výše popsanou optimalizací byly odzkoušeny na množinách pseudonáhodně vygenerovaných uzlů. Uzlů bylo 30 včetně depa. Poloha depa byla na souřadnicích [0;0], ostatní uzly měly souřadnice X a Y o hodnotě maximálně 49. Rozdělení balíků pro jednotlivá vozidla funguje na základě úhlových sektorů (výsečí) vzhledem k depu. Proto bylo chování algoritmů zkoumáno i z hlediska umístění bodů v úhlových sektorech. Pro každý zvolený sektor proběhlo testování pro 10000 tras. Některé číselné výsledky jsou uvedeny v tabulce. Úhlové sektory uzlů jsou uváděny ve stupních v matematicky kladném směru od osy x.

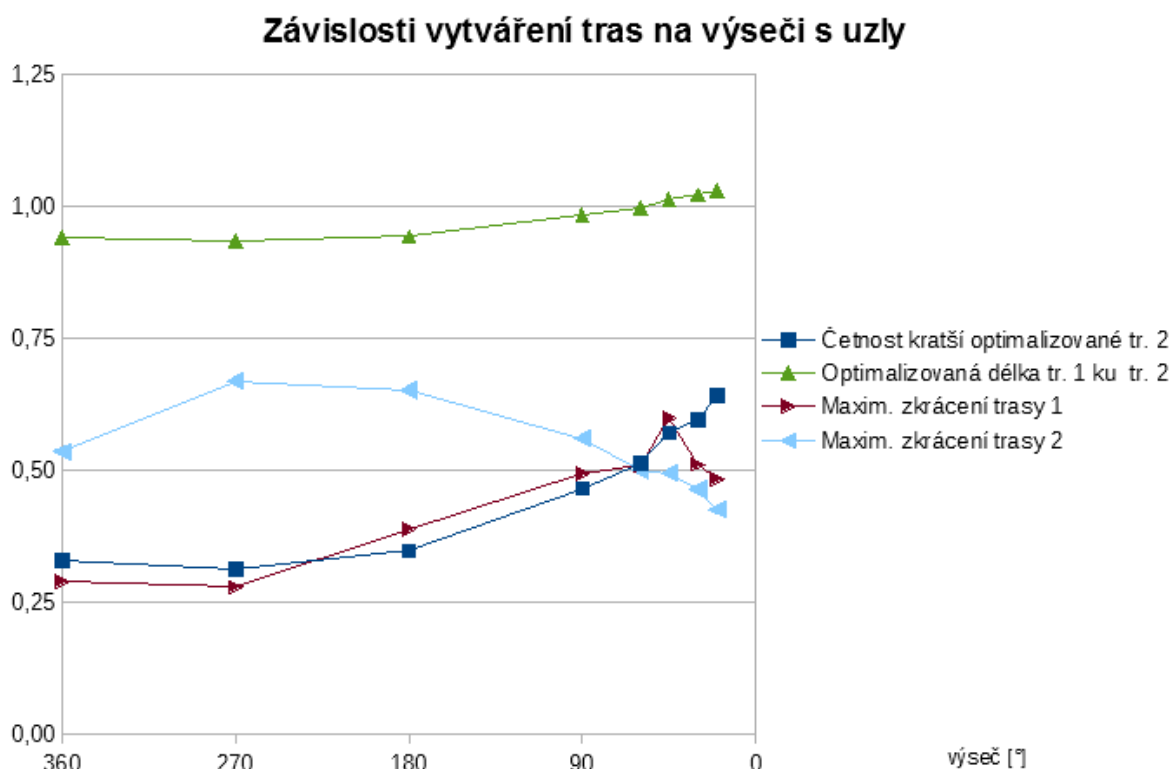
**Tabulka 2 – porovnání účinnosti optimalizací**

	Úhlový sektor pro uzly [°]							
	360	270	180	90	60	45	30	20
Relativní četnost kratší optim. tr. 2	0,3274	0,3110	0,3464	0,4639	0,5131	0,5702	0,5953	0,6415
Průměrná optimalizace tr. 1	0,9701	0,9680	0,9660	0,9656	0,9641	0,9631	0,9641	0,9665
Průměrná optimalizace tr. 2	0,9526	0,9534	0,9521	0,9526	0,9548	0,9524	0,9571	0,9637
Prům. optimal. délka tr. 1 ku tr. 2	0,9405	0,9327	0,9424	0,9823	0,9951	1,0128	1,0207	1,0282
Maximální zkrácení trasy 1	0,2888	0,2772	0,3866	0,4929	0,5077	0,5975	0,5098	0,4818
Maximální zkrácení trasy 2	0,5340	0,6679	0,6509	0,5597	0,4992	0,4938	0,4633	0,4244

Pro tabulku platí:

- Trasa 1 označuje algoritmus hledání trasy z jedné strany.
- Trasa 2 označuje algoritmus hledání trasy z obou stran.

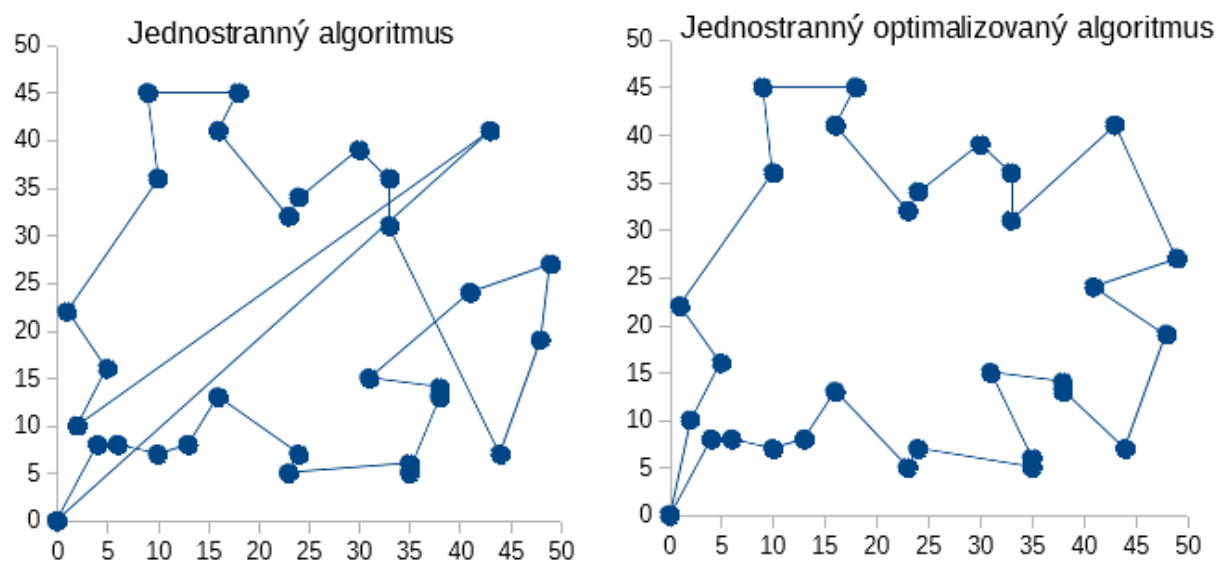
První řádek (relativní četnost kratší optimalizované tr. 2) udává relativní četnost jevu, že optimalizovaná trasa 2 je kratší než trasa 1. Hodnota průměrná optimalizace trasy 1 je podíl součtu všech délek optimalizovaných tras 1 ku součtu všech délek neoptimalizovaných tras 1. Hodnota průměrná optimalizace trasy 2 je jako výše uvedené pro trasu 2. Hodnota průměrná optimalizace délka trasy 1 ku trase 2 je podíl součtu všech délek optimalizovaných tras 1 ku součtu všech délek optimalizovaných tras 2. Další dva řádky uvádějí maximum zkrácení trasy optimalizací ku délce optimalizované trasy.



**Obrázek 11 – graf závislosti trasy na velikosti úhlového sektoru**

Pro zjištění trendu je vhodné některé hodnoty znázornit graficky. Dostí jasný je trend, že pro menší úhlový sektor je statisticky výhodnější algoritmus „dvoustranný“. Úhel, kde jsou oba přibližně stejně úspěšné, je asi 60°. Toto je zřejmé z četnosti kratší optimalizované trasy 2 i průměrné optimalizované délky trasy 1 ku trase 2. Přímou z tabulky je naopak vidět, že průměrná optimalizace jednostranného i dvoustranného algoritmu na úhlovém sektoru příliš nezáleží a také je u obou algoritmů skoro shodná. Celkem zajímavé je, že i přes nepříliš velkou průměrnou velikost optimalizace může být maximální zkrácení trasy optimalizací celkem značné. Pro oba algoritmy se tyto hodnoty v závislosti na úhlovém sektoru chovají rozdílně. V případě dvoustranného algoritmu je zkrácení největší pro asi 270°. U jednostranného algoritmu je pro tento úhel naopak zkrácení trasy nejmenší. Pro menší úhly se

hodnoty k sobě přiblíží. Tato celkem významná zkrácení trasy jsou často způsobena tím, že při procesu hledání nejbližšího nepoužitého uzlu se dojde blízko k depu, ale jeden vzdálený bod zůstane „zapomenutý“. Tím dojde k nárůstu délky trasy. Vliv optimalizace je nejlépe viditelný na níže uvedených znázorněních trasy.



Obrázek 12 – znázornění cesty nalezené jednostranným algoritmem

Na základě analýzy obou algoritmů a jejich optimalizací byl učiněn následující závěr. Oba algoritmy jsou funkční. Navržená optimalizace trasy nikdy neprodloužila. Pro některé případy je zkrácení trasy optimalizací významné. Výpočetní náročnost i pro stovky bodů není vysoká. Pro výpočet trasy je vhodné použít oba algoritmy, trasy optimalizovat a vybrat nejkratší. [29]

## **2 PRAKTICKÁ ČÁST**

### **2.1 Návrh informačního systému**

#### **2.1.1 Informační systém pro přepravní společnost**

Předmětem práce je navrhnout a naprogramovat funkční aplikaci, která bude obsahovat nástroje sloužící pro evidenci a logistické řízení přepravní společnosti. Aplikace nebude nasazena do ostrého provozu, nicméně všechny implementované funkce jsou navrženy s ohledem na snadné použití a optimální využití zdrojů.

#### **2.1.2 Základní informace o aplikaci**

Navržená aplikace je určena pro firmy zabývající se rozvozem zásilek mezi městy a vesnicemi. Plánovaný dosah od depa k místu doručení zásilky se pohybuje řádově v desítkách kilometrů. Rozvoz probíhá po automaticky naplánovaných trasách s důrazem na minimalizaci nákladů (problém obchodního cestujícího). Pro navrhování tras s optimálním využitím prostředků je nutné, aby systém měl přístup ke všem potřebným informacím. Pojmem prostředky se v tomto případě označují vozidla a zaměstnanci firmy – řidiči. Jednou z klíčových komponent aplikace je evidence řidičů – po každé jízdě se příslušnému řidiči aktualizuje počet realizovaných rozvozů a množství doručených zásilek. Další modul se stará o evidenci vozového parku společnosti. U vozidel patří mezi pravidelně aktualizované hodnoty stav tachometru a datum servisních prohlídek. Aplikace automaticky upozorní na blížící se prohlídku v servise nebo STK. Třetím klíčovým modulem je správa zásilek. Obsluha systému zadá do aplikace údaje o zásilkách, které se mají doručit a následně spustí generování tras pro rozvoz. Algoritmus pro výpočet trasy zpracuje dostupné informace a navrhne pokud možno optimální trasu jízdy. Cílem je rovnoměrně vytížit všechna dostupná vozidla i řidiče. Zároveň je prioritou i úspora nákladů – pro rozvoz se použije co nejmenší možný počet vozidel při zachování akceptovatelné rychlosti rozvozu.

Předpokládaný způsob použití je provoz na samostatném počítači nebo vnitřní firemní síti. Navržený systém nemá žádné rozhraní pro veřejný přístup, aplikaci mohou používat jen zaměstnanci, kteří mají založen uživatelský účet. Po úspěšném přihlášení se zobrazí aplikace s hlavním menu. Nabízené funkce závisí od uživatelské role a přidělených přístupových práv konkrétního zaměstnance. Aplikace je zabezpečena proti neoprávněnému přístupu (bez platných přihlašovacích údajů), chráněn je rovněž i přístup k modulům, které nejsou pro přihlášeného uživatele dostupné z důvodu nedostatečných práv.

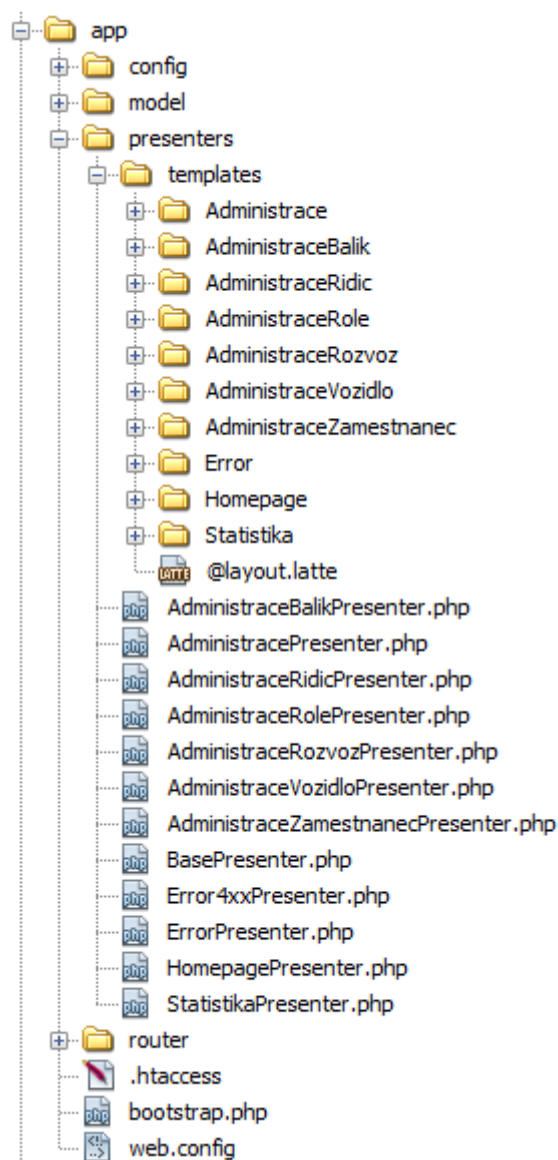


### 2.1.3 Použité technologie

Hlavní komponentou navrženého systému je PHP Framework Nette. Nette výrazně usnadňuje práci při programování aplikace – využití vestavěných funkcí pro práci s databází nebo tvorbu formulářů značně ulehčí práci a zároveň se pozitivně projeví na bezpečnosti. Nette je založeno na skriptovacím jazyce PHP, využívá i JavaScript, CSS a finální výstup, který se zobrazí ve webovém prohlížeči, je ve formátu HTML. K ukládání dat slouží databáze MySQL, samozřejmostí je ošetření všech vstupů aplikace, aby nemohlo dojít k napadení databáze a potenciálnímu ohrožení dat.

## 2.2 Architektura

### 2.2.1 Adresářová struktura projektu



Obrázek 13 – struktura projektu

Nette projekt se skládá z velkého množství komponent, které jsou uspořádány do stromové struktury. Nejdůležitější podsložky jsou `app` a `www`.

