Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Informační systém přepravní společnosti

Diplomová práce

2020 Bc. Luboš Vojáček

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2019/2020

ZADÁNÍ DIPLOMOVÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Luboš Vojáček**
Osobní číslo:	**I17227**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Informační systém přepravní společnosti**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování

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: registrace uživatelů a přístup do systému podle jejich práv, přijímání a zpracování objednávek zákazníků, evidenci automobilů a řidičů, vytváření rozpisů denních jízd pro jednotlivá auta a řidiče, generování sestav dle zadaných kritérií pro potřeby vedoucích pracovníků, řízení údržby vozového parku, logistické řešení nakládky zboží a přepravy mezi jednotlivými lokalitami z pohledu minimalizace nákladů. Práce bude obsahovat: rešerši systémů, které se zabývají touto problematikou, analýzu procesů, které budou znázorněny v procesním BPMN diagramu s cílem dosažení maximální bezpečnosti aplikace, popis použitých technologií, návrh databáze a ER diagram s využitím „Crow's Foot" notace entity-relationship. Pro vytvoření aplikace bude využita databáze MySQL nebo Oracle a objektově orientovaný programovací jazyk Java s možností využití technologií pro podporu tvorby aplikací.

Rozsah pracovní zprávy: **50-60 stran**
Rozsah grafických prací: **–**
Forma zpracování diplomové práce: **tištěná**

Seznam doporučené literatury:

ŘEPA, Václav. Analýza a návrh informačních systémů. 1. vyd. Praha : Ekopress, 1999. 403 s. ISBN 80-86119-13-0.
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.
SCHILDT, Herbert. Mistrovství Java – Kompletní průvodce vývojáře. Brno: Computer Press a.s., 2014. 1224 s. EAN: 9788025141458.

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

Datum zadání diplomové práce: **5. listopadu 2019**
Termín odevzdání diplomové práce: **15. května 2020**

Ing. Zdeněk Němec, Ph.D.
děkan

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

V Pardubicích dne 15. listopadu 2019

Prohlašuji

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

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, 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.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 14. 8. 2020

Bc. Luboš Vojáček

PODĚKOVÁNÍ

Rád bych poděkoval svému vedoucímu diplomové práce Ing. Miloslavu Macháčkovi, Ph.D., za pomoc poskytnutou jak při studiu, tak při vypracování této diplomové práce. Také děkuji svým rodičům za trpělivost, kterou se mnou po celou dobu studia a psaní této práce měli. A v neposlední řadě pak firmě Chempex s.r.o., u které jsem mohl vytvářenou aplikaci konzultovat a i reálně testovat.

ANOTACE

Tato diplomová práce se zabývá návrhem systému a vytvořením funkční aplikace pro evidenci a logické řízení přepravní společnosti. Řeší zejména problematiku správy vozového parku, přijímání a zpracováním objednávek zákazníků a efektivní distribuci zakázek v prostředí malých a středních firem.

KLÍČOVÁ SLOVA

Přepravní systém, Java, MySQL, MariaDB, BPMN diagram, ER diagram, UML

TITLE

Transport company information system

ANNOTATION

This diploma thesis deals with the design of a system and the creation of a functional application for the registration and logical management of a transport company. It mainly solves the issue of fleet management, acceptance and processing of customer orders and the effective distribution of orders in the environment of small and medium-sized companies.

KEYWORDS

Transport system, Java, MySQL, MariaDB, BPMN diagram, ER diagram, UML

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

API	Application Programming Interface
BPMN	Business Process Model and Notation
CRC	Class Responsibility Collaborator
CSS	Cascading Style Sheets
EA	Enterprise Architect
EET	Elektronická evidence tržeb
FTP	File Transfer Protocol
GPL	General Public License
GPS	Global Positioning System
HTML	Hypertext Markup Language
IBM	International Business Machines Corporation
ID	Identification
IDE	Integrated development environment
IS	Informační systém
IT	Information Technology
JAR	Java Archive
MySQL	My Structured Query Language
PHP	Hypertext Preprocessor
SPZ	Státní poznávací značka
SQL	Structured Query Language
SSH	Secure Shell
STK	Státní technická kniha
UML	Unified Modeling Language
UP	Unified Process
USDP	Unified Software Development Process

ÚVOD

Cílem této práce je vytvořit funkční aplikaci zabývající se nástroji pro evidenci a logické řízení přepravní společnosti.

Již od pradávna má lidstvo potřebu obchodovat a přepravovat zboží z jednoho místa na druhé. Dříve k tomu sloužili kupci, kteří cestovali od vesnice k vesnici a nabízeli své produkty. V té době si vystačili s vlastní pamětí a později s psanou formou záznamů. Dnes již toto v globalizovaném světě nestačí. Je potřeba centralizovaného systému, který bude efektivně a spolehlivě řídit jednotlivé toky zboží a vozidel. Proto vzniklo a vzniká mnoho systémů pro řízení vozových parků firem a přepravy jejich zboží.

Tato diplomová práce se zabývá výše zmíněnou tématikou a předkládá funkční řešení informačního systému přepravní společnosti. Práce je rozdělena do dvou hlavních částí, a to teoretické a praktické. V teoretické části se nejdříve zabývá sběrem požadavků od skutečného zákazníka, analýzou již existujících řešení a stanovením hlavních požadavků na praktickou realizaci. Následně jsou popsány hlavní nástroje použité při vývoji systému včetně popisu metodiky UP (Unified Process).

V praktické části je podrobně popsán datový model včetně důvodu existence a omezení jednotlivých tabulek a atributů. Následně jsou pak popsány v uživatelské příručce jednotlivé moduly vyvíjeného systému s popisem jejich využití. Dále práce obsahuje instalační příručku s popisem nasazení a vytvoření databáze. V poslední části pak následuje ukázka zdrojového kódu, soupis externích knihoven, závěr a rejstřík použité literatury a příloh.

Součástí této práce je také přiložená funkční vzorová aplikace, včetně aplikace pro generování hesla, konfiguračního souboru hlavní aplikace, script pro vytvoření databáze, script pro vytvoření databáze včetně testovacích dat a vytvořené diagramy.

Pro návrh a analýzu aplikace byl využit Enterprise Architect, pro návrh databáze MySQL Workbench a pro BPMN diagram pak ARIS Express. Samotná aplikace pak byla vytvořena pomocí IDE NetBeans za použití programovacího jazyku Java. Samotná databáze byla testována na lokální MySQL databázi a následně nasazena na virtuální server s databází MadriaDB.

1. TEORETICKÁ ČÁST

V této části se práce zabývá teoretickým pohledem na danou problematiku, zejména pak vysvětlením použitých technologií, analýzou existujících řešení a požadavků na vytvářený systém.

1.1 Informační systém přepravní společnosti

V první řadě je třeba si definovat, co je to vlastně informační systém (IS). Za tímto účelem můžeme toto slovní spojení rozdělit na slova informace a systém. Informace nám přináší nový údaj neboli snižuje neurčitost (entropii) světa. V informatice je nejmenší jednotkou informace bit nabývající hodnoty 0 a 1. Systém je pak uspořádaná množina prvků, kde každý prvek má vazbu aspoň na jeden další. Prvky ovlivňují jak sebe navzájem, tak chování samotného celku.

Z toho vyplývá, že informační systém je vlastně systém pro sběr, přenos, údržbu, zpracování a poskytování informací. Podle této definice se stává pojem informační systém velice abstraktní. V dnešní době si představíme pod tímto pojmem nejčastěji nějaké softwarové dílo.

Informační systém přepravní společnosti je tedy jednou z mnoha možných implementací informačního systému. Tento pojem je však stále velmi široký, a proto je dobré konkretizovat, co si vlastně pod touto konkrétní implementací představujeme.

1.2 Požadavky na systém

V rámci zadání diplomové práce jsou požadovány následující funkcionality:

- registrace uživatelů a přístup do systému podle jejich práv;
- přijímání a zpracování objednávek zákazníků;
- evidence automobilů a řidičů;
- vytváření rozpisů denních jízd pro jednotlivá auta a řidiče;
- generování sestav dle zadaných kritérií pro potřeby vedoucích pracovníků;
- řízení údržby vozového parku;
- logické řešení nakládky zboží a přepravy mezi jednotlivými lokalitami z pohledu minimalizace nákladů.

Pro potřeby sepsání této práce byla navázána spolupráce se společností Chempex s.r.o. (dále zadavatel), která provozuje v Pardubickém kraji síť několika franšízových prodejen

barev pod značkou Dům barev. Tato firma již delší dobu řeší problémy s řízením svého vozového parku. Výhodou této spolupráce byla možnost reálně testovat tuto aplikaci a díky pracovní zkušenosti v této firmě i podrobná znalost reálné problematiky daného segmentu malých a středních firem, na kterou by tato práce mohla ve skutečném světě cílit.

Analýza požadavků na systém vychází z podrobné konzultace s vedoucími pracovníky prodejen a jejich zaměstnanci, kteří řeší každodenní přijímání objednávek od zákazníků, expedici zboží a správu svého vozového parku.

V současné době firma používá pro příjem objednávek ve většině případů elektronickou formu pomocí emailu nebo telefonického kontaktu. Následně tyto objednávky vytiskne nebo ručně zapíše do knihy objednávek. Podle této knihy jsou následně objednávky postupně připraveny a expedovány v tu chvíli dostupnými vozy nebo přepravními společnostmi. Tento způsob však zapříčiňuje neefektivní způsob dopravy. Pravidelně se stává, že dvěma zákazníkům na stejné trase je expedice zboží provedena dvěma různými vozy. Toto je zejména zapříčiněno nesdílením objednávek mezi jednotlivými provozovnami.

Dalším problémem se zejména ukázala například evidence STK nebo najetých kilometrů. Opakovaně byla kontrola STK u vozů prováděna těsně před vypršením a v několika případech i po propadnutí termínu. Tento problém nevyřešil ani současný trend státní správy o zasílání upozornění o konci platnosti STK. V praxi se totiž tato informace mnohdy nedostala např. k řidiči zodpovědného za stav vozu. Vzhledem k povinnosti společnosti vést knihu jízd a samozřejmě i k vlastní kontrole řidičů, zda nevyužívají firemní vozy k nepovoleným jízdám, potřebuje firma sledovat stav tachometrů jednotlivých vozů a cest, které provedly.

Vzhledem k tomu, že společnost nedisponuje při své velikosti a zaměření vlastním IT odborníkem, požaduje po systému uživatelskou jednoduchost jak při správě, tak samotném nasazení. Při následném průzkumu trhu a situace u ostatních společností podobné velikosti bylo zjištěno, že tento jev je rozšířený. Firmy této velikosti tyto problémy buď řeší externím IT technikem, anebo mají ve svých řadách zaměstnance se základními znalostmi, který je schopen většinu základních úkonů provést sám.

Z konečné souhrnné analýzy vyplynuly následující požadavky, které byly použity i při hodnocení již existujících systémů zabývajících se touto nebo podobnou tématikou. Tyto požadavky se plně slučují se zadáním diplomové práce a rozšiřují ho. Požadavky jsou:

- registrace uživatelů a přístup pomocí práv;

- přijímání a zpracování objednávek;
- sdílení záznamů mezi vícero středisky;
- evidence vozidel a jejich správa;
- evidence řidičů;
- vytváření rozpisů jízd;
- generování sestav dle zadaných kritérií;
- logické řešení přepravy nákladu mezi lokalitami;
- minimalizace nákladů na přepravu;
- intuitivní ovládání systému;
- minimalizace ceny na pořízení;
- jednoduchost správy;
- jednoduchost zavedení;
- budoucí rozšiřitelnost.

1.3 Rešerše existujících řešení

V této části se bude práce zabývat rešerší již existujících řešení. V rámci sběru dat bylo zkoumáno mnoho různých systémů, zde jsou nejrelevantnější zástupci. V první řadě byl analyzován již používaný pokladní systém TRIFID a externí přepravní systém dopravce TOPTRANS EU. Následně byla vybrána dvě již existující řešení pro řízení vozového parku, a to TRAXEE a MAPON.

1.3.1 TRIFID

Prvním zkoumaným řešením je pokladní systém od stejnojmenné společnosti TRIFID software, která byla založena v roce 1994. Tato firma se zabývá především vytvářením programových a hardwarových řešení pro firmy a jejich obchody. [1]

Nejedná se sice o program řešící přímo danou problematiku, kterou se zabývá tato diplomová práce, ale vzhledem k tomu, že výsledné řešení bude přímo spolupracovat s tímto programem, je vhodné ho prozkoumat.

Jak již bylo řečeno, jedná se o pokladní systém, který firma zadavatele používá nejenom k markování na kasách, ale i k vytváření faktur, správě skladu a zboží, evidenci zákazníků a jejich objednávek, správu samotných zakázek a mnoha dalším účelům.

Grafické prostředí aplikace je velice jednoduché a tím umožňuje efektivní a rychlý běh. Pro jakoukoli činnost postačí pouze klávesnice a klávesové zkratky. Použití myši je vyžadováno až v případě, kdy uživatel požaduje export dat nebo jiné nadstandardní služby.