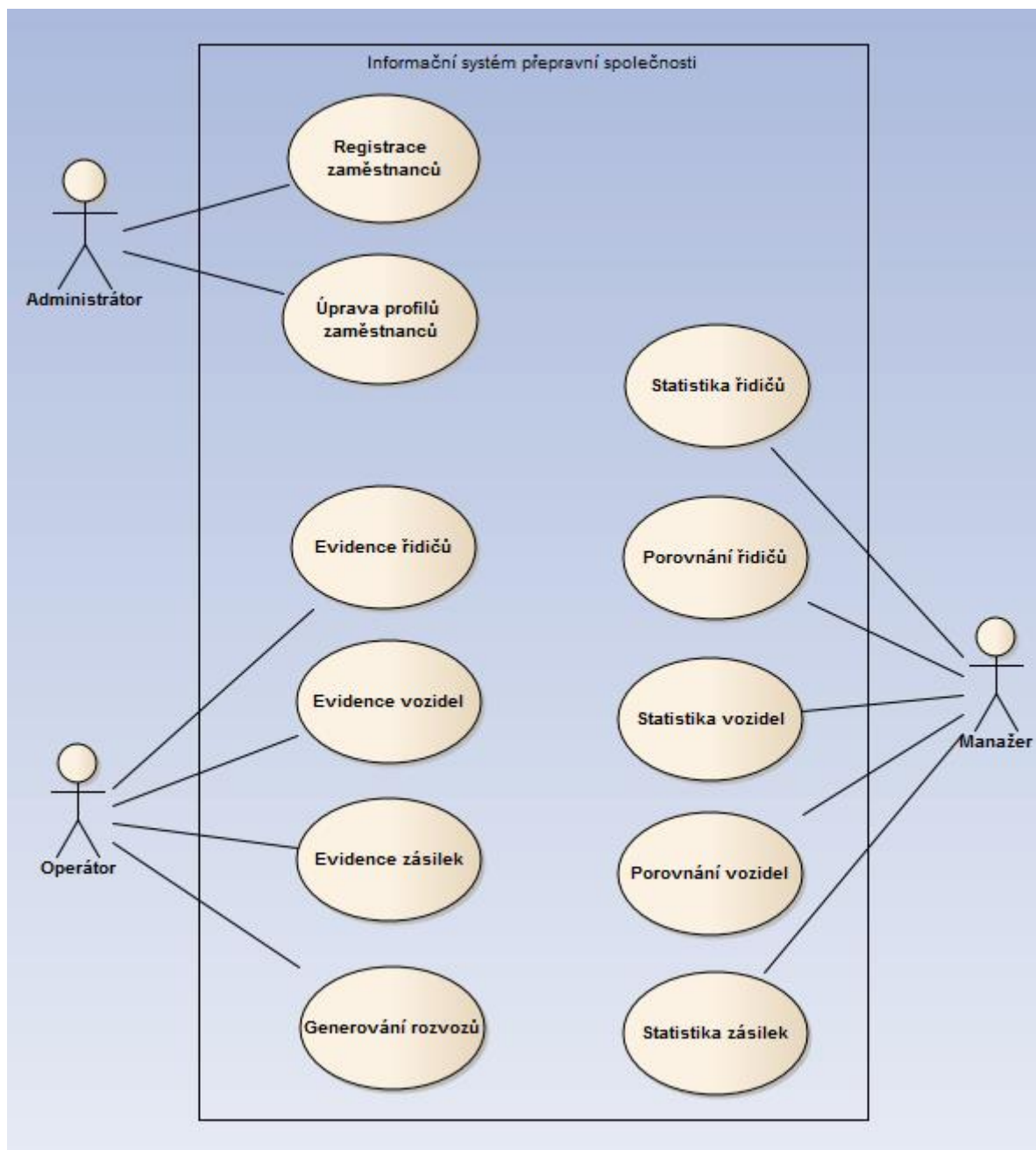
Složka `app` se dále dělí na složky `config`, `model`, `presenters` a `router`. `Config` obsahuje dva konfigurační soubory – `config.neon` a `config.local.neon`. Pomocí těchto souborů se nastavuje časové pásmo, doba platnosti session, strukturu umístění presenterů, registrace modelů a parametry připojení k databázi. Složka `model`, jak už napovídá název, obsahuje PHP třídy obsahující aplikační logiku – v architektuře MVC nazvané modely. Ve složce `presenters` jsou uloženy všechny presentery, které aplikace využívá. Výchozí konfigurace Nette vyžaduje, aby názvy tříd vždy končily řetězcem „Presenter“, v opačném případě by nefungovalo korektně skládání odkazů. Pohledy spolupracující s presentery jsou v projektu

umístěny ve složce templates. Každý z presenterů má přiřazenu jednu složku pohledů, která musí mít odpovídající název. Struktura modelů, presenterů a pohledů používaných jednotlivými částmi aplikace budou podrobněji popsány v následujících kapitolách. Do části projektu app patří ještě složka router obsahující třídu Router Factory. Zde se nastavuje struktura URL odkazů, které propojují části aplikace mezi sebou. Jednou z možností je i přepisování adres na tzv. hezké URL. Posledními komponentami části app jsou soubory bootstrap.php a .htaccess. Bootstrap stanovuje cesty ke konfiguračním souborům (popsaným výše) a dalším částem projektu. Obvykle není nutné výchozí konfiguraci měnit. Soubor .htaccess slouží ke konfiguraci webserveru Apache. V tomto případě je zakázán veškerý přístup z webového prohlížeče ke složce app pro ochranu zdrojových kódů.

Druhou klíčovou částí aplikace je složka www. Ta na rozdíl od app nemá zablokovaný přístup. Jedná se o tzv. document\_root – tedy kořenový adresář aplikace dostupný přes webový prohlížeč. Obsahem jsou všechny obrázky, CSS styly, skripty a obecně soubory přímo dostupné z webového prohlížeče. Pro zobrazení obsahu vlastní aplikace, která je definována v části app, slouží soubor index.php. Ten přesměruje příchozí požadavky právě do neveřejné složky app, kde se o zobrazení požadované stránky postarají zdrojové soubory.

### **2.2.2 Use case diagram – uživatelské role**

Informační systém pracuje se třemi rolemi. První role je administrátor. Jak již napovídá název, administrátor může obsluhovat všechny funkce aplikace. Jeho hlavním úkolem je přidávat (registrovat) do systému další zaměstnance, kteří budou pracovat s přidělenými částmi aplikace. Druhá role je klíčová pro normální fungování systému. Operátor pracuje se systémem denně a udržuje ho v chodu. Náplní jeho práce je obsluhovat evidenci vozidel, řidičů a zásilek – operátor přidává do databáze přijaté zásilky a následně z nich generuje rozvozy pro další den. Poslední uživatelská role je manažer. Ten má přístup pouze k modulu statistik, kde může sledovat množství doručených zásilek a vytížení řidičů a vozidel.



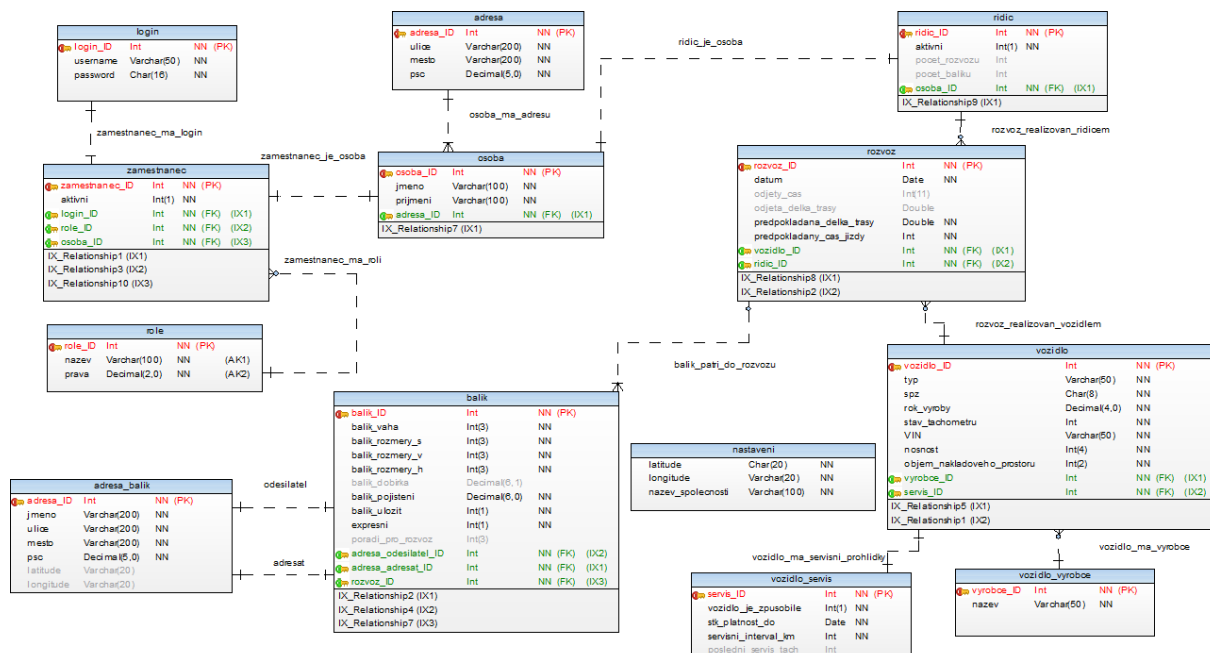
Obrázek 14 – Use case diagram

### 2.2.3 UML Class diagram

Celá aplikace se skládá celkem z 16 tříd, z toho 7 modelů, 8 presenterů (+1 BasePresenter) a 27 pohledů. Digram tříd celé aplikace je příliš rozsáhlý pro vložení do tištěné práce, proto je obsažen v příloze (formát PNG a EAP – Enterprise Architect). Ačkoliv je model tříd složený z mnoha komponent, nejsou příliš provázané, což umožňuje diagram tříd rozdělit do menších logických celků podle modulů aplikace. Ty budou podrobněji popsány v kapitole 2.4.

## 2.3 Datový model

### 2.3.1 ER diagram



Obrázek 15 – ER diagram

Obrázek 15 obsahuje databázový model použitý v navržené aplikaci. Model ve formátu TXP (Toad Data Modeler) a ve formátu PNG je součástí přílohy. Všechna data potřebná pro provoz aplikace jsou uložena v celkem 12 tabulkách. Každá tabulka obsahuje primární klíč navázaný na sloupec ID. Při vložení nového záznamu je ID vygenerováno automaticky pomocí funkce MySQL – autoincrement. Podrobnější popis tabulek a vazeb mezi nimi je popsán v následující kapitole.

### 2.3.2 Tabulky v databázi – atributy, datové typy, vazby

Tabulka 3 – login

Klíč	Sloupec	Datový typ	Popis
PK	login_ID	Int(11)	ID loginu
	username	Varchar(50)	Uživatelské jméno pro přihlášení
	password	Char(32)	Zašifrované heslo
	salt	Char(32)	Dodatek k heslu

Kardinalita	Rodič	Potomek	Název vztahu
1:1	login	zamestnanec	zamestnanec_ma_login

Tabulka login obsahuje přihlašovací údaje uživatelů systému. Vazba spojuje pouze tabulky login a zamestnanec, což značí, že pouze zaměstnanci společnosti mají možnost pracovat s informačním systémem, řidiči tuto možnost nemají. Heslo je v databázi uloženo šifrovaně pomocí funkce SHA-256. Zabezpečení je dále navýšeno použitím náhodně generovaného řetězce salt, který je použit ve spojení s heslem.

**Tabulka 4 – uživatelské role**

Klíč	Sloupec	Datový typ	Popis
PK	role_ID	Int(11)	ID role
	nazev	Varchar(10)	Název role
	prava	Decimal(2,0)	Přístupová práva – číslo

Kardinalita	Rodič	Potomek	Název vztahu
1:N	role	zamestnanec	zamestnanec_ma_rolí

Aplikace vyžívá k přístupu uživatelské role – ty jsou v databázi uloženy ve výše uvedené tabulce. Kromě automaticky generovaného ID je role reprezentována celým číslem (atribut prava) a názvem, který je po přihlášení vypsán vedle jména uživatele.

**Tabulka 5 – adresy**

Klíč	Sloupec	Datový typ	Popis
PK	adresa_ID	Int(11)	ID adresy
	ulice	Varchar(200)	Ulice a číslo popisné
	mesto	Varchar(200)	Město
	psc	Decimal(5,0)	PSČ

Kardinalita	Rodič	Potomek	Název vztahu
1:N	adresa	osoba	osoba_ma_adresu

Tabulka adresa slouží k uložení části osobních údajů zaměstnanců a řidičů, kteří pracují ve firmě. Pro uložení adresy jsou použity tři sloupce – v prvním z nich se zapisuje jméno ulice a číslo popisné, další dva pak obsahují město a PSČ. Adresa je spojena s tabulkou osoba

pomocí vazby 1:N. Pokud v aplikaci nastane situace, že dva nebo více lidí bydlí na stejné adrese, nedojde k duplikaci dat – příslušný záznam s adresou je přiřazen více osobám.

**Tabulka 6 – osoby**

Klíč	Sloupec	Datový typ	Popis
PK	osoba_ID	Int(11)	ID osoby
	jmeno	Varchar(100)	Jméno
	prijmeni	Varchar(100)	Příjmení
FK	adresa_ID	Int(11)	ID adresy osoby

Kardinalita	Rodič	Potomek	Název vztahu
1:1	osoba	zamestnanec	zamestnanec_je_osoba
1:1	osoba	ridic	ridic_je_osoba
1:N	adresa	osoba	osoba_ma_adresu

Tabulka osoba je společná pro ukládání údajů o zaměstnancích a řidičích. Obě skupiny se liší svým přístupem k aplikaci – zaměstnanci mají přiděleny přihlašovací údaje a mohou se přihlásit do systému a pracovat s ním podle úrovně přístupových práv určených uživatelskou rolí. Řidiči ke své práci přístup do aplikace nepotřebují, a proto přihlašovací údaje ani uživatelskou roli nemají. Tabulka osoba obsahuje pouze jméno, příjmení a vazbu na adresu. Tyto údaje jsou pro obě skupiny společné.

**Tabulka 7 – zaměstnanci**

Klíč	Sloupec	Datový typ	Popis
PK	zamestnanec_ID	Int(11)	ID zaměstnance
	aktivni	Int(1)	Zaměstnanec je v aplikaci dostupný
	login_ID	Int(11)	ID loginu
	role_ID	Int(11)	ID role
	osoba_ID	Int(11)	ID osoby

Kardinalita	Rodič	Potomek	Název vztahu
1:1	login	zamestnanec	zamestnanec_ma_login
1:1	osoba	zamestnanec	zamestnanec_je_osoba
1:N	role	zamestnanec	zamestnanec_ma roli

V tabulce zaměstnanci jsou zaneseni všichni lidé, kteří pracují s aplikací. Tabulka obsahuje jako přímé údaje pouze ID zaměstnance a informaci o tom, zda je dotyčná osoba stále zaměstnancem společnosti (atribut aktivni = 1). V případě, že člověk již ve firmě nepracuje,

nebo se přesunul na jinou pracovní pozici bez přístupu k informačnímu systému (například práce řidiče), nastaví se hodnota na 0. Tímto lze snadno obejít problémy s databázovými vazbami při mazání. Uživatelská role a osobní údaje jsou k tabulce zaměstnanců připojeny formou cizího klíče, fyzicky jsou uloženy v tabulkách popsaných výše.