Po zapnutí aplikace a přihlášení zvoleného uživatele se otevře úvodní rozcestník.

Obrázek 1: Aplikace TRIFID

Zdroj: vlastní, snímek aplikace [1]

- Položka „přehledy“ umožňuje vytváření široké škály přehledů od tržeb, přes toky zboží a práce, přehledů EET, až po export například do XLS formátu. Výhodou je velká škála možností pro filtrování dle zadaných kritérií například podle datumu, formy platby, typu zboží nebo pracovní stanice;
- Po zvolení položky „prodej“ se otevře pokladní okno, v kterém pracovník na kase může namarkovat zákazníkem nakupované zboží. Aplikace v rámci toho může spolupracovat například s pokladním scannerem ať už přímo od firmy TRIFID, tak se scannery od jiných výrobců;
- V rámci markování zboží aplikace umožňuje dodatečnou úpravu položek včetně změny ceny, ať již podle připravených cenových šablon, tak v případě potřeby přímo ručním zadáním. Taktéž je možnost výběru zákazníka a předpřipravených cenových nabídek;
- Položka „doklady“ zahrnuje správu objednávek, nabídek, prodejek, faktur a dobropisů, skladů, zákazníků, reklamací a automatický import dokladů podle různých kritérií;
- „Číselníky“ zajišťují správu zboží, obalů, práce, předdefinovaných textů, obchodních partnerů, ceníků dodavatelů a nástrojů pro inventuru a přecenění;

- „Výběr pracovního skladu“ zajišťuje správu a evidenci jednotlivých skladů a umožňuje náhled do skladových položek v rámci jednotlivých let a poboček. Synchronizace skladů mezi pobočkami není možná online, a proto musí být proveden fyzický import. Není proto možné zjistit stavy v reálném čase;
- „Doplňkové programy“ a „systémové funkce“ zajišťují zejména nastavení a správu celé aplikace. Umožňují zálohu, obnovu a správu konfigurací, správu EET modulu, údržbu dat, komunikaci s tvůrci a mnoho dalších nezbytných operací pro správné ladění a chod programu;
- Poslední 3 položky zajišťují přihlášení a odhlášení pracovníka. Zde je to myšleno jako přihlášení pracovního profilu, v praxi se totiž každý pracovník nepřihlašuje individuálně. Toto přihlášení spíše slouží na rozlišení běžného uživatele a správce systému. Každý účet může mít rozdílná práva a může být zabezpečen heslem;
- Poslední položka pak ukončí program. Při ukončení se program vždy ptá, zda má zálohovat data. Zde je potřeba dbát na to, aby nebyl program spuštěn na více stanicích, jinak hrozí narušení indexových souborů.

Z výše uvedeného jasně vyplývá, že i když se nejedná přímo o řešení zabývající se zadanou tématikou, je toto řešení schopné zajistit většinu funkcionalit požadovaných v zadání diplomové práce. Díky tomu se tento program stal hlavní inspirací pro realizovaný návrh řešení. Zejména pak jeho způsob zpracování a evidence objednávek zákazníků a jejich správa.

Z osobní konzultace se zaměstnanci také vyplynul požadavek, aby výsledné řešení bylo podobné této aplikaci ve smyslu jednoduchosti a rozložení.

V rámci dlouholeté pracovní zkušenosti v tomto odvětví jsem měl možnost vyzkoušet i jiná řešení a dovolím si dle vlastních zkušeností tvrdit, že tento program je jeden z nejlepších na našem trhu. Jeho výhodou je zejména rychlost, jednoduchost, spolehlivost a v neposlední řadě i výborná podpora ze strany vývojářů, kteří aktivně a rychle reagují na případné podněty zákazníků.

Aplikace je k dostání na stránkách tvůrců www.trifid-sw.cz. Zde je možné stáhnout demoverzi aplikace pro vyzkoušení, která umožňuje téměř plnohodnotný přístup a je omezena v podstatě pouze zamezením vytvoření více než jednoho skladu. [1]

1.3.2 TOPTRANS

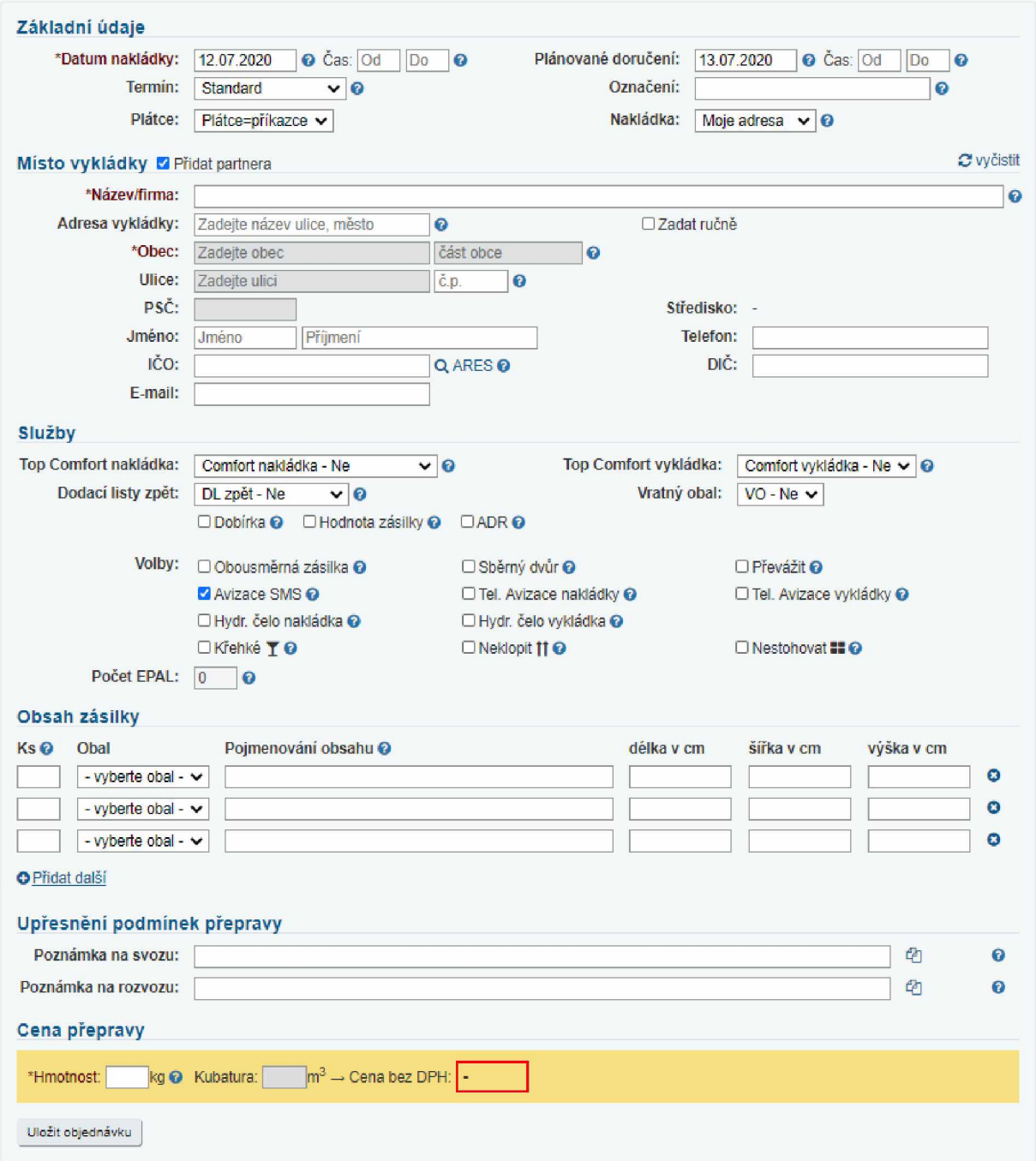
Dalším zkoumaným řešením je řešení od společnosti TOPTRANS EU, a.s., se sídlem na Slovensku, podnikající v ČR prostřednictvím své organizační složky se sídlem v Praze. [2]

Společnost TOPTRANS EU se zabývá přepravou kusových zásilek a balíků po celé České republice do 24 hodin a do Slovenské republiky do 24 nebo 48 hodin. Společnost také nabízí skladování a balení zásilek včetně logistiky s tím spojené. [2]

Toto řešení bylo do rešerše zahrnuto jako ukázka externí možnosti řešení přepravy zásilek. V současné době zadavatel využívá tento externí způsob dopravy v případě, kdy vlastní přeprava není finančně výhodná. Díky tomu bylo možné pro účely rešerše získat přístup do objednávkového formuláře pro přepravu a osobní otestování této varianty.

Objednávka přepravy probíhá přes webové rozhraní, kdy se po přechodu na úvodní stránku zákazník přihlásí do svého profilu. Po přihlášení se zákazníkovi automaticky otevře formulář pro zadání objednávky s rozsáhlou možností individuálního nastavení. Nejdůležitějšími parametry jsou pak:

- datum, čas a místo nakládky;
- požadované služby jako jsou např. způsob platby, zacházení se zásilkou, způsob vyskladnění nebo telefonické oznámení vykládky;
- obsah zásilky včetně počtu, typu obalu nebo velikosti;
- upřesňující podmínky přepravy;
- teoretická hmotnost zásilky, která určuje cenu přepravy, pokud se nejedná o rozměrově nadměrný náklad.



Obrázek 2: Formulář

Zdroj: vlastní, snímek webového rozhraní [2]

V rámci služby je také možná evidence faktur, zadávání reklamací nebo tisk přepravních štítků.

I když je celý proces objednávky přepravy poměrně jednoduchý a formulář pro vytvoření působí příjemným a přehledným dojmem, opakovaně jsem se setkal s množstvím problémů u kolegů, kteří nemají s tímto způsobem přepravy tolik zkušeností. Vzhledem k velkému

množství nastavení je jednoduché udělat chybu, která vede k nepříjemným nedorozuměním. Tomuto nezabraňují ani informační vysvětlivky u jednotlivých zadávacích polí. Vzhledem k velikosti společnosti by proto bylo na místě vytvořit krátké instruktážní video dostupné přímo u vyplňovaného formuláře.

Další zajímavostí je, že hmotnost zásilky vyplňuje přímo zákazník, a tím i určuje celkovou cenu zásilky. Nesetkal jsem se za poslední 2 roky s jediným případem, kdy by dopravce zásilku převažoval. Je tedy jasné, že tento parametr je spíše výstraha a pojistka pro případné nepoctivé obchodní partnery.

Po vytvoření objednávky se formulář odešle a zákazník je vyzván k vytištění přepravního lístku. Po příjezdu řidiče dojde pouze k načtení kódu ze štítku a převzetí zásilky. Jedná se tedy o poměrně jednoduchý a rychlý způsob přepravy kusového zboží v případě, že se nevyplatí vlastní doprava.

V porovnání s ostatními přepravními společnostmi si TOPTRANS EU stojí velice dobře a je v rámci firmy zadavatele vnímán jako spolehlivý přepravce. Za zmínku tu stojí například konkurenční společnost GEIS, se kterou má zadavatel velice špatné zkušenosti díky opakovaným poškozením, ztrátám nebo vykradením zásilek a neochotou tyto problémy řešit. V poslední době situace eskalovala do takové míry, že dodavatelé společnosti zadavatele mají zakázáno používat tuto přepravní službu. Dle názoru zadavatele, ale i jiných dotazovaných firem, patří GEIS k jedněm z nejhorších společností pro přepravu tohoto typu zboží.

V rámci rešerše byli dotazováni i řidiči přepravní firmy, a díky tomu byla možnost si udělat základní představu o logickém řízení přepravy.

1.3.3 TRAXEE

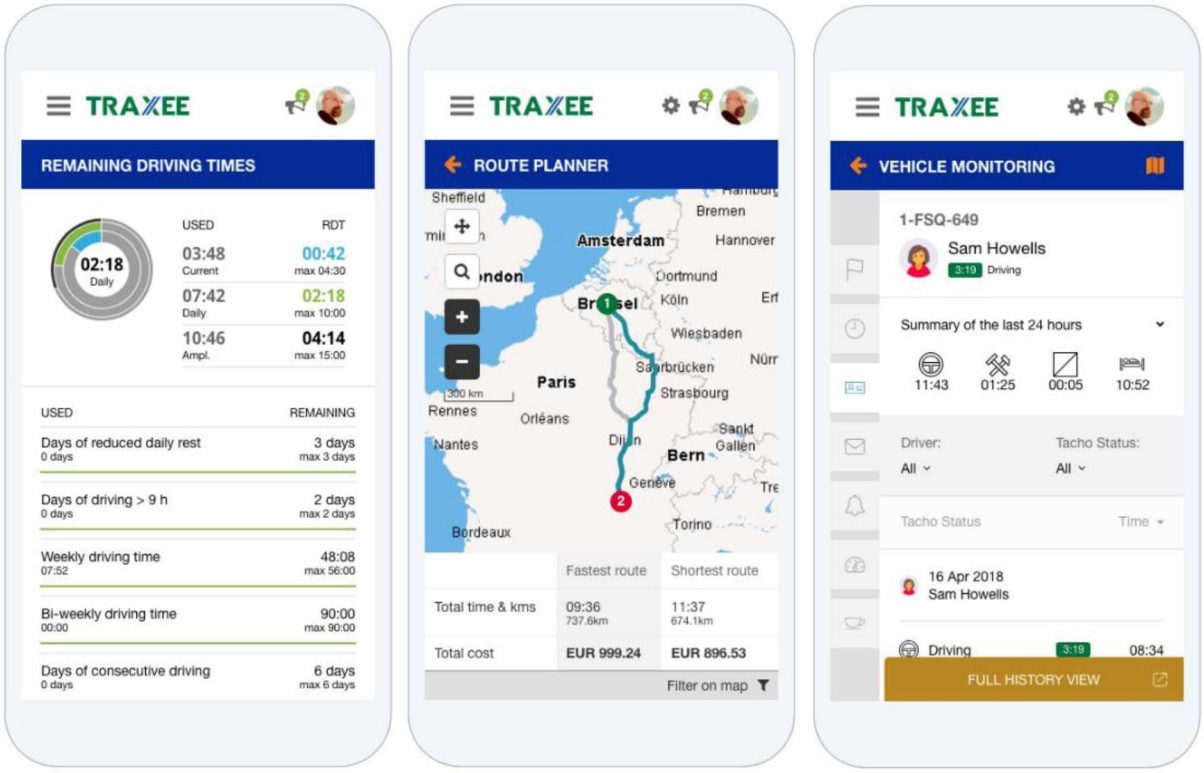
Jedná se o asistenta pro řízení vozového parku pomocí automatizace a digitalizace procesů souvisejících s provozem firemních vozidel. Poskytuje graficky přehledné a intuitivní rozhraní pro správu vozového parku. Program je k vyzkoušení na stránkách www.wabco-traxee.com. [3]

Umožňuje následující funkcionality:

- identifikaci řidiče;
- pozice a stav celého vozového parku;
- registrování doby tachografu;

- sledování zbývající doby řízení a odpočinku;
- správu aktivit;
- automatické vzdálené stažení dat tachografů a jejich archivaci;
- generování tradičních cest;
- textové zprávy;
- analýzu ECO jízdního stylu;
- pokročilé zprávy a alarmy;
- výpočet efektivních tras pro nákladní dopravu, včetně nákladů na mýtné.

Systém je dostupný v mnoha jazykových lokalizacích včetně češtiny, angličtiny, němčiny, francouzštiny a mnoha dalších. Demoverze systému je k dispozici v anglické, francouzské, německé a italské lokalizaci. Na stránkách produktu jsou k dispozici i podrobná výuková videa. [3]



Obrázek 3: Aplikace TRAXEE

Zdroj: vlastní, snímek webového rozhraní [3]

1.3.4 MAPON

Jedná se o kompletní systém řešení správy vozového parku. [4]

Nejzajímavější funkce MAPONU jsou:

- sledování GPS v reálném čase;
- digitální tachograf s možností vzdáleného stažení;
- sledování údržby vozového parku včetně sledování termínů servisu nebo vypršení licencí;
- sledování teplot chladírenských vozů;
- monitorování spotřeby paliva;
- sledování chování řidičů;
- správu úloh pomocí aplikace Mapon GO;
- mobilní varianta aplikace.

Mapon GO je aplikace k řízení pracovníků mimo kancelář. Umožňuje 4 základní funkcionality: [4]

Manažerování

Je možno vytvoření jednoduchých a rychlých úkolů pro řidiče a pracovníky v terénu. Aktualizovat trasu řidiče a pracovní plán tak, aby se předešlo zbytečným jízdám. Všechny úkoly se aktualizují v reálném čase. Je možná i zpětná vazba od samotných řidičů; [4]

Správa řidičů

Informování řidičů o úkolech a výběr vhodné navigační aplikace;

Identifikace řidičů a záznamy o jízdě

Možnost identifikace řidiče s možností rozlišení služebních a soukromých jízd. Reporty ke snížení provozních nákladů. Identifikace řidiče a jeho vozidla v daném časovém úseku. Umožnění generování pohledů k výpočtu mezd; [4]

Komunikace pomocí zpráv

Možnost spojení s terénními pracovníky. Příjem faktur přímo od řidiče. Předávání snímků poškozeného zboží v reálném čase. Umožnění archivace zpráv. [4]

Obrázek 4: Aplikace MAPON

Zdroj: vlastní, snímek webového rozhraní [4]

1.3.5 Vyhodnocení

	TRIFID	TOPTRANS	TRAXEE	MAPON
Registrace uživatelů a přístup pomocí práv	ANO	ANO	ANO	ANO
Přijímání a zpracování objednávek	ANO	ANO	?	ANO
Sdílení záznamů mezi vícero středisky	NE	ANO	ANO	ANO
Evidence vozidel a jejich správa	NE	NE	ANO	ANO
Vytváření rozpisů jízd	NE	NE	ANO	ANO
Generování sestav dle zadaných kritérií	ANO	NE	ANO	ANO
Logické řešení přepravy nákladu mezi lokalitami	NE	?	ANO	ANO
Minimalizace nákladů na přepravu	NE	ANO	ANO	ANO
Intuitivní ovládání systému	ANO	ANO	ANO	ANO
Minimalizace ceny na pořízení	ANO	ANO	?	?
Jednoduchost správy	ANO	ANO	?	?
Jednoduchost zavedení	ANO	ANO	?	?
Budoucí rozšiřitelnost	ANO	NE	?	?

Tabulka 1: Vyhodnocení

Zdroj: vlastní

? = Není možné jednoznačně odpovědět vzhledem k dostupným informacím.

I přesto, že se první dva systémy nezabývají přímo řešenou problematikou, i tak poskytují cenné informace a ukázku, jak se dají určité problémy řešit. Například zkoumaný pokladní systém podává názornou ukázku skvělého řešení faktur, evidence zákazníků a jejich objednávek. Taktéž je potřeba přihlédnout k budoucí spolupráci a uzpůsobit tak ovládání nové aplikace. Naopak přepravní systém společnosti TOPTRANS EU, a.s. je krásným náhledem do skutečného fungování objednávkového systému dopravy z pohledu zákazníka.

Poslední dvě rešerše pak ukazují již existující řešení dané problematiky. A ukazují, co vše je možné nebo co vše zákazníci můžou vyžadovat. V tomto ohledu se zdá lepší systém MAPON, který se tváří jako komplexnější a přehlednější.

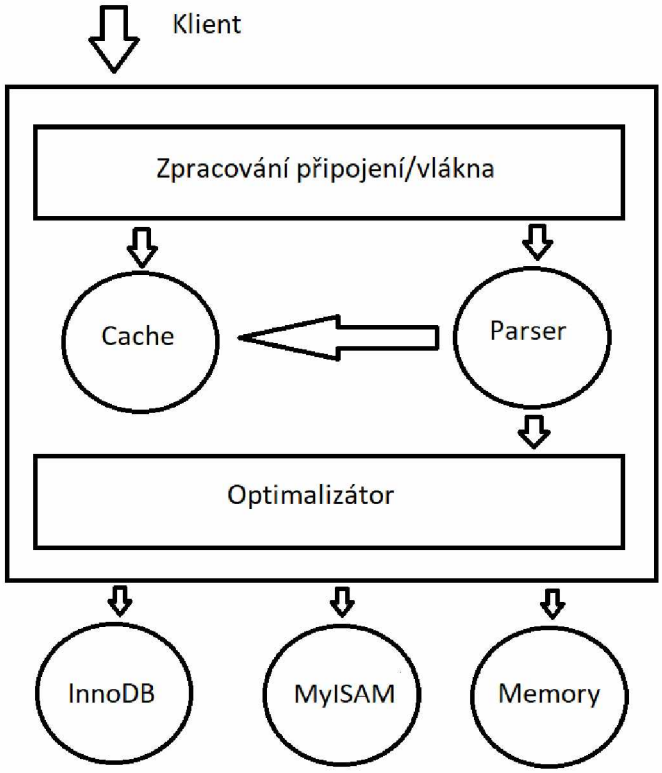
1.4 Použité technologie a služby

V této části budou popsány základní technologie a služby použité k návrhu a tvorbě výsledného systému. Zvláštní pozornost je pak věnována zejména univerzálnímu jazyku pro modelování UML a metodice UP. Metodika UP nám totiž umožňuje naplánovat správný vývoj systému a předejít zbytečným chybám, které mohou výrazně prodloužit dobu a cenu vývoje.

1.4.1 Databáze MySQL

Celým názvem My Structured Query Language neboli česky systém pro řízení databáze. Jedná se o otevřený systém řízení báze dat, u kterého se uplatňuje relační databázový model. V současné době je vlastněný společností Oracle Corporation. Původně byl vytvořen švédskou firmou MySQL AB a za jeho vznikem stojí dvojice autorů David Axmark a Michael Widenius. Společnost Oracle tento systém nabízí ve dvou variantách, a to pod bezplatnou licencí GPL a komerční variantou. [5][6]

Komunikace s databází probíhá pomocí jazyka SQL. Díky tomu, že se jedná o volně šiřitelný systém s jednoduchou implementací na různé operační systémy, získal velkou oblibu mezi uživateli a značný podíl na současném trhu. Nejčastěji je využit v kombinaci s operačním systémem Linux, webovým serverem Apache a programovacím jazykem PHP. Jelikož se jedná o otevřený systém, je umožněno uživateli, aby přímo vstoupil do zdrojového kódu. [5][6]



Obrázek 5: Architektura MySQL serveru

Zdroj: vlastní

1.4.2 Jazyk SQL

Jazyk SQL neboli Structured Query Language se zrodil v laboratořích společnosti IBM v Jose v Kalifornii na konci 70. let 20. století. Původně byl tento jazyk vyvinut pro relační databázový systém společnosti IBM. [6]

Jedná se o neprocedurální jazyk, což znamená, že nepopisuje, jak se má něco provést, ale co se má provést. [6]

Ukázka kódu SQL pro nalezení vozidel, kterým v následujících 30 dnech propadne STK:

```
SELECT ID_Vozidlo, Nazev, STK FROM Vozidla WHERE STK BETWEEN
CURDATE() AND DATE_ADD(CURDATE(), INTERVAL 30 DAY);
```

1.4.3 Netbeans

Jedná se o distribuované vývojové prostředí (IDE). Nejčastěji se využívá k programování v jazyce JAVA, ale díky své architektuře umožňuje i práci s dalšími jazyky jako jsou JavaScript, PHP, HTML5, CSS, C, C++ a další. [7]

Je vlastněn společností Oracle Corporation, která ho distribuuje pod svobodnou licencí (Open Source) většinou v kombinaci s Apache Tomcat. Společnost dnes nabízí dvě varianty produktu, a to vývojové prostředí NetBeans IDE a vývojovou platformu NetBeans Platform. [7]

1.4.4 Jazyk JAVA

Tento moderní programovací jazyk vyvinula v roce 1995 společnost Sun Microsystem. Jedná se o multiplatformní, objektově orientovaný programovací jazyk. Jeho vývoj byl založen a ovlivněn na programovacích jazycích C a C++. Sám pak ovlivnil například programovací jazyk JavaScript. Jeho hlavní výhodou je:

- jednoduchá syntaxe;
- jednoduchá přenositelnost mezi platformami;
- nízká implementační závislost.

Nevýhodou je pak zdlouhavější start. Je totiž nutné, aby byl nejdříve celý kód přeložen, a teprve poté spuštěn. [8]

V současné době se těší velké oblibě mezi programátory, jejichž počet se pohybuje okolo deseti milionů.

Ukázka části kódu jazyka Java použitého v aplikaci pro výpočet vzdálenosti mezi městy:

```
for (int i = 1; i < mesta.size(); i++) {
  soucet = soucet + getDriveDist(mesta.get(i - 1),
mesta.get(i));
}
jTextField1.setText((soucet / 1000) + " km");
```

1.4.5 Český hosting

Jedná se o českého poskytovatele, který poskytuje webhosting pro internetové prezentace a aplikace. Také zajišťuje provoz a registraci internetových domén. V rámci nabízených služeb je možné zakoupit i virtuální server umožňující zaparkování vícero domén a vytvoření databází MySQL/MariaDB pro tyto domény nebo vzdálené aplikace. Databáze MySQL/MariaDB jsou vzdáleně přístupné přes SSH tunel. [9]

Stránky poskytovatele jsou k nalezení na adrese www.cesky-hosting.cz. V rámci služeb je zajištěno i odborné poradenství závislé podle zvolené služby a cenové kategorie. Pro řešení

drobných dotazů je možné přímo na webových stránkách položit otázku přes integrovaný chat zákaznické podpory.