**Tabulka 8 – řidiči**

Klíč	Sloupec	Datový typ	Popis
PK	ridic_ID	Int(11)	ID řidiče
	aktivni	Int(1)	Řidič je v aplikaci dostupný
	počet_rozvozu	Int(11)	Počet rozvozu realizovaný řidičem
	počet_baliku	Int(11)	Počet zásilek doručených řidičem
FK	osoba_ID	Int(11)	ID osoby

Kardinalita	Rodič	Potomek	Název vztahu
1:N	ridic	rozvoz	rozvoz_realizovan_ridicem
1:1	osoba	ridic	ridic_je_osoba

Řidiči jsou v databázi uloženi odděleně od zaměstnanců pracujících se systémem. Z hlediska databáze mezi nimi není velký rozdíl, přesto je vhodné obě skupiny ukládat odděleně. Řidiči stejně jako zaměstnanci jsou navázáni na tabulku osoba a nepřímě též na tabulku adresa. Chybí ale vazba na uživatelskou roli a login, tyto údaje nejsou potřeba. Kromě sloupce aktivní, který funguje stejným způsobem, jako je popsáno u tabulky zaměstnanců, jsou přidány další dva atributy. Počet rozvozu a počet balíků se automaticky inkrementuje při úspěšném zakončení jízdy. Tyto dvě hodnoty vyjadřují počet odježděných dní a celkové množství doručených zásilek za celou dobu působení ve společnosti.

Vazba na tabulku rozvoz přiřazuje konkrétního řidiče k naplánovanému rozvozu.

**Tabulka 9 – rozvozy**

Klíč	Sloupec	Datový typ	Popis
PK	rozvoz_ID	Int(11)	ID rozvozu
	datum	Date	Datum rozvozu
	odjety_cas	Int(11)	Skutečný čas jízdy
	odjeta_delka_trasy	Double	Skutečná délka trasy
	predpoklad_delka_trasy	Double	Vypočítaná délka trasy
	predpoklad_cas_jizdy	Int(11)	Vypočítaný čas jízdy
FK	vozidlo_ID	Int(11)	ID vozidla
FK	ridic_ID	Int(11)	ID řidiče

Kardinalita	Rodič	Potomek	Název vztahu
1:N	řidič	rozvoz	rozvoz_realizovan_řidicem
1:N	vozidlo	rozvoz	rozvoz_realizovan_vozidlem
1:N	rozvoz	balík	balík_patri_do_rozvozu

Tabulka rozvozů spolu s tabulkou balíků představuje nejdůležitější část celé databáze. Parametry naplánovaných tras, které jsou počítány automaticky, se ukládají právě zde. Atribut datum určuje termín, kdy bude plánovaný rozvoz realizován, standardně to bývá následující pracovní den po provedení výpočtu. Předpokládaný čas jízdy a předpokládaná délka trasy jsou vypočítány během generování rozvozu. Vypočítaný čas jízdy nesmí přesáhnout 8 hodin (standardní pracovní doba) – trasa je naplánována tak, aby toto pravidlo bylo vždy splněno. Přesnost odhadu lze zpětně vyhodnotit porovnáním s hodnotami sloupců odjetý čas a odjetá délka trasy. Tyto údaje doplní operátor po dokončení rozvozu na základě zprávy od řidiče.

Rozvozy jsou připojeny k dalším třem tabulkám. Cizí klíče vozidlo ID a řidič ID směřují na příslušné řádky v tabulkách vozidlo a řidič. Algoritmus plánování vybere nejvhodnějšího řidiče a vozidlo, ty jsou pak připojeny ke konkrétnímu rozvozu. Jeden rozvoz má vždy přiřazeno právě jedno vozidlo a jednoho řidiče. Dále je vždy splněna podmínka, že rozvoz je vždy spojen aspoň s jedním záznamem z tabulky balíků (bude podrobněji popsáno dále).

**Tabulka 10 – balíky**

Klíč	Sloupec	Datový typ	Popis
PK	balík_ID	Int(11)	ID zásilky
	vaha	Int(3)	Hmotnost
	rozměry_s	Int(3)	Rozměry – šířka
	rozměry_v	Int(3)	Rozměry – výška
	rozměry_d	Int(3)	Rozměry – délka
	dobírka	Decimal(6,1)	Hodnota dobírky
	pojištění	Decimal(6,0)	Výše pojištění
	uložit	Int(1)	Uložit na depu
	expresní	Int(1)	Expresní zásilka
FK	adresa_odesílatel_ID	Int(11)	Adresa odesílatele
FK	adresa_adresat_ID	Int(11)	Adresa příjemce
FK	rozvoz_ID	Int(11)	ID rozvozu

Kardinalita	Rodič	Potomek	Název vztahu
1:1	adresa_balík	balík	odesílatel
1:1	adresa_balík	balík	adresat
1:N	rozvoz	balík	balík_patri_do_rozvozu



Další z hlavních tabulek. Tabulka balíky ve spojení s tabulkou adresa balík ukládá veškeré informace o přijatých zásilkách, které se budou doručovat. ID balíku je generováno automaticky pomocí funkce autoincrement, při každém vloženém balíku se zvýší o jednu. Hmotnost a rozměry vloží do databáze operátor při převzetí zásilky od podatele. Atribut dobírka je nepovinný, vyplní se pouze, pokud si odesílatel balíku přeje vybrat od adresáta stanovenou částku. Položky expresní a pojištění vyplní rovněž operátor na základě informací od odesílatele. Expresní balík je zařazen do rozvozu přednostně. Hodnota nula značí, že balík přednostní není, jednička znamená opak. Sloupec uložit může nabývat pouze hodnot 0 a 1. Implicitně je nastaveno 0, druhou možnost operátor nastaví v případě nedoručitelnosti zásilky.

Součástí každého balíku je adresa příjemce a odesílatele. Ty jsou zapsány v tabulce adresa balík, která je připojena dvojitou vazbou 1:1 – každá vazba pro jednu adresu. Obě adresy musí být při vkládání zásilky do databáze vyplněny. Vazba na tabulku rozvozů je typu 1:N. Hodnota cizího klíče rozvoz ID může být i NULL, to v případě, že balík zatím není zařazen do žádného rozvozu. Pokud balík součástí naplánovaného rozvozu je, klíč odkazuje na příslušný záznam v tabulce rozvozů.

**Tabulka 11 – adresy balíků**

Klíč	Sloupec	Datový typ	Popis
PK	adresa_ID	Int(11)	ID adresy
	jmeno	Varchar(200)	Jméno osoby / firmy
	ulice	Varchar(200)	Ulice a číslo popisné
	mesto	Varchar(200)	Město
	psc	Decimal(5,0)	PSČ
	latitude	Varchar(20)	Souřadnice GPS
	longitude	Varchar(20)	Souřadnice GPS

Kardinalita	Rodič	Potomek	Název vztahu
1:1	adresa_balik	balik	odesílatel
1:1	adresa_balik	balik	adresát

Tabulka adresa se skládá ze sloupců jméno, ulice, město, PSČ, které jednoznačně určují adresáta a odesílatele zásilky. Atributy latitude a longitude obsahují souřadnice GPS, které se při vložení balíku do systému automaticky získají ze zadané adresy prostřednictvím Google Geolocation API. Tyto souřadnice se zapisují pouze k adrese pro doručení zásilky. Vazby na tabulku balík jsou popsány v předchozím odstavci.

**Tabulka 12 – vozidla**

Klíč	Sloupec	Datový typ	Popis
PK	vozidlo_ID	Int(11)	ID vozidla
	aktivni	Int(1)	Vozidlo je v aplikaci dostupné
	typ	Varchar(50)	Typ (název) vozila
	spz	Char(8)	SPZ
	rok_vyroby	Decimal(4,0)	Rok výroby
	stav_tachometru	Int(11)	Stav tachometru
	VIN	Varchar(50)	Identifikační číslo vozidla
	nosnost	Int(4)	Nosnost nákladového prostoru
	objem_naklad_prostoru	Int(2)	Objem nákladového prostoru
FK	vyrobce_ID	Int(11)	ID výrobce
FK	servis_ID	Int(11)	ID servisu

Kardinalita	Rodič	Potomek	Název vztahu
1:1	rozvoz	vozidlo	rozvoz_realizovan_vozidlem
1:1	vozidlo_servis	vozidlo	vozidlo_ma_servis_prohlidky
1:N	vozidlo_vyrobce	vozidlo	vozidlo_ma_vyrobce

Součástí aplikace je i evidence vozidel. Veškeré informace o vozidlech jsou uloženy v tabulkách vozidlo, vozidlo výrobce a vozidlo servis. Tabulka vozidlo je z těchto tří největší a obsahuje nejvíce údajů. Podobně jako u tabulek řidičů a zaměstnanců, i zde je použit atribut aktivní, který umožňuje zvolené vozidlo deaktivovat. Tabulka uchovává parametry vozidel, se kterými pracuje algoritmus plánování rozvozu – jedná se o stav tachometru, nosnost a objem. Dále jsou zaznamenány i údaje sloužící pro identifikaci vozidla – výrobce (prostřednictvím vazby na tabulku výrobce vozidla), typové označení, SPZ, VIN a rok výroby. Servisní informace o vozidlech jsou uvedeny zapsány s separátní tabulce, která je připojena cizím klíčem servis ID.

**Tabulka 13 – výrobce vozidla**

Klíč	Sloupec	Datový typ	Popis
PK	vyrobce_ID	Int(11)	ID výrobce
	nazev	Varchar(50)	Jméno výrobce

Kardinalita	Rodič	Potomek	Název vztahu
1:N	vozidlo_vyrobce	vozidlo	vozidlo_ma_vyrobce

Jedinou funkcí tabulky výrobce vozidla je udržovat seznam výrobců. Vazba na tabulku vozidel je ve vztahu 1:N.

**Tabulka 14 – vozidlo servis**

Klíč	Sloupec	Datový typ	Popis
PK	servis_ID	Int(11)	ID servisní informace
	stk_platnost_do	Date	Platnost STK
	servisni_interval_km	Int(11)	Servisní interval
	posledni_servis_tach	Int(11)	Stav tachometru poslední s. k.
	vozidlo_je_zpusobile	Int(1)	Vozidlo je způsobilé k jízdě

Kardinalita	Rodič	Potomek	Název vztahu
1:1	vozidlo_servis	vozidlo	vozidlo_ma_servis_prohlidky

Do modulu evidence vozidel patří i údržba. Automobily vložené do systému musí mít vždy vyplněné datum platnosti STK a servisní interval pro pravidelné prohlídky. Tyto hodnoty jsou pravidelně kontrolovány a v případě blížícího se termínu STK nebo servisní prohlídky je operátor informován. Poslední atribut označuje způsobilost vozidla k provozu. V momentě, kdy vyprší platnost STK nebo je překročen servisní interval od poslední prohlídky, vozidlo je automaticky označeno jako nezpůsobilé (hodnota 0).

**Tabulka 15 – nastavení aplikace**

Klíč	Sloupec	Datový typ	Popis
	latitude	Varchar(20)	Souřadnice depa – zeměpisná š.
	longitude	Varchar(20)	Souřadnice depa – zeměpisná d.
	nazev_spolecnosti	Varchar(1000)	Název společnosti

Velmi jednoduchá tabulka, která neobsahuje ani primární klíč ani vazby na zbytek databáze. Je zde uložena pozice depa – hodnota se používá při výpočtu rozvozů. Název společnosti je uživatelsky nastavitelný, vložená řetězec se zobrazí v záhlaví aplikace.

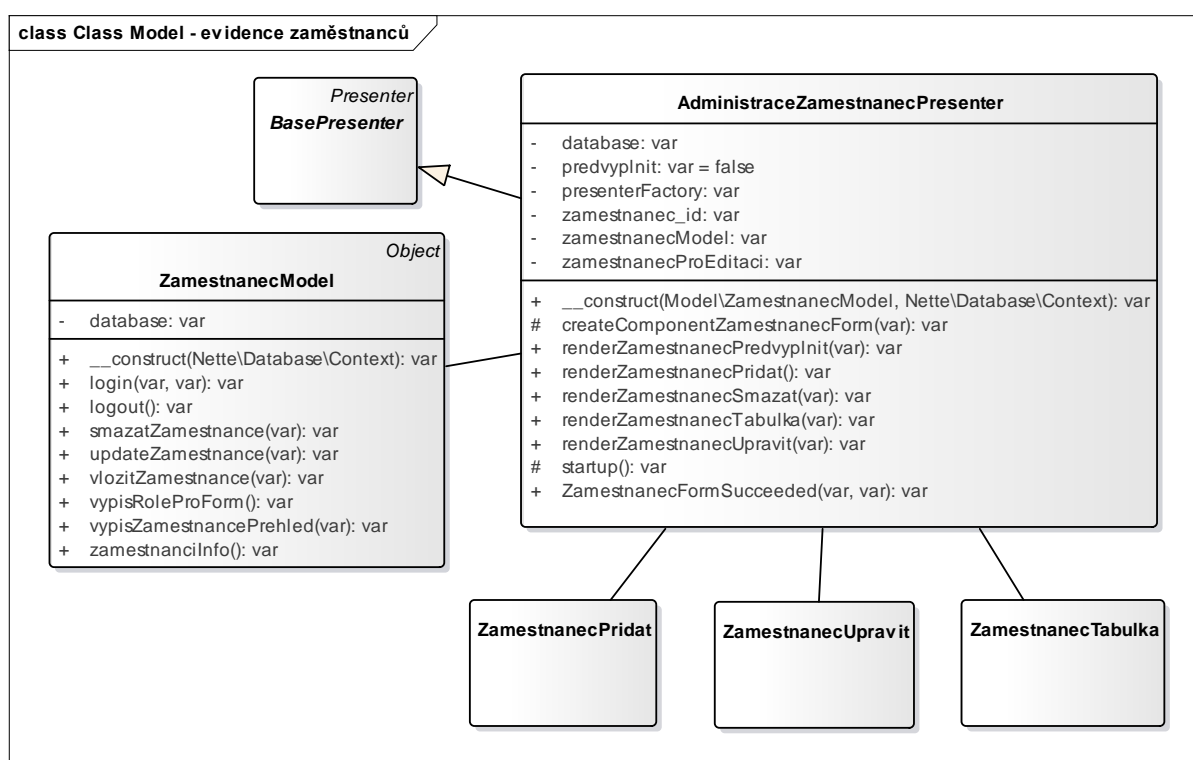
## 2.4 Moduly aplikace

Informační systém je rozdělen na moduly. Moduly jsou do jisté míry nezávislé, u některých modulů se liší i přístupová práva, která jsou stanovena podle uživatelské role právě přihlášeného uživatele. Celkový přehled implementovaných funkcí a jejich příslušnost k uživatelským rolím podrobně mapuje procesní BPMN diagram. Tento diagram je natolik rozsáhlý, že ho není možné vložit přímo do tištěné práce. Diagram ve formátech PDF, PNG a ADF je součástí přílohy.

### 2.4.1 Evidence zaměstnanců

Modul evidence zaměstnanců spravuje seznam všech zaměstnanců společnosti, kteří mají práva pro přístup do informačního systému. Jedná se o uživatele s rolí administrátor, operátor a manažer. Tento modul může obsluhovat pouze administrátor, ostatní k němu nemají přístup. Podporované funkce jsou:

- Vložení nového zaměstnance a přidělení uživatelské role.
- Úprava stávajících zaměstnanců.
- Odstranění zaměstnance ze systému (resp. deaktivace, z databáze není záznam fyzicky odstraněn).
- Zobrazení seznamu zaměstnanců v tabulce.