1.4.6 Databáze MadriaDB

Jedná se o multiplatformní relační databázi, která je odnoží databáze MySQL. Je vyvíjena jako software s otevřeným zdrojovým kódem s přístupem k datům pomocí rozhraní SQL. Důvodem jejího vzniku byla obava původních vývojářů o zachování nezávislosti a svobodné licence poté, co původní databázi MySQL odkoupila společnost Oracle. Hlavním vývojářem je tvůrce původní databáze MySQL Michael Widenius a pojmenování MadriaDB je odvozenina od jména jeho mladší dcery Marii. [10]

Protokoly databáze jsou kompatibilní s protokoly MySQL a navíc je rozšířena i o některé nativní operace. [10]

1.4.7 Enterprise Architect

Enterprise Architect (EA) je analytický nástroj pro systémové řešení návrhu aplikace od společnosti Sparx Systems. Zahrnuje virtualizaci, modelování, testování a údržbu systémů, softwarů a procesů. Obsahuje kompletní CASE nástroje pro systémovou analýzu a návrh. Zahrnuje různé diagramy a metodiky pro tvorbu BPMN diagramů, Class Diagramu, UML a mnoha dalších. [11]

1.4.8 MySQL Workbench

MySQL Workbench je vizuální nástroj pro návrh databáze MySQL. Jedná se o jednoduchý nástroj pro databázové architekty, vývojáře a administrátory. Poskytuje ucelenou škálu pro datové modelování, vývoj SQL, kompletní nástroje pro správu a konfiguraci serveru, správu uživatelů, zálohování a mnoho dalších funkcí. Umožňuje jednoduchou správu tabulek a jejich následné navázání v rámci vytvoření korektního ER diagramu. [12]

1.4.9 Aris Expres

Jedná se o bezplatný nástroj pro modelování, analýzu a správu podnikových procesů. Používá se například k modelování BPNM diagramů. Kromě komplexních nástrojů pro vytvoření modelu navrhovaného systému umožňuje také jeho export do souboru. [13]

1.4.10 Regulární výrazy

Regulární výrazy umožňují rychlé prohledání vstupního textu a případné nahrazení jeho částí. V dnešní době se snad nenajde textový editor, který by regulární výrazy nepodporoval. Regulární výraz je v podstatě textový řetězec, díky kterému můžeme popsat množinu slov. Často se regulární výraz popisuje jako jednoduchý způsob, jak popsat konečný automat. Jedná se o výborný způsob, jak kontrolovat textové vstupy do aplikace.

Princip fungování regulárního výrazu se dá skvěle demonstrovat na následujícím příkladu, kdy programátor potřebuje zajistit, aby vstupní textový řetězec byl přijat, pouze pokud obsahuje emailovou adresu.

Ukázka použití v jazyce Java

```
Pattern p;
Matcher m;
//kompilace a překlad regulárního výrazu do interní formy
p = Pattern.compile("[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,4}");
//vytvoření instance třídy Matcher
m = p.matcher("lubos.vojacek@centrum.cz");
//porovnání celého řetězce se vzorem
if (m.matches()) {
System.out.println("Ano jedná se o email");
} else {
System.out.println("Ne nejedná se o email");
}
```

Vysvětlení zápisu regulérního výrazu pro email

- [a-zA-Z0-9._-] – v této části nám hranaté závorky říkají, že text musí obsahovat alespoň jeden znak. Tento znak může být malé písmeno, velké písmeno, číslo v rozmezí 0-9 nebo speciální znak jako je tečka, podtržítko nebo pomlčka. Text může obsahovat libovolný počet znaků;
- @[a-zA-Z0-9.-] – v této části nám výraz říká, že na začátku musí být povinný znak zavináč následovaný jedním nebo více znaky dle předchozí definice;
- \\.[a-zA-Z]{2,4} – v této části následuje povinná tečka a 2 až 4 znaky velkých nebo malých písmen.

Jednotlivé části regulárního výrazu pak odděluje znaménko +.

1.4.11 ADMINER 4.7.7.

Jedná se o webové rozhraní napsané v jazyce PHP pro jednoduchou správu databází MySQL. Dále umožňuje také správu databází MariaDB, PostgreSQL, SQLite, MS SQL, Oracle, Firebird, SimpleDB, Elasticsearch a MongoDB. Jedná se o alternativu k PhpMyAdminu.

1.4.12 Google Maps API

Tato služba poskytuje zeměpisná data a mapové komponenty. Ve vytvořené aplikaci je využita k získání vzdálenosti mezi dvěma adresami. K jejímu využívání je potřeba vlastnit přístupový klíč, který je možno vygenerovat po vytvoření účtu na adrese https://cloud.google.com/maps-platform/.

1.4.13 UML

UML (Unified Modeling Language, unifikovaný modelovací jazyk). Jedná se o univerzální jazyk pro visuální modelování systémů. Jazyk je nejčastěji spojován s modelováním objektově orientovaných systémů. Není však určen pouze k tomuto účelu, a má mnohem širší využití díky svým zabudovaným rozšiřujícím mechanismům. Je navržen tak, aby spojoval nejlepší postupy modelovacích technik a softwarového inženýrství. [14] [15]

Jazyk UML není však metodika! To znamená, že poskytuje nástroje pro vytvoření modelu, ale nedefinuje postup, jak tento model vytvořit. Jednou z metodik pro vytvoření modelu je například metodika UP, která je popsána níže. [14] [15]

1.4.14 UP

Metodika UP (Unified Process) je jednou z mnoha metodik založených na jazyce UML. Jedná se o objektově orientovanou metodiku známou také pod názvem USDP (Unified Software Development Process). Metodika byla vytvořena týmem autorů jazyka UML, a i když za ní nestojí pouze jeden člověk, je za jejího otce často považován Ivory Jacobson. [14] [15]

Zatímco se jazyk UML stará o jazykovou část projektu, metodika UP zajišťuje část procesní. UP je vlastně obecná metoda tvorby softwaru. Neposkytuje tedy pouze jedno konkrétní řešení, ale stanovuje návod, jak projekt rozložit na menší části a jak s těmito částmi pracovat. [14] [15]

Metodika UP definuje základní otázky kdo, co, kdy a jak. Dá se také definovat jako řízení rizik a případů užití, soustředění se na architekturu, iterace a přizpůsobení tvorby

softwarového procesu. Jak již bylo řečeno, metodika UP je obecnou metodou pro tvorbu softwaru, a proto je nutné pro každý nový projekt vytvořit její novou instanci. [14] [15]

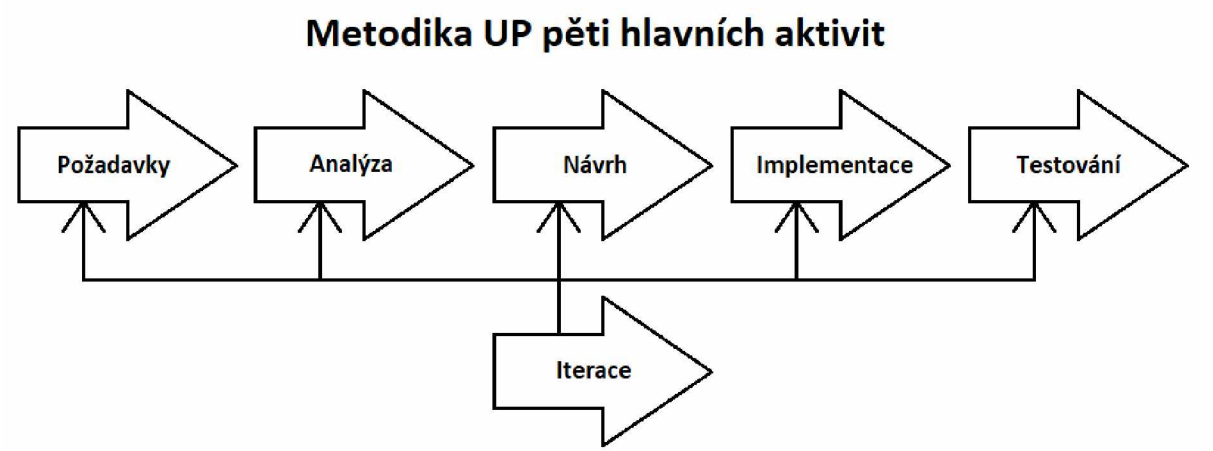
Základním stavebním prvkem metodiky jsou iterace. Iterace lze chápat jako jakýsi miniprojekt. Každý problém, který chceme řešit, můžeme rozdělit na menší části označované jako fáze. Každá fáze je složena z jedné nebo více iterací neboli opakování. Počet iterací ve fázi závisí na velikosti projektu. Každá fáze je vždy zakončena zvoleným hlavním milníkem. [14] [15]

Práce na jednotlivých iteracích může probíhat paralelně, pokud to závislosti mezi nimi dovolí. Díky tomu je urychlena práce na návrhu systému. [14] [15]

Metodika UP definuje čtyři fáze:

- **Zahájení** – období plánování. Fáze se zabývá stanovením proveditelnosti, definováním požadavků pro systém, odhadem rizik, obchodními případy a odsouhlasením cílů projektu investorem. V této fázi mohou být vytvořeny i funkční prototypy sloužící jako ukázka pro investora, který se na jejich základě může rozhodnout, zda v práci pokračovat. Během této fáze obvykle nedochází k testování, jelikož výsledné prototypy budou v budoucnu zahozeny nebo přepracovány. [14] [15];
- **Rozpracování** – období vytváření architektury. Zabývá se vytvořením spustitelného architektonického základu, vytvořením statického a dynamického modelu UML a případů užití. Taktéž je nutné aktualizovat odhad rizik, obchodních případů, plánu projektu a stabilizaci vize projektu. V poslední řadě pak dosáhnout shody s investorem a rozhodnout, zda pokračovat v projektu. [14] [15];
- **Konstrukce** – počátky provozuschopnosti. V této fázi se klade zejména důraz na pracovní postup implementace. Hlavním cílem je představit odladěný produkt, který je dostatečně stabilní a připravený k nasazení u investora. [14] [15];
- **Zavedení** – nasazení produktu do uživatelského prostředí. Fáze zavedení začíná ve chvíli, kdy je dokončeno testování a produkt je připraven k nasazení. Cílem této fáze je oprava chyb, příprava uživatelského prostředí, v kterém bude systém nasazen, tvorba manuálů a dokumentací, konzultace s uživateli a koncová revize projektu.

Každá fáze, jak již bylo zmíněno, se skládá z jedné nebo více iterací. Každá iterace má pět hlavních pracovních procesů, jimiž jsou požadavky, analýza, návrh, implementace a testování. [14] [15]



Obrázek 6: Metodika UP pěti hlavních aktivit.

Zdroj: vlastní

Požadavky

V rámci začátku každé iterace je nutné stanovit základní požadavky, tedy stanovit, čeho chceme vlastně dosáhnout a co je po systému požadováno. V praxi se tento krok často zanedbává, což vede ke vzniku zbytečných problémů při vývoji systému. Je nutné stanovit, co přesně od systému očekáváme, jaké omezení mu stanovuje prostředí, v kterém bude nasazen, a kdo a za jakých podmínek s ním bude moci pracovat. [14] [15]

Lze předpokládat, že se systémem bude pracovat více než jeden uživatel, proto je nutné určit jednotlivé typy uživatelů neboli aktérů. Po stanovení aktérů je nutné specifikovat, k čemu konkrétně který aktér může systém využívat a za jakých podmínek. K tomuto slouží takzvané případy užití. Také je nutné stanovit, jakou prioritou budou mít požadavky jednotlivých aktérů. [14] [15]

V praxi bychom danou problematiku mohli přirovnat k fungování například nemocnice. Pokud budeme považovat nemocnici za systém, pak jsou aktéry například doktoři a pacienti. Pacienty dále můžeme ještě rozdělit na akutní a běžné případy. Všichni aktéři mohou nemocnici, neboli systém využívat, ale každý za jiným účelem a s jinou prioritou. [14] [15]

Z pohledu priority bude mít akutní pacient přednost před běžným pacientem, a to buď absolutní, nebo bude na jeho ošetření uvolněno méně zdrojů. Z pohledu možnosti využití pak bude moci doktor využívat například operační sál k provádění operací, což oba zbylí aktéři nebudou moci. [14] [15]

V rámci práv mohou však všichni aktéři využít například funkci nechat se operovat. Tato varianta se dá zajistit pomocí dědičnosti, kdy doktor a akutní pacient dědí všechna práva od běžného pacienta. [14] [15]

Požadavky na systém můžeme dělit do různých skupin, nejčastěji se pak používá dělení na funkční a nefunkční. [14] [15]

- Funkční požadavky popisují, co by měl systém dělat jak obecně, tak pro konkrétní aktéry;
- Nefunkční požadavky popisují omezení daného systému.

Špatná specifikace nebo zanedbání sběru požadavků může v budoucnu vést k výraznému prodloužení vývoje systému a jeho následnému prodražení. [14] [15]

Analýza

Po stanovení požadavků na systém přichází čas na jejich analýzu. Během analýzy je tvořen takzvaný analytický model. Tento model se zabývá tím, co systém musí udělat, ne však jakým způsobem. Toto je přenecháno části návrh. [14] [15]

V rámci analýzy jsou tvořeny různé diagramy, které pomáhají pochopit principy a požadavky na systém.