Obrázek 16 – UML Class diagram - evidence zaměstnanců

Z hlediska struktury zdrojového kódu se modul evidence zaměstnanců skládá celkem z pěti komponent – model, presenter a tři pohledy. Model kromě běžných funkcí pro práci se zaměstnanci (vypsát, přidat, upravit, smazat) obsluhuje i přihlašování do aplikace. Funkce `login` přijímá jako argumenty uživatelské jméno a heslo. V případě správně zadaných přihlašovacích údajů je identita zaměstnance uložena do session a tím je mu povolen vstup do částí aplikace, které odpovídají jeho roli. Funkce `logout` naopak tyto informace z paměti

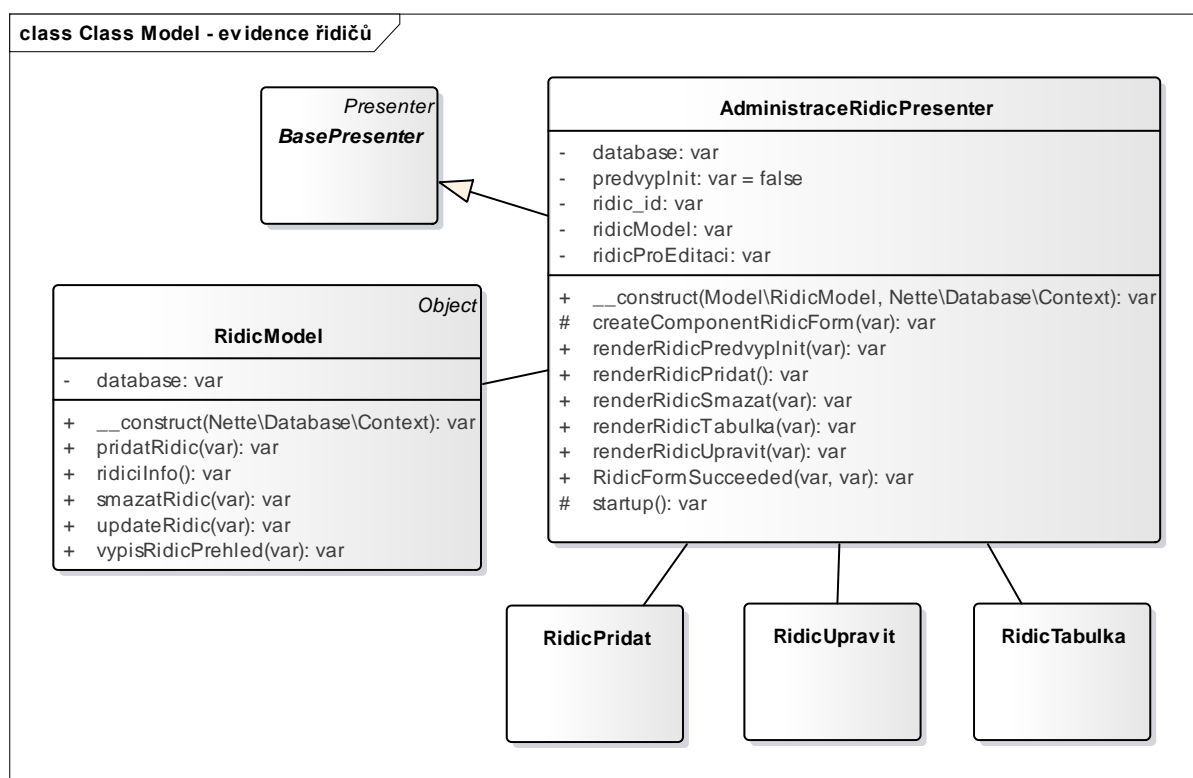
odstraní a uživatel nemůže dále pracovat. Odhlášení probíhá ručně nebo automaticky po uplynutí přednastaveného času.

## 2.4.2 Evidence řidičů

Velmi podobně jako předchozí funguje i druhý modul – evidence řidičů. Zde jsou uloženy informace o všech řidičích, kteří jsou zaměstnání v přepravní společnosti, nebo zde pracovali v minulosti. Řidič je identifikován pomocí jména a příjmení, další evidované údaje jsou adresa, počet doručených zásilek a množství realizovaných rozvozů. K evidenci řidičů mají přístup pracovníci s rolí operátor a administrátor.

Podporované funkce jsou:

- Vložení nového řidiče.
- Úprava stávajících řidičů.
- Odstranění řidiče ze systému (resp. deaktivace, z databáze není záznam fyzicky odstraněn).
- Zobrazení seznamu řidičů v tabulce.



Obrázek 17 – UML Class diagram - evidence řidičů

Modul evidence řidičů je složen z jednoho modelu, jednoho presenteru a tří pohledů. V modelu jsou implementovány funkce zajišťující komunikaci s databází – vypsát, vložit,

upravit a smazat zaměstnance. Presenter podle potřeby volá tyto funkce a pracuje s poskytnutými daty. Jednou z možností je data – obvykle ve formě matice – přeposlat do pohledu aktivovaného uživatelem. Tímto způsobem funguje pohled Řidič Tabulka – přijme pole se seznamem řidičů a vypíše ho do připravené tabulky (šablony). Ostatní dva pohledy zobrazují formulář pro vložení nebo editaci řidiče. V obou případech se jedná o stejný formulář, který se liší pouze předvyplněnými hodnotami. Pro editaci jsou do polí předeepsány původní hodnoty, formulář pro vložení nového řidiče je prázdný. Struktura formuláře je definována presenterem, pohled ho pouze zobrazí na zvoleném místě šablony.

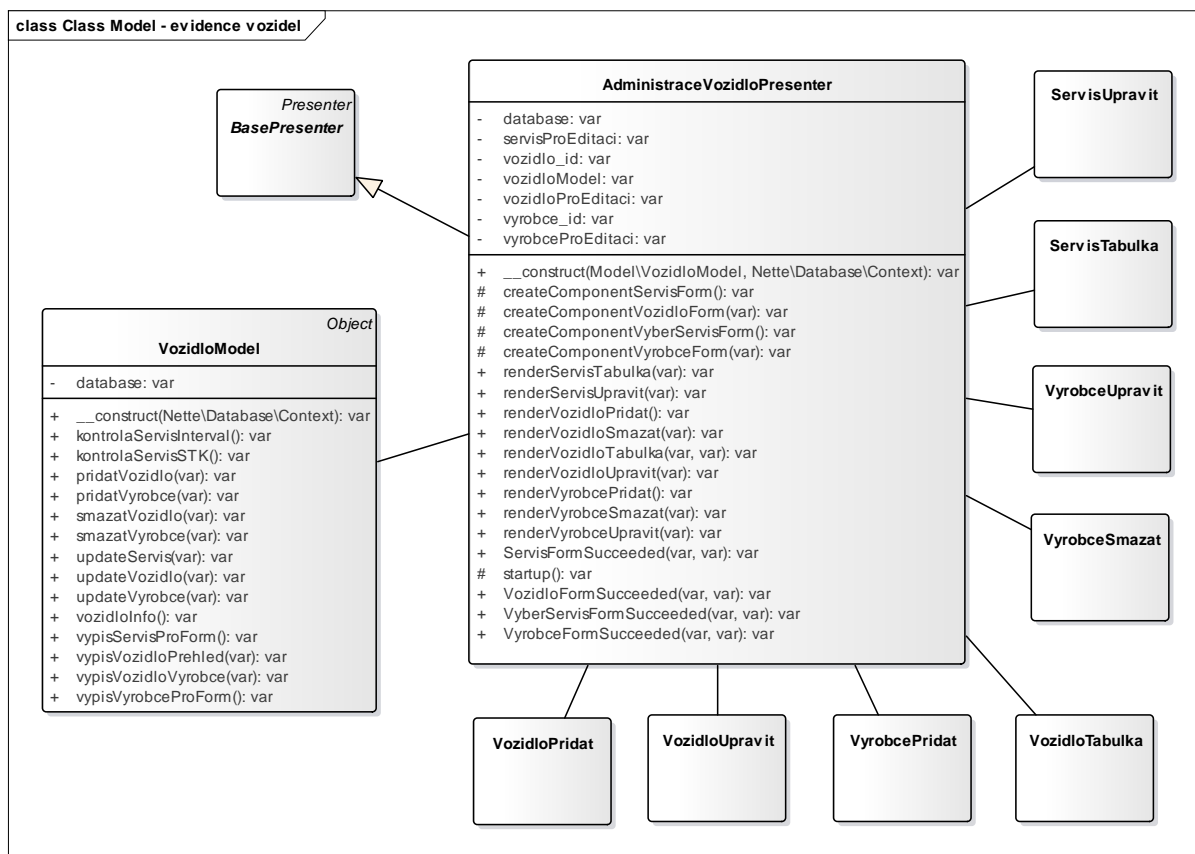
### **2.4.3 Evidence vozidel**

Detailní přehled o vozovém parku je pro společnost zabývající se přepravou zásilek velice důležitý. Součástí modulu evidence vozidel jsou základní informace o automobilu – výrobce, typové označení, SPZ, rok výroby a identifikační kód VIN. Pro správnou funkci dalších modulů informačního systému je nutné ukládat i další údaje – stav tachometru, nosnost vozidla a objem nákladového prostoru.

Součástí evidence vozidel je i řízení údržby vozidel. Každý automobil má stanoven servisní interval, po jehož uplynutí musí na servisní prohlídku. Evidován je i termín platnosti STK. Obojí je automaticky kontrolováno vždy při přihlášení uživatele, a pokud se blíží konec platnosti jednoho nebo druhého parametru, uživatel je na tuto skutečnost upozorněn. Upozornění se budou vypisovat, dokud nebude vykonána servisní kontrola / STK a termíny v databázi aktualizovány. Evidenci vozidel mohou obsluhovat pracovníci s rolí operátor nebo administrátor.

Podporované funkce jsou:

- Vložení nového vozidla.
- Úprava stávajících vozidel.
- Odstranění vozidla ze systému (resp. deaktivace, z databáze není záznam fyzicky odstraněno).
- Zobrazení seznamu vozidel v tabulce.
- Zobrazení a úprava servisních informací o vozidlech.
- Automatická kontrola platnosti STK a servisního intervalu.



Obrázek 18 – UML Class diagram - evidence vozidel

Výše uvedený UML Class diagram představující tuto část aplikace už je o něco složitější. Model i presenter obsahují velké množství funkcí a pohledů je celkem 8. Většina funkcí modelu se volá pouze v rámci bloku evidence vozidel. Výjimkou jsou dvě funkce pro kontrolu servisního intervalu a STK, které jsou spouštěny bezprostředně po přihlášení z Homepage Presenteru – tato vazba byla pro zjednodušení UML diagramu vynechána. Vozidlo Presenter má připojeno 8 pohledů, které zobrazují tři podčásti evidence vozidel – samostatnou tabulku výrobců vozidel, tabulku přehledu vozidel a tabulku servisních informací. Presenter pro všechny tři tabulky generuje formuláře pro editaci, v případě vozidel a výrobců se formuláře používají i pro přidávání nových záznamů. Servisní údaje jsou pevně spojené s příslušným vozidlem a nelze je mazat ani ručně přidávat – vždy vznikají (případně zanikají) současně s konkrétním vozidlem.

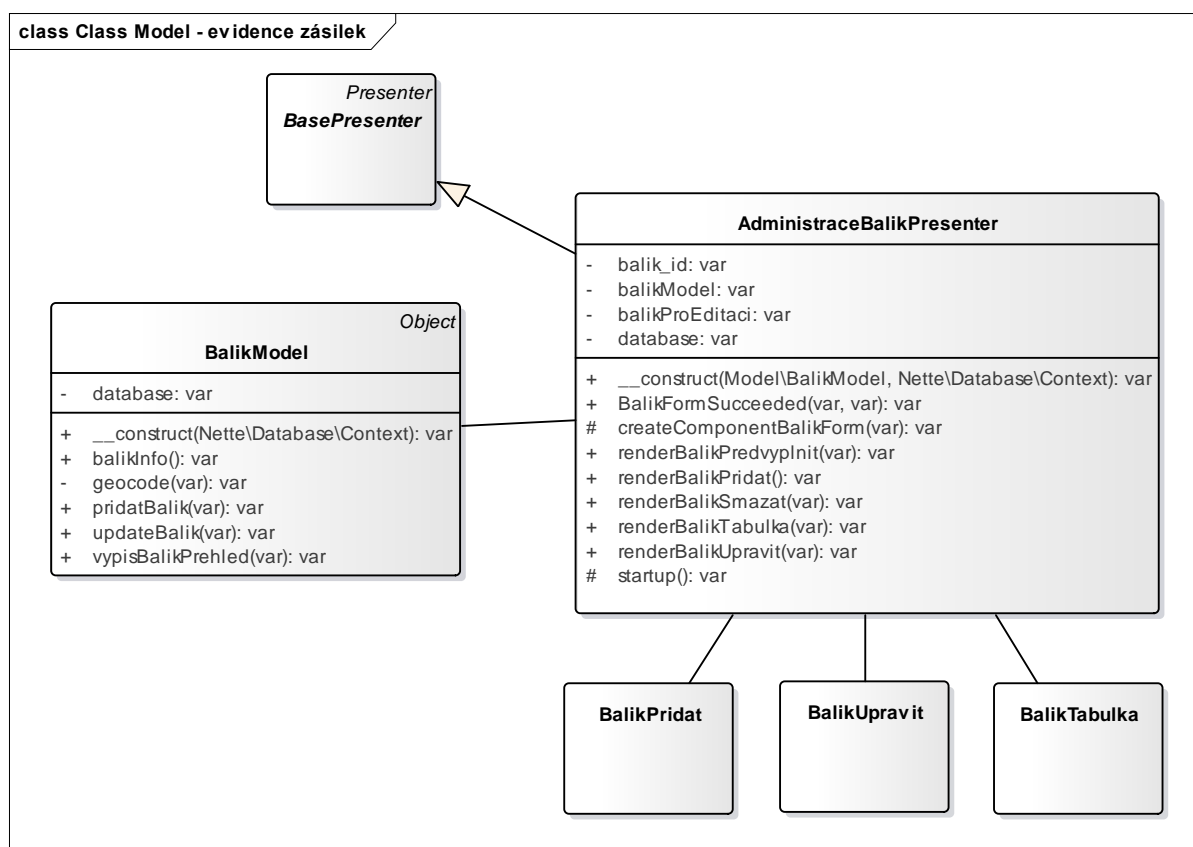
#### 2.4.4 Evidence zásilek

Jedním z klíčových modulů aplikace je evidence zásilek. Tato část aplikace umožňuje uživatelům vkládat zásilky do systému. Každý balík se vkládá ručně přes formulář. Vždy je nutné vyplnit adresu příjemce i odesílatele, rozměry balíku a jeho hmotnost. K dalším parametrům patří výše pojištění zásilky a případná hodnota dobírky. V případě zájmu

zákazníka lze balík označit jako expresní – ta bude doručována přednostně. V případě úspěšné validace vložených dat se automaticky převede adresa příjemce zásilky na souřadnice GPS, které se uloží do databáze spolu s parametry zásilky. Pro převod adresy se využívá služba Google Geolocation API. Aplikace povoluje přístup k modulu evidence zásilek nositelům role administrátor a operátor.

Podporované funkce jsou:

- Vložení nové zásilky.
- Úprava a mazání zásilek, které nejsou přiřazeny k žádnému rozvozu.
- Zobrazení seznamu zásilek v tabulce.



Obrázek 19 – UML Class diagram - evidence zásilek

UML Class diagram této části aplikace se podobně jako u dříve popisovaných modulu skládá z modelu, presenteru a tří pohledů. Implementovány jsou všechny standardní funkce pro manipulaci s daty.



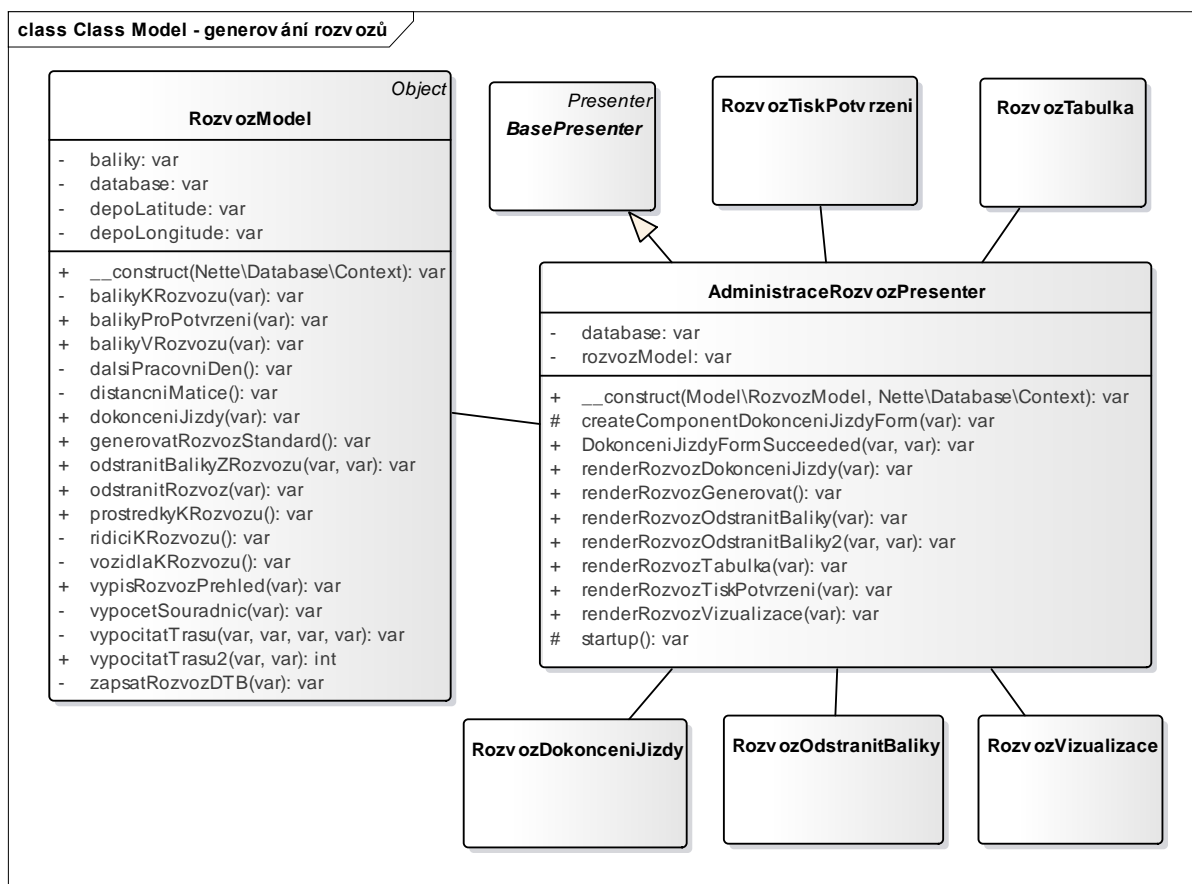
### 2.4.5 Generování rozvozů

Klíčovou komponentou celého systému je modul generování rozvozů. Trasy rozvozů se počítají automaticky po aktivaci příslušné funkce. Aplikace umožňuje generovat trasu pro následující pracovní den (svátky a víkendy jsou vynechány). Pro výpočet jsou důležité i dříve popsané moduly aplikace, pomocí kterých se do databáze vloží potřebná data. Algoritmus výpočtu pracuje se seznamem zásilek (dosud nedoručených), seznamem dostupných řidičů a provozuschopných vozidel. Cílem je minimalizovat náklady za rozvoz a zároveň dodržet přijatelnou rychlost doručování. Pro přednostní doručení lze zásilku označit jako expresní. Všechny expresní zásilky se do rozvozu vkládají jako první a doručují se odděleně od běžných balíků. Algoritmus generování tras bude podrobně popsán v kapitole 2.5. Kromě vytváření tras je lze rozvozy i mazat (v případě, že ještě nebyly vykonány) a upravovat. Připravenou trasu a adresy pro doručení zásilek je možné zobrazit v mapě.

Součástí modulu je i funkce pro tisk potvrzovacího archu. Ten obsahuje seznam všech zásilek, které má řidič rozvést včetně doplňujících informací. Zákazník při dodání balíku potvrdí převzetí. Na základě tohoto dokumentu se po dokončení jízdy vyřadí případné nedoručené zásilky a přes formulář dokončení jízdy operátor запиše stav tachometru vozidla a ujetý čas, čímž je rozvoz považován za vyřízený. Nedoručené balíky může operátor vyřadit z dokončeného rozvozu dvěma způsoby. První možnost je označit zásilku pro doručení následující den. Při dalším generování rozvozů se tyto zásilky automaticky zařadí do nově vytvořených tras. Druhá volba je uložení balíku na depu a případné vrácení odesílateli. Uložené balíky se do rozvozů nepřidávají. Stejně jako u dříve popsaných modulů, i zde má oprávnění k přístupu administrátor a operátor.

Podporované funkce jsou:

- Plánování nových rozvozů.
- Mazání rozvozů, které zatím nebyly dokončeny.
- Tisk potvrzovacího archu.
- Zobrazení naplánované trasy a adres pro doručení v mapě.
- Odebrání nedoručených zásilek a nastavení pro další rozvoz nebo uložení na depu.
- Dokončení rozvozu.



**Obrázek 20 – UML Class diagram - generování rozvozů**

Protože modul generování rozvozů je nejsložitější částí aplikace, i UML Class diagram obsahuje více funkcí a komponent. Základem je model, který implementuje všechny funkce používané algoritmem výpočtu trasy. Některé z nich jsou pouze pomocné pro získání různých dat a mezivýpočtů, proto jsou označeny jako private (soukromé). Presenter v závislosti na volbách uživatele v grafickém rozhraní pomocí příslušných funkcí „render“ volá veřejné funkce modelu. Ty provedou stanovené akce a v případě potřeby vrátí zpět data k zobrazení. Každá funkce „render“ obsluhuje svůj vlastní pohled, který je připraven pro vybraná data. Například pohled Rozvoz Vizualizace zobrazí pouze Google mapu s naplánovanou trasou a zvýrazněnými body reprezentujícími místa doručení zásilek.

## 2.4.6 Statistiky

Poslední částí aplikace určenou pro běžné používání je modul statistik. Standardně je sledování statistik určeno pouze pro vedení společnosti (role manažer), ostatní zaměstnanci sem nemají přístup. Je možné sledovat vytížení řidičů a vozidel, jsou zde informace i o doručených zásilkách.

První funkce umožňuje sledovat statistiku vybraného řidiče. Uživatel vybere ze seznamu řidiče a období, za které si přeje zobrazit statistiky. Období je ve výchozím stavu nastaveno na celou dobu působení řidiče ve společnosti, lze vybrat i kratší dobu – poslední týden, měsíc nebo rok. Po odsouhlasení parametrů výběru se zobrazí přehled dat ve formě tabulky a grafu. V tabulce jsou vypsány všechny uskutečněné rozvozy za vybrané období – datum rozvozu, délka trasy, čas jízdy, použité vozidlo a počet doručených zásilek v rámci rozvozu. Graf je sloupcového typu (horizontální pruhy) – na ose x je vynesena počet zásilek, osa y reprezentuje jednotlivé rozvozy.

Druhá funkce slouží k porovnávání řidičů mezi sebou. Opět se volí období ve stejných intervalech, jako je popsáno v předchozím odstavci. Tabulka a graf ukazují celkový počet balíků za zvolené období pro všechny zaměstnané řidiče. Pokud za zvolený časový úsek některý z řidičů nedoručil žádnou zásilku, není v grafu ani tabulce uveden.

Statistika vozidel funguje na stejném principu jako statistika řidičů. Vybírá se vozidlo a období pro zobrazení informací. Pro vozidla je vypsáno datum rozvozu, čas jízdy a najeté kilometry.

Porovnání vozidel zobrazí formou tabulky a grafu celkový čas strávený jízdou a celkovou vzdálenost ujetou za vybrané období pro všechny vozidla zanesené v systému. V grafu jsou vyneseny pouze najeté kilometry.

#### **2.4.7 Nastavení**

Nastavení je nejmenší modul celé aplikace. Slouží pro zapsání polohy depa (důležité pro algoritmus výpočtu trasy) a dalších pomocných informací. Přístupová práva pro modul nastavení má pouze administrátor.

### **2.5 Algoritmus generování rozvozů – implementace**

Jedním z hlavních bodů celé aplikace je řešení rozvozu zásilek s ohledem na minimalizaci nákladů a rovnoměrné vytížení řidičů a vozidel. Zároveň je cílem dodržet rozumnou dobu pro doručení zásilek. Základem pro úspěšné plánování rozvozů je nalezení pokud možno optimální (nejkratší) trasy, která spojuje místa, kam se mají balíky doručit. Tato část se nazývá „Úloha obchodního cestujícího“ a je popsána v teoretické části práce, včetně navrženého algoritmu pro řešení. To ale nestačí. Zásilky je nutné podle potřeby rozdělit na

více částí, přičemž každá z nich bude doručována jiným vozidlem na základě časových a technologických limitací.

### 2.5.1 Příprava pro výpočet

Nutným krokem pro zahájení tvorby rozvozů je získání všech dat, která algoritmus výpočtu potřebuje. Celý popsaný postup probíhá v cyklu, který trvá tak dlouho, dokud není vyčerpán jeden z prostředků. Pod pojmem prostředky se rozumí:

- Seznam řidičů, kteří jsou pro následující pracovní den dostupní. Seznam je seřazen vzestupně podle počtu doručených balíků. Princip výběru je podrobně popsán v kapitole 2.6.3.
- Seznam vozidel, které mohou v daném termínu jet. Uvažují se jen vozidla označená v systému jako aktivní a technicky způsobilá. Vozidla jsou seřazena vzestupně podle stavu tachometru.
- Seznam zásilek, které dosud nebyly doručeny, nejsou zařazeny do už naplánovaného rozvozu a nejsou označeny pro uložení na depu.

U zásilek se rozlišuje, jestli jsou určeny pro expresní nebo standardní rozvoz. Pokud jsou v databázi oba typy, vytvoří se dva separátní seznamy a pro každý z nich probíhá výpočet zvlášť. Pro další zpracování seznamu zásilek je nezbytné provést správné seřazení. Rozdělení zásilek pro jednotlivá vozidla probíhá na základě uhlového sektoru. Proto je potřeba převést souřadnice systému WGS-84 na souřadnice rovinné. S ohledem na ostatní zjednodušení je přesnost tohoto převodu dostatečná. Principem tohoto válcového zobrazení je zobrazit glóbus na plášť válce. V normální poloze jsou osy rotace shodné a dotykovou kružnicí je rovník. Pro oblast mimo rovník dochází k chybě, proto se dotyková kružnice se volí tak, aby tvořila osu zobrazovaného území. Rovnice pro přepočet tedy jsou:

$$x = R_z V \cos(U_0)$$

$$y = R_z U$$

$R_z$  značí poloměr Země,  $V$  je zeměpisná délka,  $U$  zeměpisná šířka,  $U_0$  je zeměpisná šířka středu pásu (počítá se s úhly v radiánech). Pro potřeby aplikace  $R_z = 6371$  km a  $U_0$  je zeměpisná šířka depa.

Z kartézských souřadnic se následně vypočítají ještě polární souřadnice, kde je poloha bodu pro doručení určena vzdáleností od počátku a úhlem. Za počátek je v tomto případě stanoveno depo. Seznam zásilek je seřazen vzestupně podle polárního úhlu. Nulový úhel leží vpravo od

depa a zvětšuje se proti směru hodinových ručiček. Nyní už jsou připravena všechna potřebná data a je možné spustit hlavní výpočet rozvozu. [34]

### 2.5.2 Hlavní algoritmus výpočtu

Hlavní výpočet běží v nekonečném cyklu, který je přerušen, pokud je vyčerpán některý ze tří výše uvedených zdrojů. Na začátku cyklu jsou vždy obnoveny všechny tři seznamy (zásilky, řidiči, vozidla), aby se v těle funkce vždy pracovalo s aktuálními daty. Uvnitř hlavního cyklu se nachází vedlejší cyklus, který prochází seřazené pole zásilek v pořadí určeném podle úhlu polárních souřadnic. V každé iteraci se přidá do trasy jeden uzel cesty, spočítá se celková vzdálenost (začíná v depu, prochází všemi adresami, končí v depu) a vyhodnocují se následující tři parametry:

- Časová náročnost trasy. Vypočítá se ze vzdálenosti – uvažována je průměrná rychlost 50 km/h a doba pro předání balíku činí 5 minut. Pokud čas jízdy přesáhne 4 hodiny nebo 8 hodin (pro expresní, resp. standardní rozvoz), cyklus končí.
- U zásilek se eviduje hmotnost a vozidlo má stanovenou maximální nosnost. Jestliže součet hmotností všech balíků přesáhne povolenou nosnost vozidla, cyklus končí.
- Zásilky mají specifikovány i rozměry (šířka, výška, hloubka). Zároveň je pro každé vozidlo stanoven objem nákladového prostoru. Problematika optimálního rovnání balíků do nákladového prostoru tak, aby se daný prostor co nejlépe využil, je velice rozsáhlá a komplikovaná. Řešení tohoto problému je v algoritmu generování rozvozu značně zjednodušené. U všech zásilek se uvažuje tvar kvádra o zadaných rozměrech. Objem zásilek zařazených do vozidla se sčítá a celková hodnota je vynásobena konstantou 1,3. Tím se kompenzuje nedokonalé uložení balíků v nákladovém prostoru. Pokud takto vypočítaný objem přesáhne kapacitu vozidla, cyklus je přerušen.

V momentě, kdy některý ze tří parametrů překročí povolenou mez, dojde k přerušení vnitřního cyklu. Ze zálohy se obnoví trasa, která obsahuje o jeden uzel méně. Tato cesta vyhovuje všem podmínkám a může být použita pro rozvoz. Vypočítaná trasa je v aplikaci uložena v poli – skládá se z ID řidiče, ID vozidla, celkové délky, vypočítaného času jízdy a seznamu zásilek, které do trasy náleží. Seznam zásilek je uložen v optimálním pořadí, jakým se mají projet, aby byla dodržena nejkratší vzdálenost cesty. Výsledná trasa se všemi údaji se zapíše do databáze a je možné pokračovat další iterací hlavního cyklu. Pokud seznam řidičů, vozidel nebo zásilek neobsahuje už žádné položky, není možné pokračovat ve výpočtu rozvozu a algoritmus je ukončen.