- Analytické třídy

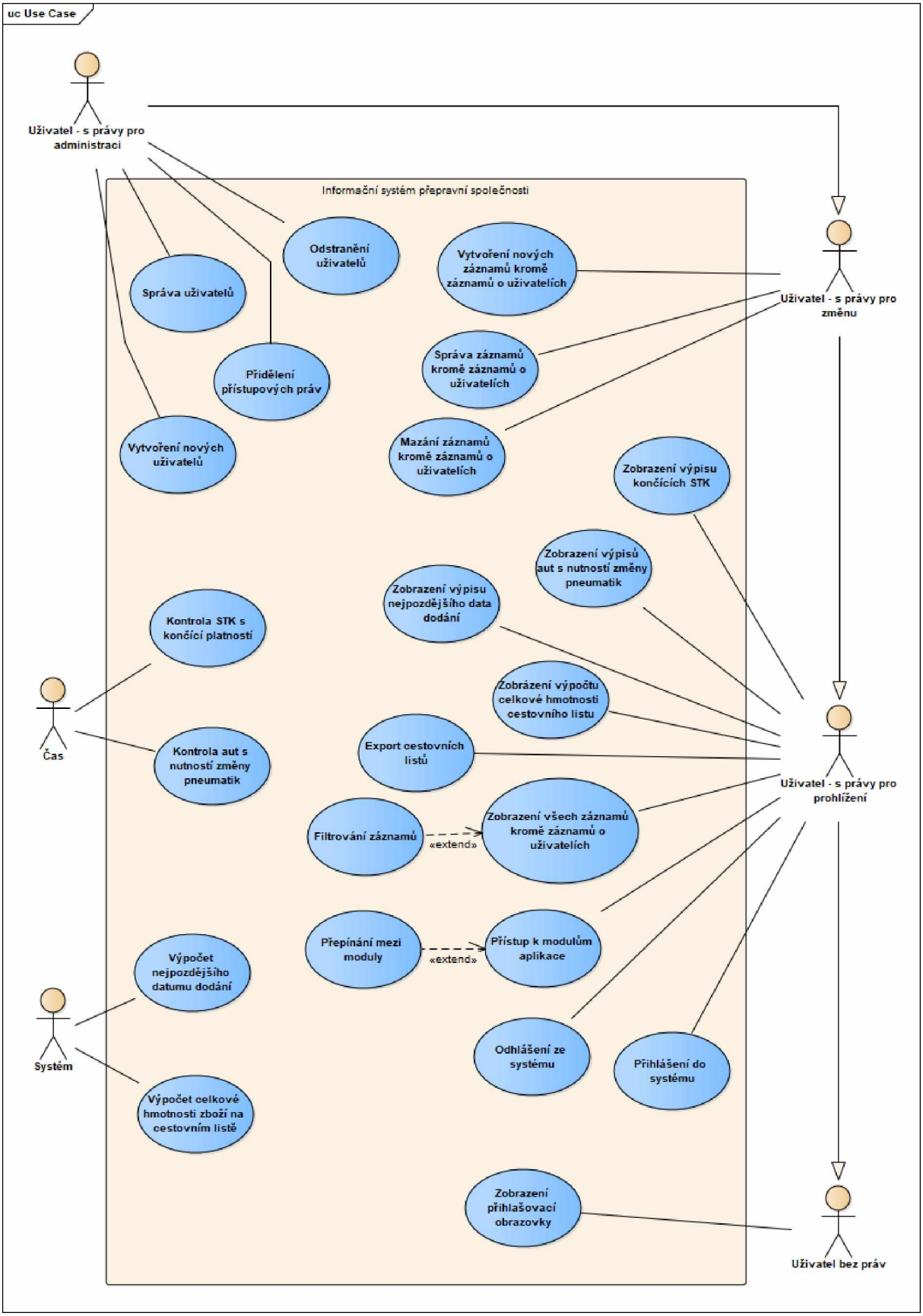
V rámci tohoto diagramu jsou zobrazeny jednotlivé třídy s jejich atributy a vazbami. Třídy by měly obsahovat pouze nejdůležitější atributy, které budou pravděpodobně použity i u tříd navrhovaného modelu. Třídy také musí obsahovat operace specifikující klíčové služby, které budou poskytovat. [14] [15]

K hledání tříd je využívána metoda analýzy podstatných jmen a sloves a metoda štítků CRC;

- Diagram případů užití (Use Case Diagram)

Diagram případů užití slouží k zachycení chování systému vůči jednotlivým aktérům. Aktér je v podstatě entita, která využívá dané funkcionality systému. Aktérem může být nejenom uživatel, ale i například jiný systém nebo čas. [14] [15]

Tímto způsobem je tedy možno znázornit funkční požadavky vůči jednotlivým uživatelům systému. V rámci diagramu není stanoveno, jak konkrétně bude systém danou funkcionalitu provádět, je pouze uvedeno, co umí a k čemu je jí možno využít;

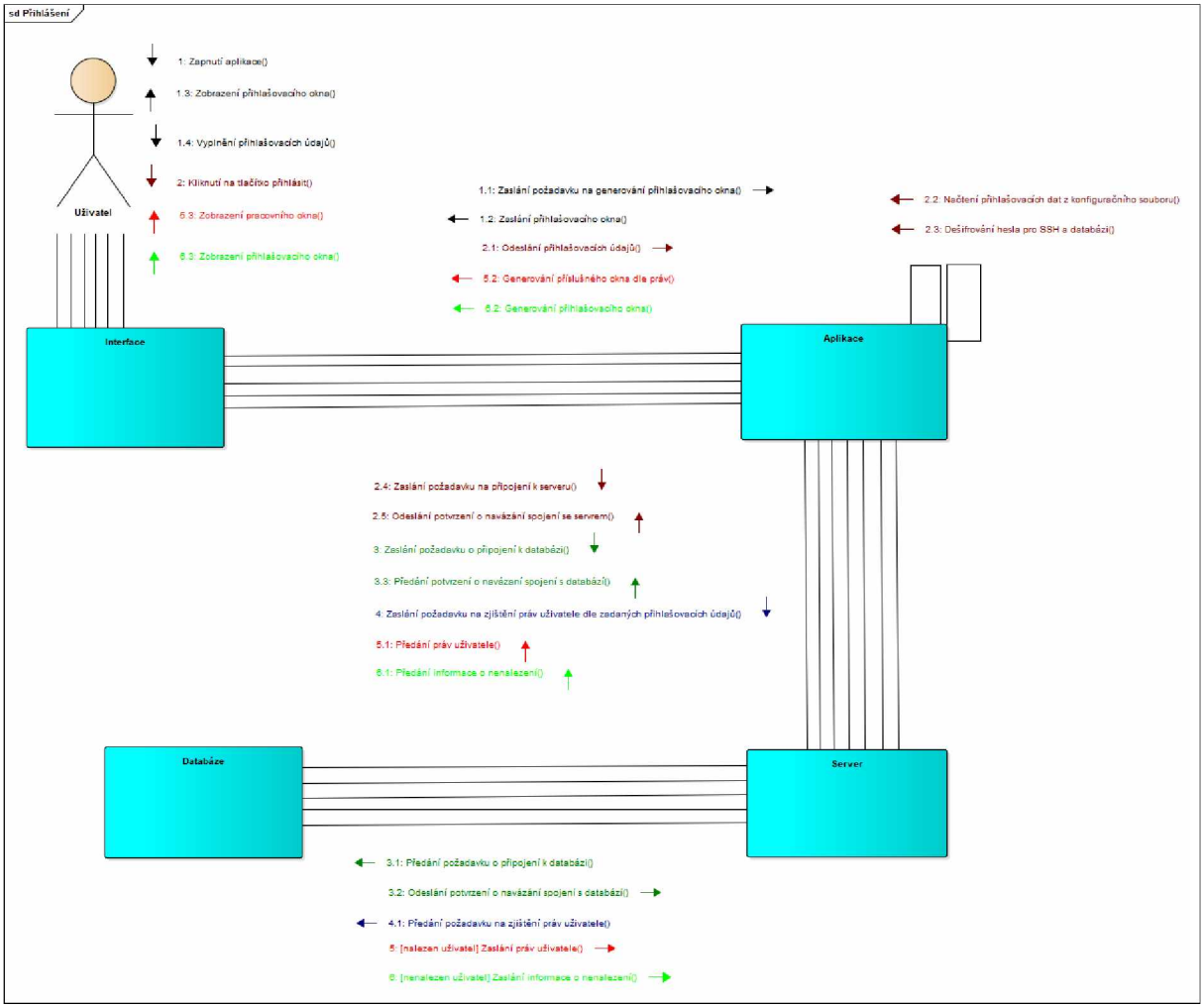


Obrázek 7: Diagram případů užití

Zdroj: vlastní

- Komunikační diagram

Jedná se o diagram iterací, což znamená, že zachycuje předávání zpráv mezi jednotlivými objekty systému a vazby mezi nimi. Na rozdíl od sekvenčního diagramu popsaného níže zde objekty nemají čáry života a jsou vzájemně propojeny pomocí linek. Objektem může být například i uživatel, a tedy všechny objekty nemusí být součástí systému. V takovémto případě je komunikace na diagramu znázorněna tak, že uživatel komunikuje s uživatelským rozhraním, a toto rozhraní pak přeposílá zprávu, neboli požadavek, příslušné aplikaci; [14] [15]



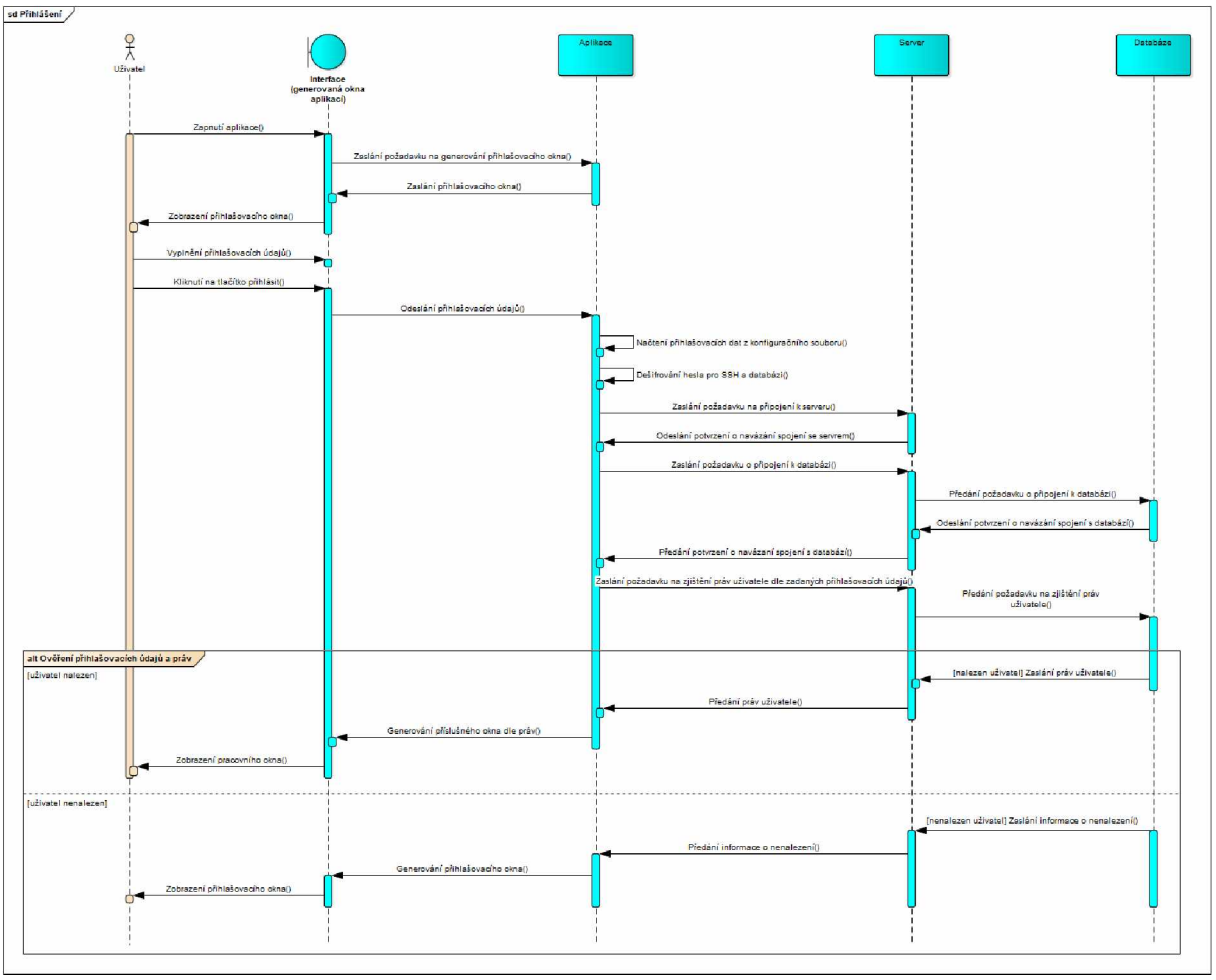
Obrázek 8: Komunikační diagram – přihlášení

Zdroj: vlastní

Vzhledem k velkému rozsahu je diagram hůře čitelný a v plné velikosti je uložen v příloze v elektronické podobě.