## 2.6 Zdrojové kódy vybraných částí aplikace

### 2.6.1 Přihlášení do systému

Pro správné zabezpečení informačního systému je nezbytné omezit přístup uživatelů k jeho jednotlivým částem. Tato aplikace nemá žádné veřejné uživatelské rozhraní, pro jakýkoliv přístup je potřeba mít platné přihlašovací údaje. Možnost pracovat s moduly systému je dále omezen podle uživatelské role.

Vždy po spuštění informačního systému je uživatel vyzván k zadání svého přístupového jména a hesla do formuláře. Funkce login (obrázek 21) využívá pro zlepšení bezpečnosti náhodně generovaný řetězec salt. Při registraci uživatele se vygeneruje vždy unikátní řetězec, který se připojí ke zvolenému heslu a pomocí kryptografické funkce SHA-256 se obojí dohromady převede na hash. Do databáze se uloží uživatelské jméno, salt a hash. Proces přihlášení funguje podobným způsobem. Z databáze se získá salt a hash odpovídající zadanému uživatelskému jménu, heslo napsané do formuláře v textové podobě se spojí se saltem a z výsledného spojení se spočítá hash. Ten se následně porovná s tím, který se vložil do databáze během registrace nového uživatele. V případě shody je přihlášení úspěšné a z databáze se přečtou informace o uživateli, které se následně vloží do superglobální proměnné `$_SESSION`.

```
protected function startup() {
    parent::startup();

    $login = isset($_SESSION['login']) && $_SESSION['login'] == 1;
    $role = isset($_SESSION['prava']) && ($_SESSION['prava'] == 1 || $_SESSION['prava'] == 2);

    if (!$login || !$role) {
        $this->flashMessage('Nedostatečná práva pro vstup do této sekce', 'error');
        $this->redirect(':Homepage:default');
    }
    $_SESSION['modul'] = " - Evidence zásilek";
}
```

Obrázek 21 – kontrola přihlášení a uživatelské role

Druhá fáze zabezpečení spočívá v důsledné kontrole přihlášení a přístupových práv ve všech částech aplikace. Každý presenter (z uživatelského hlediska modul aplikace) po své aktivaci automaticky spustí funkci `startup()`, která přečte ze `$_SESSION`, informace o přihlášeném uživateli. Pokud je vše v pořádku, je možné pokračovat ve spuštění kódu. V opačném případě je zobrazena chybová zpráva a přístup zamítnut.

Funkce odhlášení je velmi jednoduchá, z proměnné `$_SESSION` se smažou údaje o přihlášeném uživateli a další práce s informačním systémem není možná, dokud znovu neproběhne proces přihlášení.

```
public function login($username, $password) {

    $sql = "select salt "
        . " from "
        . " login where username=?";

    $result = $this->database->query($sql, $username);
    $salt = array();
    foreach ($result as $row) {
        $salt = $row->salt;
    }
    $pass = $password . $salt;
    $passHash = hash("sha256", $pass);

    $sql = "select zamestnanec_ID, osoba.jmeno as zamestnanec_jmeno, osoba.prijmeni as zamestnanec_prijmeni, "
        . " username, prava, role.nazev as role_nazev "
        . " from "
        . " zamestnanec join osoba using(osoba_ID) join role using(role_ID) "
        . " join login using(login_ID) where username=? and password=? and aktivni = 1 "
        . " order by zamestnanec_ID ";

    $result = $this->database->query($sql, $username, $passHash);

    $data = array();
    $i = 0;
    foreach ($result as $row) {
        $data[$i][0] = $row->zamestnanec_ID;
        $data[$i][1] = $row->zamestnanec_jmeno;
        $data[$i][2] = $row->zamestnanec_prijmeni;
        $data[$i][3] = $row->username;
        $data[$i][4] = $row->prava;
        $data[$i][5] = $row->role_nazev;
        $i++;
    }
    $data['size'] = $i;

    if ($i == 1) {
        $_SESSION['login'] = 1;
        $_SESSION['prava'] = $data[0][4];
        $_SESSION['role'] = $data[0][5];
        $_SESSION['uzivatel'] = $data[0][1] . " " . $data[0][2];
    }
}
```

Obrázek 22 – funkce login

## 2.6.2 Převod adresy na GPS souřadnice

Při vložení každé zásilky je nutné získat GPS souřadnice adresy pro doručení, které se použijí k dalším výpočtům. Vložená adresa se pošle jako argument funkce geocode. Přijatá adresa se skládá z názvu ulice a čísla popisného, města a PSČ. Tyto tři části jsou odděleny mezerami a posílány jako jeden řetězec. Prvním krokem funkce geocode je převod adresy na řetězec kompatibilní s formátem URL, k čemuž slouží PHP funkce urlencode. Získaná adresa se

připojí k odkazu na adresu webové služby Google Maps Geocoding API. Součástí odkazu je i unikátní klíč – ten slouží pro identifikaci uživatele posílajícího dotaz. Neplacená verze API povoluje 2500 převodů adresy za den. Geocoding API vrátí odpověď ve formátu JSON. V případě úspěšného zpracování adresy následuje dekodování odpovědi a uložení získaných hodnot do pole. Pole obsahující GPS souřadnice funkce geocode vrátí zpět do aplikace, kde se dále pracuje se získanými daty.

```
private function geocode($adresa) {  
  
    // prevod adresy na validni URL string  
    $adresa = urlencode($adresa);  
  
    // google map geocode api url  
    $url = "https://maps.googleapis.com/maps/api/geocode/"  
        . "json?key=AIzaSyCN2ajc_bXp3V5KK8-tUAR2-_rk8J8wn60&address={$adresa}";  
    $resp_json = file_get_contents($url);  
    $resp = json_decode($resp_json, true);  
  
    if ($resp['status'] == 'OK') {  
  
        // ziskani dat  
        $lati = $resp['results'][0]['geometry']['location']['lat'];  
        $longi = $resp['results'][0]['geometry']['location']['lng'];  
        $formatAdresa = $resp['results'][0]['formatted_address'];  
  
        // kontrola dat  
        if ($lati && $longi && $formatAdresa) {  
  
            $data = array();  
  
            array_push(  
                $data, $lati, $longi, $formatAdresa  
            );  
  
            return $data;  
        } else {  
            return false;  
        }  
    } else {  
        return false;  
    }  
}
```

Obrázek 23 – funkce geocode

### 2.6.3 Seznam řidičů pro rozvoz

Generování rozvozů je velmi rozsáhlá činnost vyžadující spolupráci mnoha dílčích funkcí. Jednou z nich je funkce pro získání seznamu řidičů, kteří mohou být přiděleni k rozvozu.



Hlavním bodem je správně sestavený SQL dotaz, který získá potřebná data, zbytek funkce je jen obslužný kód pro zpracování výstupu SQL dotazu.

Cílem dotazu je vypsat všechny řidiče, kteří splňují následující podmínky:

- Vypsat pouze aktivní řidiče.
- Vypsat řidiče, kteří nejsou pro plánovaný den využiti.
- Seřadit vzestupně podle počtu dosud rozvezených zásilek.

Pro zjednodušení databázové struktury nejsou vedeny zvlášť záznamy o rozvrhu řidičů. Veškeré informace je nutné získat z tabulky rozvoz. První část podmínky where specifikuje řidiče, kteří nemají naplánován rozvoz pro cílové datum. Zároveň je však nutné odfiltrovat ty řidiče, kteří již pro tento den mají naplánován rozvoz. Toho je dosaženo výčtem ID vloženým do podmínky „not in“. Bez této podmínky by v seznamu figurovali i řidiči, co už mají na vybraný den naplánovaný rozvoz, protože je velmi pravděpodobné, že už nějaký rozvoz vykonali v minulosti. Proto by samotná první podmínka nestačila. Zároveň je nutné do výsledku zahrnout i začínající řidiče s nulovým počtem doručených zásilek (a tudíž bez záznamu v tabulce rozvoz). To zajistí spojení typu left join.

```
private function ridiciKRozvozu() {  
  
    $sql = " select ridic_ID, pocet_rozvozu, pocet_baliku "  
        . " from "  
        . " ridic left join rozvoz using(ridic_ID) "  
        . " where (date(datum) != date ? or datum is null) and aktivni = 1 and ridic_ID not in "  
        . " (select ridic_ID from ridic join rozvoz using(ridic_ID) "  
        . " where date(datum) = date ? and aktivni=1 "  
        . " group by ridic_ID order by pocet_baliku) "  
        . " group by ridic_ID order by pocet_baliku ";  
  
    $result = $this->database->query($sql, $this->dalsiPracovniDen(), $this->dalsiPracovniDen());  
  
    $poleRidicu = array();  
    $i = 0;  
    foreach ($result as $row) {  
        $poleRidicu[$i][0] = $row->ridic_ID;  
        $poleRidicu[$i][1] = $row->pocet_rozvozu;  
        $poleRidicu[$i][2] = $row->pocet_baliku;  
  
        $i++;  
    }  
    $poleRidicu['size'] = $i;  
  
    return $poleRidicu;  
}
```

Obrázek 24 – výpis řidičů pro rozvoz

## 2.7 Instalační příručka

Pro úspěšnou instalaci aplikace je potřebné následující softwarové vybavení, které musí být nainstalováno na serveru nebo počítači, kde bude aplikace spuštěna.

- Apache web server.
- PHP verze 5.3.1 nebo vyšší.
- MySQL + phpMyAdmin.
- Připojení k Internetu.

### 2.7.1 Instalace informačního systému

Jako první krok je nutné se ujistit, že na cílovém stroji je nainstalováno výše zmíněné softwarové vybavení. Po nastartování webového serveru a internetového prohlížeče je možné pokračovat dalším krokem – vytvoření databáze, kde se budou ukládat data aplikace. Databáze se založí pomocí nástroje phpMyAdmin, funkce „Vytvořit databázi“. Název lze zvolit libovolný (později se nastaví i v konfiguračním souboru aplikace), kódování UTF-8. Nyní už nic nebrání vytvoření struktury tabulek a vložení počátečních dat. Na přiloženém CD se nachází SQL skript pro import databáze – soubor nazvaný doprava.sql. Ten se otevře pomocí funkce import v phpMyAdmin a zpracuje stisknutím tlačítka „Proved“. Všechny volby importu zůstávají na výchozích hodnotách.

V tuto chvíli je databáze připravena a je možné pokračovat další fází instalace. Všechny soubory a složky aplikace se musí nakopírovat na server do kořenového adresáře webového serveru. Totéž platí i v případě instalace na lokální počítač bez dedikovaného serveru. Pro správný běh aplikace je nutné zkopírovat celou adresářovou strukturu včetně souborů umístěných v kořenovém adresáři. Složky „log“ a „temp“ potřebují přístupová práva pro zápis, pro ostatní stačí povolit jen čtení.

Závěrečný bod instalace se zabývá nastavením přístupových údajů k databázi v konfiguračním souboru aplikace. Tento soubor se jmenuje „config.local.neon“ a nachází se ve složce app/config. Řádek označený dsn určuje typ databáze, adresu serveru a název databáze. Adresa serveru v naprosté většině případů může zůstat nastavena na výchozí hodnotu, změnit IP adresu by bylo zapotřebí jen, kdyby webový server s aplikací a databázový server nebyly nainstalovány na stejném PC. Položka „dbname“ specifikuje název databáze, ke které se informační systém připojí. Zde se vyplní název zvolený při založení

databáze. Řádky user a password slouží k zadání přihlašovacích údajů k databázi. Ostatní nastavení není nutné měnit.

```
database:
    dsn: 'mysql:host=127.0.0.1;dbname=dp'
    user: root
    password:
    options:
        lazy: yes
```

Obrázek 25 – konfigurační soubor databáze

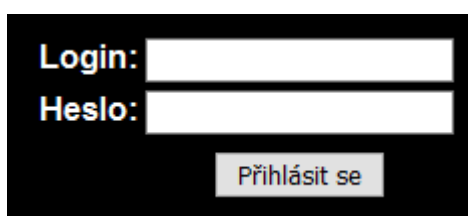
### 2.7.2 První spuštění a konfigurace

Po úspěšné instalaci by aplikace měla být funkční a připravená k práci. Po zadání adresy serveru do webového prohlížeče se zobrazí přihlašovací dialog. Čistá instalace má založen pouze účet administrátora (login: admin, heslo: admin123), všechny moduly aplikace jsou prázdné. Po přihlášení může administrátor s aplikací začít pracovat. Doporučený postup práce je změnit heslo administrátorského účtu a založit uživatelské účty pro zaměstnance. Ti potom mohou do systému zadat řidiče a vozidla společnosti a začít využívat funkce pro provoz přepravní společnosti.

## 2.8 Uživatelská příručka

### 2.8.1 Přihlášení do systému

Aplikace nemá žádné veřejně přístupné rozhraní, pro veškerou práci je nutné být přihlášen pod svým uživatelským účtem. Každý zaměstnanec dostane svůj účet přidělen od administrátora. Po spuštění aplikace se zobrazí přihlašovací formulář, do kterého zaměstnanec zadá svoje přihlašovací údaje.



The image shows a login form with a black background. It contains two white input fields. The first field is labeled 'Login:' and the second is labeled 'Heslo:'. Below these fields is a button with the text 'Přihlásit se'.

Obrázek 26 – přihlašovací formulář

### 2.8.2 Hlavní menu

V případě úspěšného přihlášení se zobrazí hlavní menu aplikace, které obsahuje funkce odpovídající roli přihlášeného uživatele. Menu je vždy generováno podle uživatelské role, což zamezuje neoprávněnému přístupu k funkcím nad rámec role. Zároveň každá z funkcí na svém začátku kontroluje přístupová práva uživatele a v případě jejich nesplnění nepovolí spuštění. K procesu přihlášení do aplikace je přiřazena funkce kontroly platnosti STK a servisních intervalů vozidel. Pokud se na některém vozidle blíží konec platnosti, při každém přihlášení je automaticky zobrazena zpráva o této skutečnosti. Po uplynutí zbývajících doby je vozidlo automaticky označeno jako nezpůsobilé a uživatel automaticky upozorněn.



Obrázek 27 – hlavní menu aplikace pro roli operátor

### 2.8.3 Evidence řidičů

Tato kapitola popisuje práci s modulem evidence řidičů. Velmi podobně funguje i modul evidence zaměstnanců, který není z menu na obrázku 27 dostupný, protože přístup k němu má pouze uživatel s právy administrátora. V evidenci řidičů jsou zaznamenány údaje o všech řidičích, kteří ve společnosti pracují, i o řidičích již nepracujících. Grafické rozhraní se skládá ze dvou tabulek – aktivní a neaktivní řidiči – a informačního panelu, který zobrazuje počet řidičů obsažených v obou skupinách. Řidiče ze systému nelze kompletně odstranit, pouze označit jako neaktivního. V tomto případě není daný řidič zařazen do plánování nových rozvozů. Neaktivní stav lze kdykoliv zrušit. Pomocí formuláře zobrazeného na obrázku 28 je možné upravit údaje řidiče, případně vložit nového – v tomto případě se formulář otevře prázdný. Tučně označené položky jsou povinné. Počet rozvozů se aktualizuje automaticky a za normálních okolností není nutné tuto hodnotu upravovat ručně. Po úspěšném vložení nového řidiče nebo aktualizaci stávajícího je operátor přesměrován zpět na tabulku řidičů a v záhlaví je zobrazena zpráva potvrzující provedenou akci.