- Sekvenční diagram

Jedná se o nejpoužívanější diagram iterací, který znázorňuje komunikaci mezi objekty v rámci časové posloupnosti. Nejčastěji zobrazuje spolupráci objektů v rámci jednoho případu užití; [14] [15]



Obrázek 9: Sekvenční diagram – přihlášení

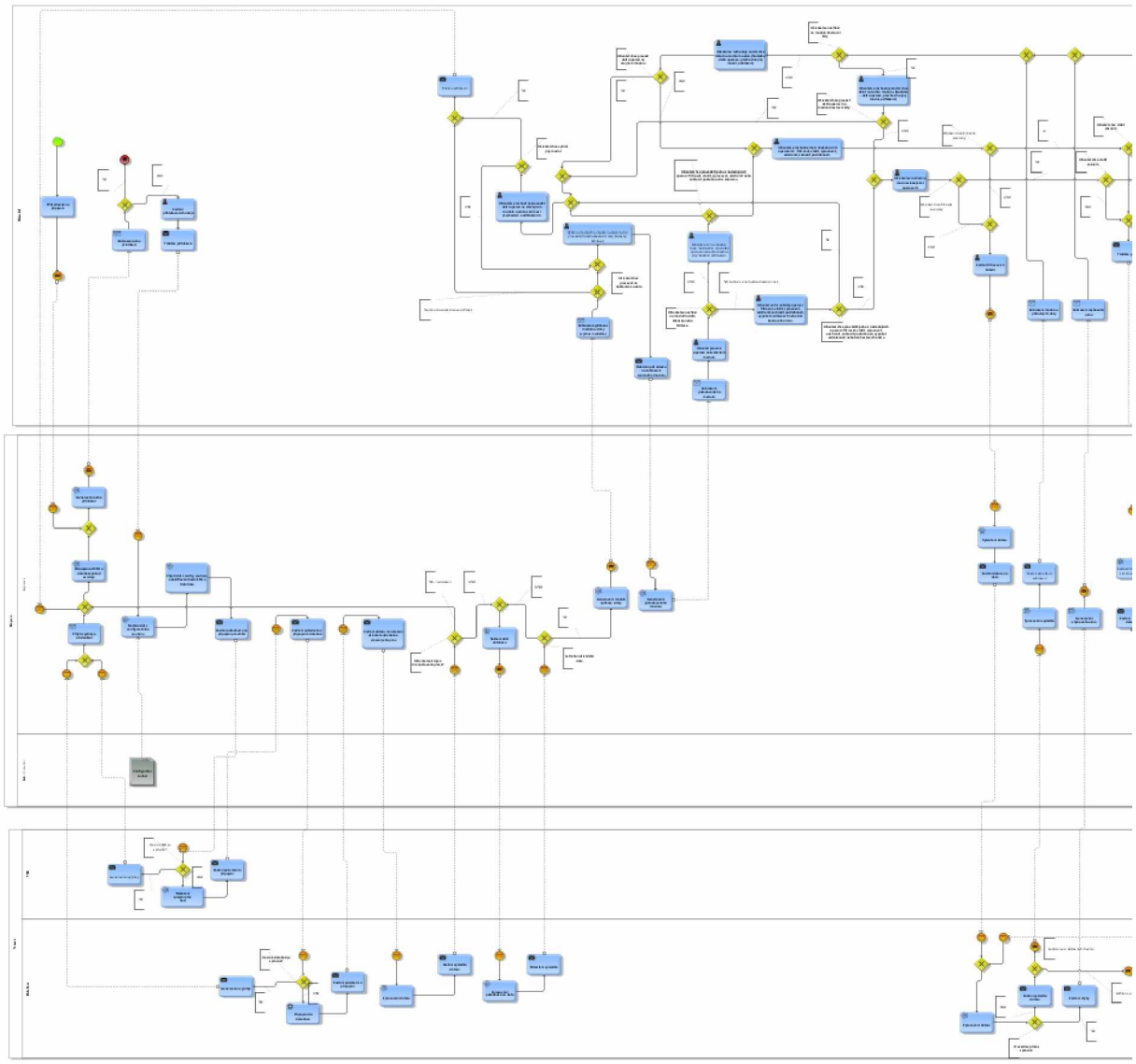
Zdroj: vlastní

Vzhledem k velkému rozsahu je diagram hůře čitelný a v plné velikosti je uložen v příloze v elektronické podobě.

- BPMN diagram

Diagram BPMN slouží k modelování procesů pomocí procesních diagramů. V rámci toho je sledována cesta každého požadavku za účelem znázornění, jak a čím bude obsluhován. V rámci větší přehlednosti se systém rozděluje na takzvané swimlanes neboli plavecké dráhy. Jsou to vlastně obdélníkové rámečky, které slouží k seskupování modelovacích prvků systému. Existují dva typy plaveckých drah, a to bazény a dráhy. Bazény představují

jednotlivé účastníky procesu jako je například uživatel, aplikace nebo server. Dráhy jsou pak podmnožinou bazénu. Slouží k tomu, aby bylo možné uspořádat aktivity v rámci bazénu na základě rolí a funkcí. Příkladem může být rozdělení bazénu „uživatel“ na dráhy „administrátor“ a „zaměstnanec“. [16]



Obrázek 10: Ukázka části BPMN diagramu

Zdroj: vlastní

Vzhledem k velkému rozsahu je celý BPMN diagram uložen v příloze v elektronické podobě.

Návrh

Během této fáze dochází k přesnému určení jak, kde a s jakými datovými strukturami budou implementovány jednotlivé funkce. Vychází se přitom z analytického modelu, kde byly tyto funkce specifikovány. [14] [15]

Implementace

V rámci implementace probíhá zejména tvorba samotného kódu. Jedná se v podstatě o transformaci dříve navrženého modelu do spustitelné podoby. [14] [15]

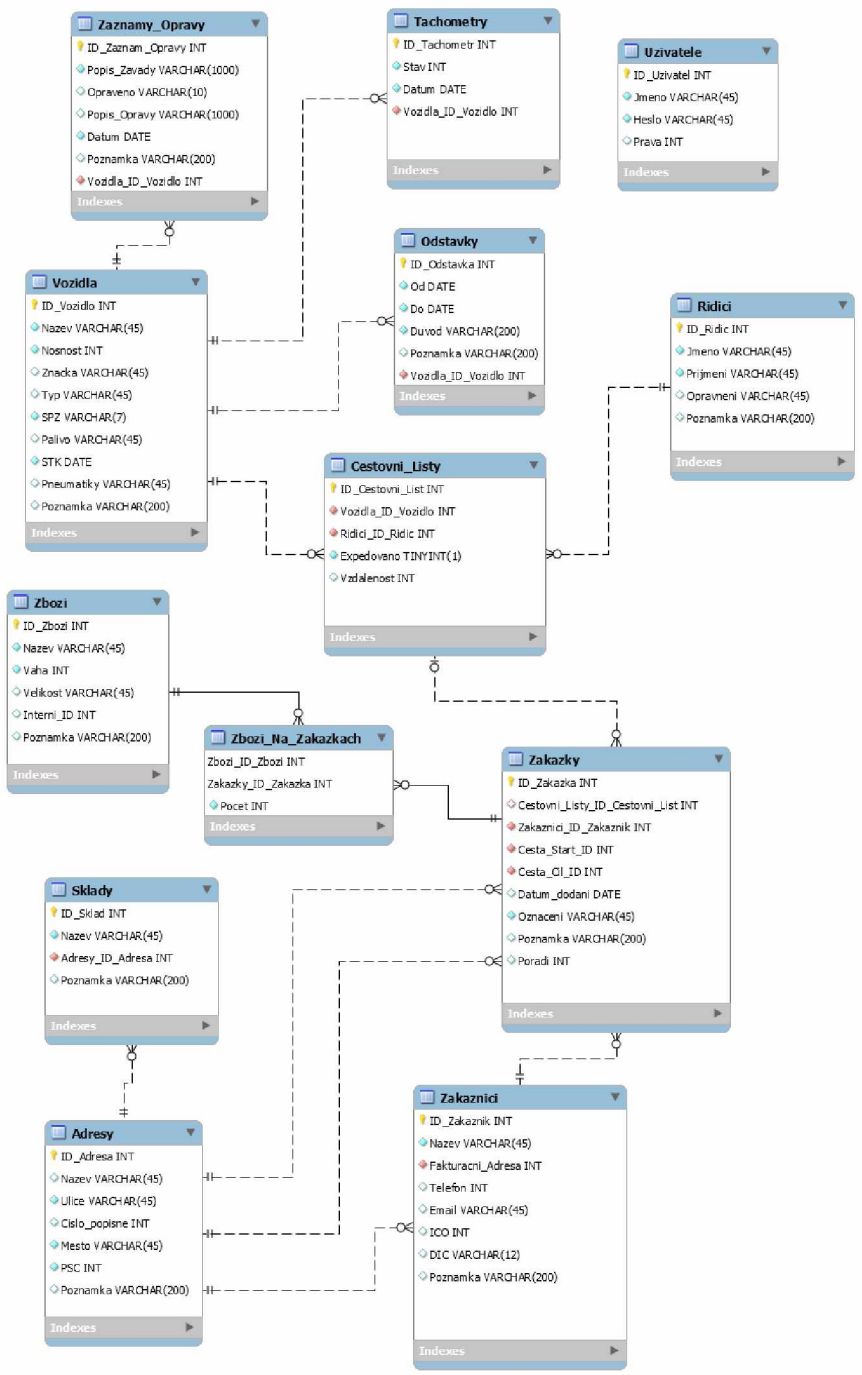
Testování

Posledním krokem je samotné testování. Testování probíhá ve chvíli, kdy je vytvořen spustitelný kód, a tedy po implementační části. Cílem je ověřit, zda systém splňuje všechny požadavky na něj kladené, zda je realizuje správně, bezpečně a efektivně. [14] [15]

2. PRAKTICKÁ ČÁST

2.1 Datový model

Model byl vytvořen pomocí prostředí MySQL Workbent s využitím SQL kódu na základě souhrnných dat z teoretické části diplomové práce.



Obrázek 11: Datový model

Zdroj: vlastní

2.1.1 Tabulka cestovních listů

Cestovni_Listy			
Název	Datový typ	Typ klíče	Ostatní omezení
ID_Cestovni_List	INT	Primární klíč	Index
Vozidla_ID_Vozidlo	INT	Cizí klíč	Index
Ridici_ID_Ridic	INT	Cizí klíč	Index
Expedovano	TINYINT(1)		
Vzdalenost	INT		NULL

Tabulka 2: Tabulka Cestovni_Listy

Zdroj: vlastní

Tabulka Cestovni_Listy stojí uprostřed celého modelu a jedná se o nejdůležitější tabulku, která propojuje a dává smysl celému modelu. V reálné aplikaci je použita k propojení určitého řidiče a auta se zakázkami od zákazníků. Uchovává kromě svého ID[1] také ID vozidla a řidiče. Oba atributy jsou použity jako cizí klíče k tabulkám Vozidla a Ridici a jsou povinné pro vytvoření cestovního listu.

Dále uchovává záznam, zda byla provedena expedice a o vzdálenosti cesty. Expedice je povinný údaj a je nastaven na datový typ TINYINT(1), což znamená, že databáze vrací pro aplikaci TRUE nebo FALSE. Vzdálenost cesty je obyčejný sloupec uchovávající hodnotu INT a nemusí být vyplněn. Svůj pravý význam získává až v prostředí aplikace, kde je využíván pro uložení skutečné vzdálenosti všech zakázek na cestovním listu pomocí Google Maps API.

ID cestovního listu je, jak již vyplývá z výše uvedeného, využito v příslušných záznamech tabulky Zakazky jako cizí klíč.

Tabulka Cestovni_Listy má vztah k tabulkám Vozidla, Ridici a Zakazky.

Cestovni_Listy – Vozidla

- Jednomu řádku v tabulce Cestovni_Listy odpovídá právě jeden řádek tabulky Vozidla;
- Jednomu řádku v tabulce Vozidla může odpovídat 0 nebo více řádků tabulky Cestovni_Listy.

[1] ID (Identification)

Cestovni_Listy – Ridici

- Jednomu řádku v tabulce Cestovni_Listy odpovídá právě jeden řádek tabulky Ridici;
- Jednomu řádku v tabulce Ridici může odpovídat 0 nebo více řádků tabulky Cestovni_Listy.

Cestovni_Listy – Zakazky

- Jednomu řádku v tabulce Cestovni_Listy může odpovídat 0 nebo více řádků tabulky Zakazky;
- Jednomu řádku v tabulce Zakazky může odpovídat 0 nebo 1 řádek tabulky Cestovni_Listy.