## Upravit řidiče

<b>Jméno:</b>	<input type="text" value="Petr"/>
<b>Příjmení:</b>	<input type="text" value="Černý"/>
<b>Ulice + ČP:</b>	<input type="text" value="Na jezírkách 257"/>
<b>Město:</b>	<input type="text" value="Hradec Králové"/>
<b>PSČ:</b>	<input type="text" value="50011"/>
<b>Počet rozvozů:</b>	<input type="text" value="40"/>
<input type="button" value="Vložit / upravit"/>	

Obrázek 28 - formulář pro úpravu řidiče

### Informační systém přepravní společnosti

START
ZAMĚSTNANCI ▾
VOZIDLA ▾
ZÁSLIKY ▾

Eva Horáková (operátor)  
[Odhlásit](#)

### Evidence řidičů

Aktivních řidičů: 2  
 Neaktivních řidičů: 1

**Vložit nového řidiče**

Řidič ID	Jméno	Příjmení	Ulice	Město	PSČ	Počet rozvozů	
1	Petr	Černý	Na jezírkách 257	Hradec Králové	50011	40	<input type="button" value="Upravit"/> <input type="button" value="Smazat"/>
3	Anna	Pokorná	V Domkách 209	Hradec Králové	50004	0	<input type="button" value="Upravit"/> <input type="button" value="Smazat"/>

**Neaktivní**

Řidič ID	Jméno	Příjmení	Ulice	Město	PSČ	Počet rozvozů	
2	Pavel	Veselý	Ivana Olbrachta 928	Hradec Králové	50002	10	<input type="button" value="Obnovit"/>

Obrázek 29 – evidence řidičů

## 2.8.4 Evidence vozidel

Modul evidence vozidel spravuje vozový park firmy. Jsou zde uvedena všechna vozidla a jejich základní parametry. Důležitou součástí modulu představuje funkce servis. Zde je pro každé vozidlo vyplněn termín platnosti kontroly STK a zapsány jsou i údaje o servisních intervalech. Každý automobil má v databázi zapsán současný stav tachometru, předepsaný servisní interval a stav tachometru, kdy byl naposledy navštíven servis. Všechny tyto

informace se automaticky kontrolují a po překročení stanoveného termínu nebo stavu kilometrů je vozidlo přesunuto do sekce nezpůsobilých k jízdě. Nezpůsobilé automobily nejsou zařazeny do plánování rozvozu.

Informační systém přepravní společnosti								Eva Horáková (operátor)
								<a href="#">Odhlásit</a>
START	ZAMĚŠTNANCI	VOZIDLA	ZÁSKLKY					
Evidence vozidel - servis								Způsobilých vozidel: 2
								Nezpůsobilých vozidel: 1
Servis ID	Vozidlo	SPZ	STK platnost do	Najeto km	Servisní interval	Poslední servis na	Způsobilé	
1	Mercedes Sprinter	4H5 3900	07.09.2016	48400	16000	40000	ano	<a href="#">Upravit</a>
2	Ford Transit	2H9 3246	19.08.2016	48330	12000	45000	ano	<a href="#">Upravit</a>
Nezpůsobilé k jízdě								
Servis ID	Vozidlo	SPZ	STK platnost do	Najeto km	Servisní interval	Poslední servis na	Způsobilé	
3	Mercedes Sprinter	6H5 7861	18.05.2016	20000	30000	0	ne	<a href="#">Upravit</a>

Obrázek 30 – servis vozového parku

Servisní informace se upravují pro každé vozidlo zvlášť, slouží k tomu následující formulář. Datum platnosti STK se vybírá pomocí interaktivního kalendáře.

## Upravit servis

STK:	18.05.2016
Servisní interval:	
Poslední servis [tach]:	

May 2016

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

Obrázek 31 – formulář pro nastavení servisních údajů

## 2.8.5 Evidence zásilek a generování rozvozů

V tomto modulu jsou ukládány zásilky a trasy pro rozvoz. Zásilka má vždy uvedenu cílovou adresu (pro doručení) a také zpáteční adresu. Pro úspěšné sestavení rozvozu je nezbytné zadat správně i hmotnost a rozměry balíku. Tyto hodnoty jsou následně použity pro výpočet. Hlavní obrazovka je rozdělena na tři části:

- Tabulka aktivních zásilek, které zatím nebyly zpracovány a čekají na zařazení do rozvozu.
- Tabulka zásilek, které byly po neúspěšném pokusu o doručení uloženy na depu.
- Tabulka doručených zásilek.

Informační systém přepravní společnosti

Eva Horáková (operátor)  
[Odhlásit](#)

STARTZAMĚSTNANCI VOZIDLAZÁSILKY

Evidence zásilek

Aktivních zásilek: 0  
Uložených zásilek: 1  
Doručených zásilek: 5

Aktuální  
Vložit balík

Balík ID	Hmotnost	Rozměry ŠxVxH	Dobírka	Pojištění	Adresát	Expresní
----------	----------	---------------	---------	-----------	---------	----------

Uložené na depu

Balík ID	Hmotnost	Rozměry ŠxVxH	Dobírka	Pojištění	Adresát				Expresní		
7	10 kg	70x30x50 cm	3500 Kč	5000 Kč	Josef Švejk	Smetanova 165	Chlumec nad Cidlinou	50351	50.1574565 15.4576112	ne	<a href="#">Upravit</a>

Doručené

Balík ID	Hmotnost	Rozměry ŠxVxH	Dobírka	Pojištění	Adresát				Expresní	
4	4 kg	50x40x20 cm	-	5000 Kč	Jiří Marek	Záchlumí 83	Záchlumí	56186	50.0962482 16.3753341	ano
5	1 kg	50x40x20 cm	-	6000 Kč	Hana Kučerová	Palackého 97	Smířice	50303	50.3001142 15.8661287	ne
6	2 kg	25x10x32 cm	-	2000 Kč	Michal Černý	Lesní 166	Opatovice nad Labem	53345	50.1425553 15.7864633	ne
8	20 kg	100x40x50 cm	-	10000 Kč	Marie Novotná	Zahradní 22	Čáslav	28601	49.9062678 15.3879909	ano
9	12 kg	70x30x32 cm	-	10000 Kč	Jaroslav Svoboda	Husova 76	Jičín	506	50.4366155 15.356369	ne

Obrázek 32 – zásilky rozdělené podle stavu

Funkce generování rozvozů navazuje na evidenci zásilek a umožňuje plánovat jejich doručení. Z dostupných zásilek se po stisku tlačítka „Vygenerovat rozvozy“ automaticky vypočítají trasy, po kterých řidiči zásilky doručují. Algoritmus výpočtu bere ohled na rovnoměrné vytížení řidičů a vozidel, současně je snaha minimalizovat náklady. Dále jsou zásilky děleny do více vozidel podle doby jízdy, hmotnosti a objemu zásilek. Vypočítané rozvozy se zobrazí v tabulce (obrázek 33). Tabulka informuje o vybraném vozidle, řidiči a

předpokládané délce trasy včetně časové náročnosti. Pro každý rozvoz operátor vytiskne potvrzovací arch, který následně předá řidiči. Potvrzovací arch je seřazen vzestupně podle optimálního pořadí, ve kterém je vhodné zásilky doručovat. Adresát při převzetí potvrdí, že zásilku přijal. Na základě tohoto seznamu, který řidič po návratu do depa předá operátorovi, je možné dokončit rozvoz. Nedoručené zásilky operátor odebere pomocí funkce „Odebrat balíky z rozvozu“ a označí je buď jako připravené pro rozvoz následující den nebo pro uložení na depu. Pokud seznam zásilek odpovídá skutečnému stavu, operátor запиše do systému nový stav tachometru vozidla a čas jízdy (oba údaje po skončení jízdy řidič napíše na potvrzovací arch), čímž je rozvoz označen za dokončený.

Informační systém přepravní společnosti

Eva Horáková (operátor)

Odhlásit

START

ZAMĚSTNANCI

VOZIDLA

ZÁSILKY

Evidence rozvozů

Aktuální

Standardních zásilek k doručení: 0

Expresních zásilek k doručení: 0

Volných řidičů: 0

Volných vozidel: 0

Vygenerovat rozvozy

Rozvoz ID	Datum	Délka trasy	Čas jízdy	Vozidlo	Řidič	Počet balíků					
144	25.04.2016	159.8 km	197 min	Ford Transit (2H9 3246)	Anna Pokorná	2	Tisk potvrzovacího archu	Dokončení jízdy	Zobrazení v mapě	Odebrat balíky z rozvozu	Zrušit rozvoz
145	25.04.2016	102.6 km	128 min	Mercedes Sprinter (4H5 3900)	Petr Černý	3	Tisk potvrzovacího archu	Dokončení jízdy	Zobrazení v mapě	Odebrat balíky z rozvozu	Zrušit rozvoz

Dokončené

Rozvoz ID	Datum	Délka trasy - plán / skutečná	Čas jízdy - plán / skutečný	Vozidlo	Řidič	Počet balíků
-----------	-------	-------------------------------	-----------------------------	---------	-------	--------------

Obrázek 33 – seznam rozvozu

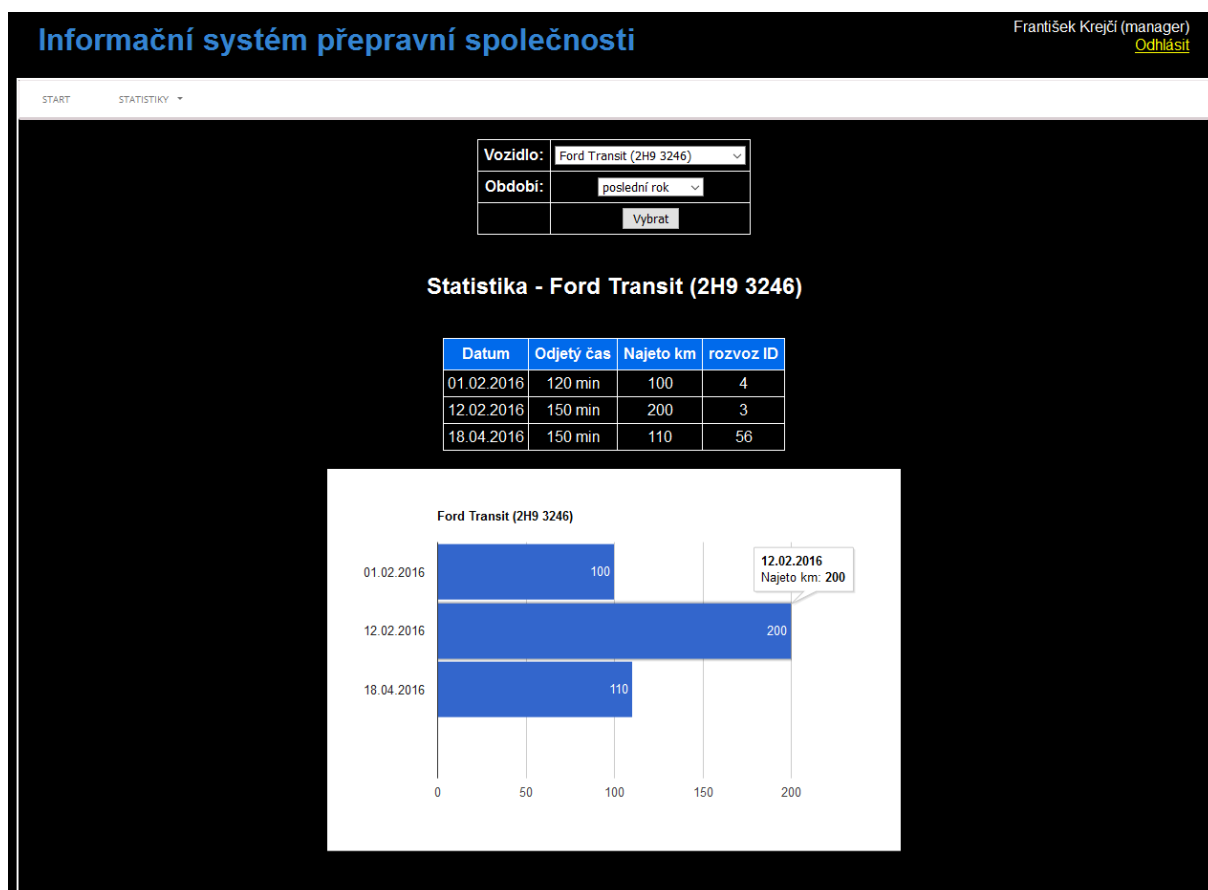
## 2.8.6 Modul statistik

Tato část aplikace je přístupná pouze pro roli manažera. Ve formě tabulek a grafů je možné prohlížet data popisující činnost firmy. Menu obsahuje čtyři volby.

- Statistika řidičů – přes formulář se vybere požadovaný řidič a období pro zobrazení dat (poslední týden, měsíc, rok nebo bez omezení). V závislosti na zvolených hodnotách se zobrazí tabulka a graf obsahující požadované informace.



- Porovnání řidičů – vybírá se období pro zobrazení dat. V tabulce a grafu má každý řidič uveden počet doručených zásilek za zvolené období.
- Statistika vozidel – formou grafu a tabulky vypíše rozvozy (čas a vzdálenost) realizované na konkrétním vozidle za určenou dobu.
- Porovnání vozidel – zobrazí celkový čas strávený jízdou a najeté kilometry vozidel, která byla používána ve vybraném období.



Obrázek 34 – modul statistik – vozidlo

## ZÁVĚR

Cílem této práce bylo navrhnout a implementovat funkční aplikaci pro zajištění činnosti přepravní společnosti. V úvodní části práce byla nastíněna problematika elektronických informačních systémů, jejich výhody a specifika. Následně byly stanoveny základní požadavky na informační systém pro provoz přepravní společnosti – evidence vozidel, řidičů, zásilek. Pro hlubší prozkoumání problému byla vypracována rešerše porovnávající několik existujících informačních systémů se zaměřením na dopravu a spedici. Na základě zjištěných detailů o těchto aplikacích byl upřesněn návrh vlastního systému o další funkce a stanoveno zaměření na rozvoz zásilek po blízkém a středně vzdáleném okolí, řádově jednotky až desítky kilometrů.

Po naplánování určení a funkcí nově navrhované aplikace bylo nutné vybrat technologie a celkovou koncepci pro implementaci. Pro vlastní proces implementace byla vybrána forma webové aplikace založená na programovacím jazyce PHP s využitím Nette frameworku a databáze MySQL. Výhoda webové aplikace spočívá ve snadném provozu na více klientských stanicích – hlavní logika aplikace je nainstalována na serveru. Zároveň je možný i lokální provoz na běžném PC bez nutnosti použít server, což může být výhodné pro menší firmy. Nette framework je založený na architektuře MVP, což umožňuje při návrhu udržet dobrou přehlednost kódu i při tvorbě aplikace většího rozsahu. Nette současně poskytuje mnoho funkcí, které usnadňují tvorbu aplikace a zajišťují vysokou úroveň zabezpečení.

Teoretická část práce pojednává o technologiích použitých pro tvorbu vlastní aplikace, potažmo webových aplikací obecně. Jsou zde popsány použité programovací jazyky, způsoby použití, pravidla syntaxe a krátké ukázky zdrojových kódů naznačující základní možnosti jazyka. V další kapitole je probrána problematika zabezpečení webových aplikací – časté typy útoků a jak se proti nim bránit. V závěrečné kapitole teoretické části je přiblížen problém hledání optimální cesty pro rozvoz zásilek – úloha obchodního cestujícího. Na základě požadavků informačního systému byl vytvořen vlastní algoritmus hledání pokud možno nejkratší cesty mezi jednotlivými místy pro doručení zásilek. Tento algoritmus nedokáže vždy poskytnout optimální řešení, ale výsledná trasa je vhodná pro praktické použití a výpočetní náročnost není vysoká.

Praktická část práce pojednává o aplikaci vytvořené na základě návrhu stanoveného teoretickou částí. Jsou zde popsány použité algoritmy, struktura databáze a některé vybrané části zdrojového kódu. Prostor je věnován i pohledu na logickou strukturu aplikace, jak

z pohledu implementace, tak i z pohledu uživatele. Závěrečné dvě kapitoly slouží k popisu instalace a základnímu používání vytvořené aplikace.

Vyvinutá aplikace byla navržena s důrazem na snadnou použitelnost a přehledné grafické rozhraní. Podporovány jsou všechny základní funkce, které přepravní společnost potřebuje pro svoji činnost. Patří mezi ně evidence řidičů, vozidel a sledování jejich vytížení ve formě tabulek a grafů. Zvýšená pozornost byla věnována práci se zásilkami – jedná se o nejsložitější modul aplikace. Vložené zásilky jsou s využitím vlastního algoritmu vyhledání nejkratší trasy automaticky skládány do naplánovaných rozvozů. Funkce tvorby rozvozů je implementována s cílem minimalizovat náklady, ale zároveň nabídnout přijatelnou rychlost doručování.

Aplikace v současném stavu splňuje všechny náležitosti pro nasazení do reálného provozu včetně systému zabezpečení, který zabraňuje neoprávněné manipulaci s daty. Díky modulárnímu návrhu je zde prostor pro rozšíření a vylepšení funkčnosti v budoucnu. Jedním z možných rozšíření by mohla být spolupráce aplikace s mobilním zařízením, které by mělo přístup k informacím o konkrétním rozvozu, umožňovalo by zobrazit naplánovanou trasu a podle potřeby dynamicky měnit stav zásilek v systému v závislosti na úspěšnosti doručení. Celá operace dokončení rozvozu by mohla proběhnout automaticky, což by vedlo ke zvýšení efektivity práce a úsporám nákladů – odpadla by nutnost pro každý rozvoz tisknout potvrzovací arch.

### 3 POUŽITÁ LITERATURA

- [1] *Dopravní a spediční informační systém SPEiS* [online]. 2016 [cit. 2016-02-28]. Dostupné z: <http://www.speis.eu/index.php?id=0>
- [2] *Rinkai Routing* [online]. 2015 [cit. 2016-02-28]. Dostupné z: <http://www.rinkairouting.cz/>
- [3] *QI – Doprava a spedice* [online]. 2012 [cit. 2016-02-28]. Dostupné z: <http://www.qi.cz/moduly/specializovane-moduly/doprava-a-spedice/>
- [4] *Informační systémy pro dopravu, spedici, logistiku a autoservisy - Doprava 3K* [online]. 2016 [cit. 2016-03-03]. Dostupné z: <http://www.ksh-data.cz/produkty/doprava-3k/>
- [5] *Study CCNA – HTTP & HTTPS* [online]. 2016 [cit. 2016-03-03]. Dostupné z: <http://study-ccna.com/http-https/>
- [6] *List of Common HTTP Status Codes* [online]. 2016 [cit. 2016-03-08]. Dostupné z: <http://www.smartlabsoftware.com/ref/http-status-codes.htm>
- [7] *HTTP - Requests* [online]. 2016 [cit. 2016-03-09]. Dostupné z: [http://www.tutorialspoint.com/http/http\\_requests.htm](http://www.tutorialspoint.com/http/http_requests.htm)
- [8] KOSEK, Jiří. *Aplikace na Webu: 5. Základy protokolu HTTP* [online]. 1999 [cit. 2016-03-9]. Dostupné z: <http://www.kosek.cz/clanky/iweb/05.html>
- [9] *HTML Version Statistics* [online]. 2016 [cit. 2016-03-11]. Dostupné z: <http://try.powermapper.com/Stats/HtmlVersions>
- [10] *HTML Input Attributes: W3Schools.com* [online]. 2016 [cit. 2016-03-11]. Dostupné z: [http://www.w3schools.com/html/html\\_form\\_attributes.asp](http://www.w3schools.com/html/html_form_attributes.asp)
- [11] *HTML5 Video: W3Schools.com* [online]. 2016 [cit. 2016-03-11]. Dostupné z: [http://www.w3schools.com/html/html5\\_video.asp](http://www.w3schools.com/html/html5_video.asp)
- [12] JANOVSKEÝ, Dušan. *Verze HTML a XHTML: Jak psát web* [online]. 2010 [cit. 2016-03-11]. Dostupné z: <http://www.jakpsatweb.cz/html/verze-html.html>
- [13] *HTML5 Semantic Elements: W3Schools.com* [online]. 2016 [cit. 2016-03-11]. Dostupné z: [http://www.w3schools.com/html/html5\\_semantic\\_elements.asp](http://www.w3schools.com/html/html5_semantic_elements.asp)
- [14] *HyperText Markup Language: Wikipedie* [online]. 2016 [cit. 2016-03-11]. Dostupné z: [https://cs.wikipedia.org/wiki/HyperText\\_Markup\\_Language](https://cs.wikipedia.org/wiki/HyperText_Markup_Language)
- [15] *CSS Syntax and Selectors: w3schools.com* [online]. 2016 [cit. 2016-03-15]. Dostupné z: [http://www.w3schools.com/css/css\\_syntax.asp](http://www.w3schools.com/css/css_syntax.asp)
- [16] *Úvod do CSS: Web tvorba* [online]. 2004 [cit. 2016-03-15]. Dostupné z: <http://www.webtvorba.cz/css/uvod-do-css.html>
- [17] JANOVSKEÝ, Dušan. *Úvod do JavaScriptu: Jak psát web* [online]. 2004 [cit. 2016-03-18]. Dostupné z: <http://www.jakpsatweb.cz/javascript/javascript-uvod.html>

- [18] *JavaScript: Wikipedia, the free encyclopedia* [online]. 2016 [cit. 2016-03-18]. Dostupné z: <https://en.wikipedia.org/wiki/JavaScript>
- [19] *PHP 5 Global Variables - Superglobals: w3schools.com* [online]. 2016 [cit. 2016-03-20]. Dostupné z: [http://www.w3schools.com/php/php\\_superglobals.asp](http://www.w3schools.com/php/php_superglobals.asp)
- [20] *PHP: Visibility - Manual* [online]. 2016 [cit. 2016-03-20]. Dostupné z: <http://php.net/manual/en/language.oop5.visibility.php>
- [21] *PHP: Wikipedia, the free encyclopedia* [online]. 2016 [cit. 2016-03-20]. Dostupné z: <https://en.wikipedia.org/wiki/PHP>
- [22] BERNARD, Borek. *Úvod do architektury MVC: Zdroják* [online]. 2009 [cit. 2016-03-21]. Dostupné z: <https://www.zdrojak.cz/clanky/uvod-do-architektury-mvc/>
- [23] ČÁPKA, David. *MVC architektura: ITnetwork.cz* [online]. 2016 [cit. 2016-03-21]. Dostupné z: <http://www.itnetwork.cz/navrhove-vzory/mvc-architektura-navrhovy-vzor/>
- [24] *Rychlý a pohodlný vývoj webových aplikací v PHP | Nette Framework* [online]. 2016 [cit. 2016-03-23]. Dostupné z: <https://nette.org/cs/>
- [25] *Description of the Secure Sockets Layer (SSL) Handshake* [online]. 2008 [cit. 2016-04-05]. Dostupné z: <https://support.microsoft.com/en-us/kb/257591>
- [26] KOSEK, Jiří. *Bezpečnost webových aplikací: 4IZ228 – tvorba webových stránek a aplikací* [online]. 2014 [cit. 2016-04-05]. Dostupné z: <http://www.kosek.cz/vyuka/4iz228/prednasky/bezpecnost.pdf>
- [27] *SQL Injection: w3schools.com* [online]. 2016 [cit. 2016-04-07]. Dostupné z: [http://www.w3schools.com/sql/sql\\_injection.asp](http://www.w3schools.com/sql/sql_injection.asp)
- [28] *Zabezpečení webových aplikací I. - klientské skriptovací jazyky* [online]. 2007 [cit. 2016-04-08]. Dostupné z: <http://access.feld.cvut.cz/view.php?cislocanku=2007090001>
- [29] *Behind the Scenes of OptiMap* [online]. 2016 [cit. 2016-04-20]. Dostupné z: <http://gebweb.net/blogpost/2007/07/05/behind-the-scenes-of-optimap/>
- [30] *Problém obchodního cestujícího: Algoritmy.net* [online]. 2015 [cit. 2016-04-21]. Dostupné z: <https://www.algoritmy.net/article/5407/Obchodni-cestujici>
- [31] DÍTĚTOVÁ, Tereza. *Testování heuristik pro úlohy obchodního cestujícího* [online]. Praha, 2012 [cit. 2016-04-22]. Dostupné z: <http://www.vse.cz/vskp/eid/28403>. Bakalářské práce. Vysoká škola ekonomická v Praze, Katedra ekonometrie. Vedoucí práce Josef Jablonský
- [32] HRDINA, Zdeněk. *Transformace souřadnic ze systému WGS-84 do systému S-JTSK* [online]. Praha, 1997 [cit. 2016-04-23]. Dostupné z: [http://www.geospeleos.com/Mapovani/WGS84toSJTSK/WGS\\_JTSK.pdf](http://www.geospeleos.com/Mapovani/WGS84toSJTSK/WGS_JTSK.pdf). České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra radioelektroniky.
- [33] STEINER, Ivo a Jiří ČERNÝ. *GPS od A do Z*. 4. aktualizované vydání. Praha: eNav, s. r. o., 2006. ISBN 80-239-7516-1.

- [34] *Válcová zobrazení jednoduchá* [online]. 2016 [cit. 2016-04-25]. Dostupné z:  
[http://geomatika.kma.zcu.cz/studium/mk2/multimedialni\\_texty/index\\_soubory/hlavni\\_soubory/jednoduch\\_soubory/valec.html](http://geomatika.kma.zcu.cz/studium/mk2/multimedialni_texty/index_soubory/hlavni_soubory/jednoduch_soubory/valec.html)
- [35] *World Wide Web Timeline* [online]. 2014 [cit. 2016-04-26]. Dostupné z:  
<http://www.pewinternet.org/2014/03/11/world-wide-web-timeline/>
- [36] *PHP MySQL Database: w3schools.com* [online]. 2016 [cit. 2016-04-26]. Dostupné z:  
[http://www.w3schools.com/php/php\\_mysql\\_intro.asp](http://www.w3schools.com/php/php_mysql_intro.asp)
- [37] STEINER, Ivo a Jiří ČERNÝ. *GPS od A do Z*. 4. aktualizované vydání. Praha: eNav, s. r. o., 2006. ISBN 80-239-7516-1.
- [38] NARAMORE, Elizabeth, Jason GERNER, Scouarnec YANN LE and Timothy BORONCZYK. *PHP 6, MySQL, Apache: Vytváříme webové aplikace*. Brno: Computer Press a.s., 2009. 816 s. EAN:9788025127674.

## 4 PŘÍLOHY

Příloha A – <i>Zdrojový kód funkce pro plánování rozvozů</i> .....	88
Příloha B – <i>Zdrojový kód celé aplikace a další dokumenty</i> .....	90

## Příloha A – Zdrojový kód funkce pro plánování rozvozu

```
public function generovatRozvozStandard($mezniDobaRozvozu) {
    $skill = 0;

    while (1) {
        $baliky['size'] = 0;
        $vozidla = $this->vozidlaKRozvozu();
        $ridici = $this->ridiciKRozvozu();
        if ($mezniDobaRozvozu == 480) { //standard
            $baliky = $this->balikyKRozvozu(0);
        } else if ($mezniDobaRozvozu == 240) { //express
            $baliky = $this->balikyKRozvozu(1);
        }

        if ($vozidla['size'] == 0) {
            return -1;
        }
        if ($ridici['size'] == 0) {
            return -2;
        }
        if ($baliky['size'] == 0) {
            return -3;
        }

        if ($skill == 100) {
            return -4;
        }

        $payment = array();
        $this->baliky = array();
        $this->baliky = $this->vypocetSouradnic($baliky);
        $distančniMatic = $this->distančniMatic();
        $this->balikyOptimalizaceSeznam = array();

        foreach ($this->baliky as $key => $row) {
            $payment[$key] = $row[10]; //sort podle stupnu
        }

        array_multisort($payment, SORT_ASC, $this->baliky);

        for ($i = (count($this->baliky) - 1); $i > 0; $i--) {
            $this->baliky[$i] = $this->baliky[$i - 1]; //posun pro
            pridani depa na první pozici
        }

        $this->baliky[0][0] = 0; // id 0 pro matici vzdalenosti
        $this->baliky[0][1] = null;
        $this->baliky[0][2] = null;
        $this->baliky[0][3] = null;
        $this->baliky[0][4] = null;
        $this->baliky[0][5] = $this->depoLatitude;
        $this->baliky[0][6] = $this->depoLongitude;

        $idVozidla = 0;
        $idRidice = 0;
        $prirazenychBaliku = 0;
        $prekroceniCasu = false;
    }
}
```



```

$trasa = array();
$trasaZaloha = array();
$i = 1;

$this->balikyOptimalizaceSeznam[] = $this->baliky[0][0]; //depo
//na vozidlo
$celkovaVahaBaliku = 0;
$celkovyObjemBaliku = 0;
$zbyvajiciVaha = $vozidla[$idVozidla][2];
$zbyvajiciObjem = $vozidla[$idVozidla][3];

$trasa[$idVozidla]['vzdalenost'] = 0;

for (; $i < ($this->baliky['size']); $i++) { // na nule je depo
    $celkovaVahaBaliku+=$this->baliky[$i][1];
    $celkovyObjemBaliku+=($this->baliky[$i][2] * $this-
>baliky[$i][3] * $this->baliky[$i][4]) / 1000000.0; //v metrech krychlovych

    $trasaZaloha = $trasa;
    $trasa = $this->vypocitatTrasu($trasa, $idVozidla, $i,
$distančniMatice);

    if (($zbyvajiciVaha >= $celkovaVahaBaliku) &&
($zbyvajiciObjem >= $celkovyObjemBaliku * 1.3)) {
        if ($trasa[$idVozidla]['cas'] < $mezniDobaRozvozu) {
            //ok
            $prirazenychBaliku++;
            $trasa[$idVozidla]['prirazenych'] =
$prirazenychBaliku;
            $trasa[$idVozidla]['vozidlo'] =
$vozidla[$idVozidla][0];
            $trasa[$idVozidla]['ridic'] =
$ridici[$idRidice][0];
        } else { //nevejde se do casoveho limitu - navrat k
predchozi trase bez posledniho bodu
            //a prestup na dalsi vozidlo
            $prekroceniCasu = true;
            $trasa = $trasaZaloha;
            break;
        }
    } else { //nesplnuje podminku objemu nebo hmotnosti - navrat
k predchozi trase bez posledniho bodu
        //a prestup na dalsi vozidlo
        $trasa = $trasaZaloha;
        break;
    }
}

if ($trasa[$idVozidla]['vzdalenost'] > 0) {
    $this->zapsatRozvozDTB($trasa);
}
$kill++;
}
}

```

## Příloha B – *Zdrojový kód celé aplikace a další dokumenty*

Kompletní zdrojový kód aplikace včetně instalačního skriptu je uložen na přiloženém CD.

Dále je zde umístěn i BPMN diagram všech procesů a model databáze.