2.1.2 Tabulka zakázek

Zakazky			
Název	Datový typ	Typ klíče	Ostatní omezení
ID_Zakazka	INT	Primární klíč	Index
Cestovni_Listy_ID_Cestovni_List	INT	Cizí klíč	Index, NULL
Zakaznici_ID_Zakaznik	INT	Cizí klíč	Index
Cesta_Start_ID	INT	Cizí klíč	Index
Cesta_Cil_ID	INT	Cizí klíč	Index
Datum_dodani	DATE		NULL
Oznaceni	VARCHAR(45)		
Poznamka	VARCHAR(200)		NULL
Poradi	INT		NULL

Tabulka 3: Tabulka Zakazky

Zdroj: vlastní

Tabulka Zakazky v sobě uchovává záznamy o zakázkách jednotlivých zákazníků včetně všech potřebných parametrů pro správný chod systému.

Sloupec Cestovni_Listy_ID_Cestovni_List uchovává propojení k příslušnému cestovnímu listu a jedná se o cizí klíč. Sloupec může nabývat nulových hodnot, protože v době vytvoření není v rámci aplikace žádoucí, aby zakázka byla přiřazena k určitému cestovnímu listu.

Tabulka také ukládá ID zákazníka a ID počáteční a konečné adresy cesty. Tyto sloupce jsou cizími klíči tabulek Zakaznici a Adresy. Všechny tři sloupce jsou povinné, a tedy nemohou nabývat nulových hodnot.

Dále je zde uložen nejpozdější datum dodání celé zakázky ve sloupci Datum_dodani. Tento sloupec nabývá datového typu DATE ve formátu rrrr-mm-dd.

Sloupec Oznaceni a Poznamka datového typu VARCHAR(45) a VARCHAR(200) slouží k uložení dodatečných informací o zásilce. V rámci aplikace je sloupec Oznaceni používán pro uložení například čísla objednávky nebo jiného označení, podle kterého se dá daná zakázka identifikovat, a nesmí být proto nulový. Sloupec Poznamka pak slouží jako volně využitelný záznam pro dodatečné informace, a tedy může nabývat i nulových hodnot.

Posledním sloupcem tabulky je Poradi. Tento sloupec slouží k identifikaci pořadí zakázky v rámci cestovního listu. Díky tomu je možné v rámci aplikace vytvářet cestovní plán rozvozu a vypočítat celkovou vzdálenost cesty. V aplikaci je záznam o pořadí přidělen až po přidání zakázky do cestovního listu, a proto sloupec může nabývat i nulových hodnot.

ID zakázky je pak využito v propojující tabulce Zbozi_Na_Zakazkach, která řeší vazbu M:N mezi tabulkou Zakazky a Zbozi.

Tabulka Zakazky má vztah k tabulkám Cestovni_Listy, Zbozi_Na_Zakazkach, Adresy a Zakaznici.

Zakazky – Cestovni_Listy

- Jednomu řádku v tabulce Zakazky může odpovídat 0 nebo 1 řádek tabulky Cestovni_Listy;
- Jednomu řádku v tabulce Cestovni_Listy může odpovídat 0 nebo více řádků tabulky Zakazky.

Zakazky – Zbozi_Na_Zakazkach

- Jednomu řádku v tabulce Zakazky může odpovídat 0 nebo více řádků tabulky Zbozi_Na_Zakazkach;
- Jednomu řádku v tabulce Zbozi_Na_Zakazkach odpovídá právě jeden řádek tabulky Zakazky.

Zakazky – Adresy

- Jednomu řádku v tabulce Zakazky odpovídá právě jeden řádek tabulky Adresy;
- Jednomu řádku v tabulce Adresy může odpovídat 0 nebo více řádků tabulky Zakazky.

Zakazky – Zakaznici

- Jednomu řádku v tabulce Zakazky odpovídá právě jeden řádek tabulky Zakaznici;
- Jednomu řádku v tabulce Zakaznici může odpovídat 0 nebo více řádků tabulky Zakazky.

2.1.3 Tabulka zboží na zakázce

Zbozi_Na_Zakazkach			
Název	Datový typ	Typ klíče	Ostatní omezení
Zbozi_ID_Zbozi	INT	Primární klíč, Cizí klíč	Index
Zakazky_ID_Zakazka	INT	Primární klíč, Cizí klíč	Index
Pocet	INT		

Tabulka 4: Tabulka Zbozi_Na_Zakazkach

Zdroj: vlastní

Jedná se o pomocnou spojovací tabulku mezi tabulkami Zbozi a Zakazka. Primární klíč je složený z cizích klíčů těchto tabulek. Dalším sloupcem tabulky je pak Pocet. Sloupec Pocet je datového typu INT a slouží k udržení informace o počtu kusů jednotlivého zboží na zakázce. Jedná se o povinný údaj.

Tabulka Zbozi_Na_Zakazkach má vztah k tabulkám Zakazky a Zbozi.

Zbozi_Na_Zakazkach – Zakazky

- Jednomu řádku v tabulce Zbozi_Na_Zakazkach odpovídá právě jeden řádek tabulky Zakazky;
- Jednomu řádku v tabulce Zakazky může odpovídat 0 nebo více řádků tabulky Zbozi_Na_Zakazkach.

Zbozi_Na_Zakazkach – Zbozi

- Jednomu řádku v tabulce Zbozi_Na_Zakazkach odpovídá právě jeden řádek tabulky Zbozi;
- Jednomu řádku v tabulce Zbozi může odpovídat 0 nebo více řádků tabulky Zbozi_Na_Zakazkach.

2.1.4 Tabulka zboží

Zbozi			
Název	Datový typ	Typ klíče	Ostatní omezení
ID_Zbozi	INT	Primární klíč	Index
Nazev	VARCHAR(45)		
Vaha	INT		
Velikost	VARCHAR(45)		NULL
Interni_ID	INT		NULL
Poznamka	VARCHAR(200)		NULL

Tabulka 5: Tabulka Zbozi

Zdroj: vlastní

Tabulka Zbozi uchovává informace o skladových položkách. Povinnými údaji jsou sloupce Nazev VARCHAR(45) a Vaha INT. Nepovinnými pak sloupce Velikost VARCHAR(45), Interni_ID INT a Poznamka VARCHAR(200).

V rámci aplikace je zamýšleno do sloupce Vaha ukládat skutečnou váhu zboží v kilogramech, která je pak využita k výpočtu celkové váhy zakázek na cestovním listě tak, aby měl uživatel přehled, zda například nepřetíží zvolený dopravní prostředek. Ve sloupci Velikost je pak zamýšleno uchovávat záznam o velikosti balení dle obalu výrobku nebo specifikací výrobce. Z tohoto důvodu byl použit datový typ VARCHAR, aby mohl být uložen libovolný tvar například „100 ml“, „15 kg“ nebo „15+3 kg“.

Sloupec Interni_ID slouží k internímu označení zboží. V rámci již používaného pokladního systému zákazník používá vlastní identifikační číslo u každého zboží. Toto interní ID není spojeno s ID zboží z toho důvodu, že každá provozovna má vlastní nezávislý systém číslování, a proto je žádoucí, aby se jednalo pouze o pomocný údaj.

Sloupec Poznamka pak slouží jako u všech tabulek k doplnění libovolných informací o záznamu. U zboží to většinou bývá distributor, poměr tužení u dvousložkových barev nebo odkaz na bezpečnostní a technické listy výrobku.

ID zboží je využito v propojující tabulce Zbozi_Na_Zakazkach, která řeší vazbu M:N mezi tabulkou Zbozi a Zakazky.

Tabulka Zbozi má vztah k tabulce Zbozi_Na_Zakazkach.

Zbozi – Zbozi_Na_Zakazkach

- Jednomu řádku v tabulce Zbozi může odpovídat 0 nebo více řádků tabulky Zbozi_Na_Zakazkach;
- Jednomu řádku v tabulce Zbozi_Na_Zakazkach odpovídá právě jeden řádek tabulky Zbozi.

2.1.5 Tabulka zákazníků

Zakaznici			
Název	Datový typ	Typ klíče	Ostatní omezení
ID_Zakaznik	INT	Primární klíč	Index
Nazev	VARCHAR(45)		
Fakturacni_Adresa	INT	Cizí klíč	Index
Telefon	INT		NULL
Email	VARCHAR(45)		NULL
ICO	INT		NULL
DIC	VARCHAR(12)		NULL
Poznamka	VARCHAR(200)		NULL

Tabulka 6: Tabulka Zakaznici

Zdroj: vlastní

Tabulka Zakaznici slouží k uchovávání záznamů o jednotlivých zákaznících.

Jednotlivé názvy sloupců tabulky jsou samy o sobě dostatečně vypovídající, a proto není nutné je zvláště rozebírat. Jediný povinný údaj kromě primárního a cizího klíče je sloupec Nazev. Zde se předpokládá uložení názvu společnosti nebo živnostenského oprávnění,

a to ve stejném tvaru, jaký je uveden na výstupu informačního systému ARES Ministerstva financí. Cizí klíč Fakturacni_Adresa udržuje vazbu na tabulku Adresy.

ID zákazníka je využito v tabulce Zakazky jako povinný cizí klíč.

Tabulka Zakaznici má vztah k tabulkám Zakazky a Adresy.

Zakaznici – Zakazky

- Jednomu řádku v tabulce Zakaznici může odpovídat 0 nebo více řádků tabulky Zakazky;
- Jednomu řádku v tabulce Zakazky odpovídá právě jeden řádek tabulky Zakaznici.

Zakaznici – Adresy

- Jednomu řádku v tabulce Zakaznici odpovídá právě jeden řádek tabulky Adresy;
- Jednomu řádku v tabulce Adresy může odpovídat 0 nebo více řádků tabulky Zakaznici.

2.1.6 Tabulka adres

Adresy			
Název	Datový typ	Typ klíče	Ostatní omezení
ID_Adresa	INT	Primární klíč	Index
Nazev	VARCHAR(45)		NULL
Ulice	VARCHAR(45)		
Cislo_popisne	INT		NULL
Mesto	VARCHAR(45)		
PSC	INT		
Poznamka	VARCHAR(200)		NULL

Tabulka 7: Tabulka Adresy

Zdroj: vlastní

Tabulka Adresy slouží k uchovávání záznamů o adresách skladů, zákazníků, počátečního a koncového místa zakázek.

Jednotlivé názvy sloupců jsou samy o sobě dostatečně vypovídající, a proto není nutné je blíže popisovat. Povinnými sloupci jsou kromě primárního klíče sloupce Nazev,

Cislo_popisne a Poznamka. Nejdůležitějším sloupcem pro samotnou aplikaci je pak sloupec Mesto. Pomocí tohoto záznamu vypočítává aplikace prostřednictvím Google Maps API vzdálenost mezi městy na přepravním listu.

ID tabulky Adresy je využito jako cizí klíč v tabulkách Zakazky, Zakaznici, Sklady.

Tabulka Adresy má vztah k tabulkám Zakazky, Zakaznici, Sklady.

Adresy – Zakazky

- Jednomu řádku v tabulce Adresy může odpovídat 0 nebo více řádků tabulky Zakazky;
- Jednomu řádku v tabulce Zakazky odpovídá právě jeden řádek tabulky Adresy.

Adresy – Zakaznici

- Jednomu řádku v tabulce Adresy může odpovídat 0 nebo více řádků tabulky Zakaznici;
- Jednomu řádku v tabulce Zakaznici odpovídá právě jeden řádek tabulky Adresy.

Adresy – Sklady

- Jednomu řádku v tabulce Adresy může odpovídat 0 nebo více řádků tabulky Sklady;
- Jednomu řádku v tabulce Sklady odpovídá právě jeden řádek tabulky Adresy.

2.1.7 Tabulka skladů

Sklady			
Název	Datový typ	Typ klíče	Ostatní omezení
ID_Sklad	INT	Primární klíč	Index
Nazev	VARCHAR(45)		
Adresy_ID_Adresa	INT	Cizí klíč	Index
Poznamka	VARCHAR(200)		NULL

Tabulka 8: Tabulka Sklady

Zdroj: vlastní

Tabulka Sklady slouží k uchovávání záznamů o skladech.

Tabulka je složena pouze z primárního klíče, názvu skladu, ID adresy a poznámky. Stejně pojmenované sloupce byly již vysvětleny v předchozích tabulkách, a proto zde nebudou

podrobně popisovány. Tabulka Sklady je spíše pomocnou tabulkou a v aplikaci slouží jen pro uložení názvů a adres skladů/provozoven s možností popisu, například zda má sklad/provozovna vlastní rampu pro vykládku a nakládku.

Tabulka Sklady má vztah k tabulce Adresy.

Sklady – Adresy

- Jednomu řádku v tabulce Sklady odpovídá právě jeden řádek tabulky Adresy;
- Jednomu řádku v tabulce Adresy může odpovídat 0 nebo více řádků tabulky Sklady.

2.1.8 Tabulka řidičů

Ridici			
Název	Datový typ	Typ klíče	Ostatní omezení
ID_Ridic	INT	Primární klíč	Index
Jmeno	VARCHAR(45)		
Prijmeni	VARCHAR(45)		
Opravneni	VARCHAR(45)		NULL
Poznamka	VARCHAR(200)		NULL

Tabulka 9: Tabulka Ridici

Zdroj: vlastní

Tabulka Ridici uchovává jméno, příjmení, řidičské oprávnění a poznámku řidiče. Její povinné sloupce kromě primárního klíče jsou sloupce Jmeno a Prijmeni.

ID řidiče je využito v tabulce Cestovni_Listy jako povinný cizí klíč.

Tabulka Ridici má vztah k tabulce Cestovni_Listy.

Ridici – Cestovni_Listy

- Jednomu řádku v tabulce Ridici může odpovídat 0 nebo více řádků tabulky Cestovni_Listy;
- Jednomu řádku v tabulce Cestovni_Listy odpovídá právě jeden řádek tabulky Ridici.

2.1.9 Tabulka vozidel

Vozidla			
Název	Datový typ	Typ klíče	Ostatní omezení
ID_Vozidlo	INT	Primární klíč	Index
Nazev	VARCHAR(45)		
Nosnost	INT		
Znacka	VARCHAR(45)		NULL
Typ	VARCHAR(45)		NULL
SPZ	VARCHAR(7)		
Palivo	VARCHAR(45)		NULL
STK	DATE		
Pneumatiky	VARCHAR(45)		NULL
Poznamka	VARCHAR(200)		NULL

Tabulka 10: Tabulka Vozidla

Zdroj: vlastní

Tabulka Vozidla udržuje informaci o vozidlech vozového parku.

Ke své práci využívá v rámci aplikace trojici pomocných tabulek Zaznamy_Opravy, Tachometry a Odstavky. Jednotlivé názvy sloupců jsou samy o sobě dostatečně vypovídající, a proto se zde práce bude věnovat jen těm nejzajímavějším v rámci samotné aplikace.

Sloupec STK uchovává datum konce platnosti STK. V rámci aplikace je pomocí SQL příkazu nastaveno upozornění na končící platnost této kontroly a to 30 dnů dopředu:

```
SELECT ID_Vozidlo, Nazev, STK FROM Vozidla WHERE STK BETWEEN
CURDATE() AND DATE_ADD(CURDATE(), INTERVAL 30 DAY);
```

Sloupec Pneumatiky uchovává informaci o aktuálním typu pneumatik na vozidle. I zde aplikace upozorňuje na potřebu změny například letních pneumatik na zimní.

Nepovinné sloupce jsou Znacka (myšleno značka vozu např. RAV4), Typ (např. dodávka), SPZ a STK.

ID vozidla je využito v tabulkách Zaznamy_Opravy, Tachometry, Odstavky, Cestovni_Listy jako povinný cizí klíč.

Tabulka Vozidla má vztah k tabulkám Zaznamy_Opravy, Tachometry, Odstavky, Cestovni_Listy.

Vozidla – Zaznamy_Opravy

- Jednomu řádku v tabulce Vozidla může odpovídat 0 nebo více řádků tabulky Zaznamy_Opravy;
- Jednomu řádku v tabulce Zaznamy_Opravy odpovídá právě jeden řádek tabulky Vozidla.

Vozidla – Tachometry

- Jednomu řádku v tabulce Vozidla může odpovídat 0 nebo více řádků tabulky Tachometry;
- Jednomu řádku v tabulce Tachometry odpovídá právě jeden řádek tabulky Vozidla.

Vozidla – Odstavky

- Jednomu řádku v tabulce Vozidla může odpovídat 0 nebo více řádků tabulky Odstavky;
- Jednomu řádku v tabulce Odstavky odpovídá právě jeden řádek tabulky Vozidla.

Vozidla – Cestovni_Listy

- Jednomu řádku v tabulce Vozidla může odpovídat 0 nebo více řádků tabulky Cestovni_Listy;
- Jednomu řádku v tabulce Cestovni_Listy odpovídá právě jeden řádek tabulky Vozidla.

2.1.10 Tabulka záznamů o opravách

Zaznamy_Opravy			
Název	Datový typ	Typ klíče	Ostatní omezení
ID_Zaznam_Opravy	INT	Primární klíč	Index
Popis_Zavady	VARCHAR(1000)		
Opraveno	VARCHAR(10)		NULL
Popis_Opravy	VARCHAR(1000)		NULL
Datum	DATE		
Poznamka	VARCHAR(200)		NULL
Vozidla_ID_Vozidlo	INT	Cizí klíč	Index

Tabulka 11: Zaznamy_Opravy

Zdroj: vlastní

Tabulka Zaznamy_Opravy je pomocnou tabulkou pro tabulku Vozidla.

Její sloupce jsou složeny kromě primárního klíče také z povinných sloupců Popis_Zavady a Popis_Opravy pro uložení informací o závadě a provedené opravě a ze sloupce Opraveno, kde je očekáváno jednoslovné vyjádření jako například „ano“, „ne“ nebo „částečně“. Sloupec Datum uchovává datum opravy a Poznamka umožňuje uchovávat dodatečné informace, jako například cenu opravy nebo číslo faktury. A také povinného cizího klíče Vozidla_ID_Vozidlo, který zajišťuje propojení s příslušným záznamem v tabulce Vozidla.

Tabulka Zaznam_Opravy má vztah k tabulce Vozidla.

Zaznam_Opravy – Vozidla

- Jednomu řádku v tabulce Zaznam_Opravy odpovídá právě jeden řádek tabulky Vozidla;
- Jednomu řádku v tabulce Vozidla může odpovídat 0 nebo více řádků tabulky Zaznamu_Opravy.

2.1.11 Tabulka záznamů stavu tachometru

Tachometry			
Název	Datový typ	Typ klíče	Ostatní omezení
ID_Tachometr	INT	Primární klíč	Index
Stav	INT		
Datum	DATE		
Vozidla_ID_Vozidlo	INT	Cizí klíč	Index

Tabulka 12: Tabulka Tachometry

Zdroj: vlastní

Tabulka Tachometry je pomocnou tabulkou pro tabulku Vozidla.

Její sloupce jsou složeny kromě primárního klíče také z povinných sloupců Stav, Datum a cizího klíče Vozidla_ID_Vozidla vztaženého k určitému záznamu tabulky Vozidla. Tyto sloupce uchovávají stav najetých kilometrů vůči konkrétnímu datu a vozidlu.

Tabulka Tachometry má vztah k tabulce Vozidla.

Tachometry – Vozidla

- Jednomu řádku v tabulce Tachometry odpovídá právě jeden řádek tabulky Vozidla;
- Jednomu řádku v tabulce Vozidla může odpovídat 0 nebo více řádků tabulky Tachometry.

2.1.12 Tabulka záznamů o odstávce

Odstavky			
Název	Datový typ	Typ klíče	Ostatní omezení
ID_Odstavka	INT	Primární klíč	Index
Od	DATE		
Do	DATE		
Duvod	VARCHAR(200)		
Poznamka	VARCHAR(200)		NULL
Vozidla_ID_Vozidlo	INT	Cizí klíč	Index

Tabulka 13: Tabulka Odstavky

Zdroj: vlastní

Tabulka Odstavky je pomocnou tabulkou pro tabulku Vozidla.

Její sloupce jsou složeny kromě primárního klíče také z povinných sloupců Od, Do, Duvod, povinného cizího klíče Vozidla_ID_Vozidla a nepovinného sloupce Poznamka. Sloupce Od a Do jsou datového typu DATE a uchovávají interval, od kdy do kdy bylo vozidlo odstaveno. Sloupec Duvod slouží pro uchování informací o důvodu odstávky a sloupec Poznamka slouží pro dodatečné informace.

Tabulka Odstavky má vztah k tabulce Vozidla.

Odstavky – Vozidla

- Jednomu řádku v tabulce Odstavky odpovídá právě jeden řádek tabulky Vozidla;
- Jednomu řádku v tabulce Vozidla může odpovídat 0 nebo více řádků tabulky Odstavky.

2.1.13 Tabulka uživatelů

Uzivatele			
Název	Datový typ	Typ klíče	Ostatní omezení
ID_Uzivatel	INT	Primární klíč	Index
Jmeno	VARCHAR(45)	UNIQUE (Jedinečné)	Index
Heslo	VARCHAR(45)		
Prava	INT		NULL

Tabulka 14: Tabulka Uzivatele

Zdroj: vlastní

Tabulka Uzivatele slouží k ukládání přístupových údajů a práv uživatelů aplikace.

Důvodem její existence je skutečnost, že se aplikace připojuje automaticky k databázi pomocí zašifrovaného hesla uloženého v konfiguračním souboru. Aplikace by tedy bez existence této tabulky nemohla rozlišit jednotlivé uživatele a jejich práva k přístupu do aplikace. Zajímavostí této tabulky je sloupec Jmeno, kde byl použit index UNIQUE. To znamená, že se v databázi nemůže vyskytovat uživatel se stejným jménem. Sloupec Prava pro ukládání přístupových práv je sice označen jako nepovinný, ovšem v současné verzi aplikace je vždy právo přiděleno. V případě nulové hodnoty dojde k automatickému odpojení od databáze.

2.2 Uživatelská příručka

Kapitola popisuje všechny důležité funkcionality a ovládání vytvářeného systému z pohledu uživatele. V rámci lepšího pochopení dané problematiky jsou některé části opatřeny i vnitřním vysvětlením chodu této aplikace.

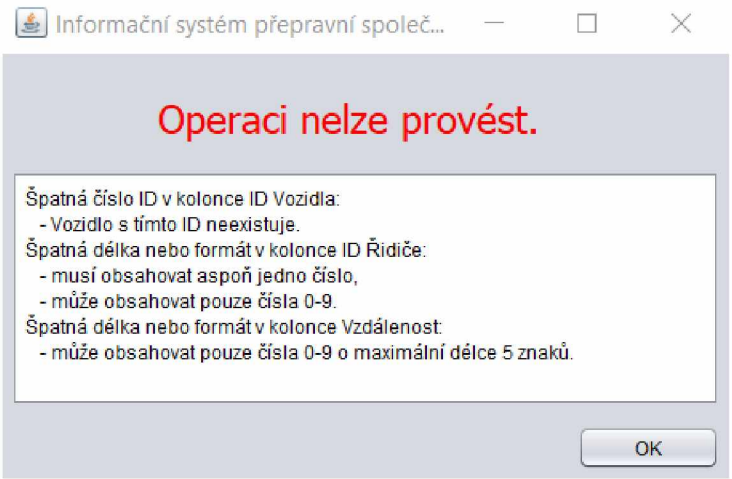
2.2.1 Základní informace o systému/aplikaci

Systém se skládá z databázové části, která byla popsána v předchozí kapitole a aplikační části. Databáze je psána pomocí příkazů SQL a byla otestována na lokální MySQL databázi. Následně po schválení vedoucím diplomové práce byla nasazena na virtuální server od společnosti THINline s.r.o., provozující službu cesky-hosting.cz, kde běží na databázi MariaDB. Aplikace přistupuje k databázi přes SSH tunel.

Celá aplikace se skládá z pěti hlavních modulů:

- Přihlášení;
- Vozidla;
- Cestovní listy;
- Zakázky;
- Editace.

Všechny textové vstupy aplikace jsou ošetřeny, a to většinou pomocí regulárních výrazů. V případě nevhodného vstupního textu je operace přerušena a obsloužena dle potřeby. V aplikaci ve většině případů dojde k vygenerování chybového okna s popisem problému.

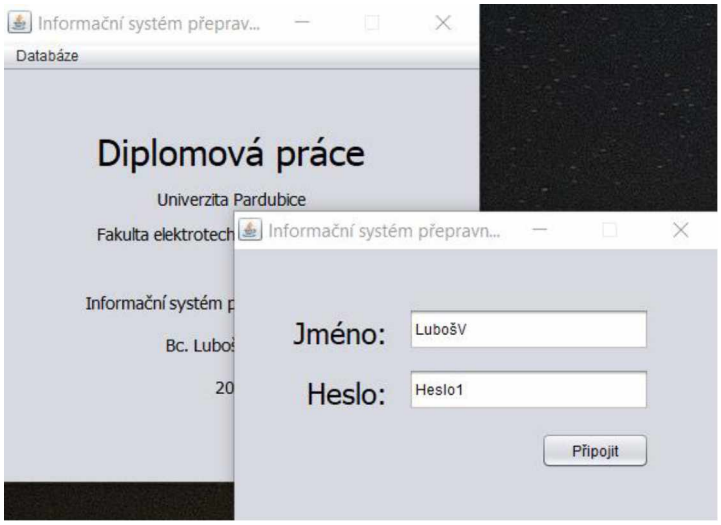


Obrázek 12: Chyba textového vstupu

Zdroj: vlastní

2.2.2 Přihlášení

Po zapnutí aplikace se uživateli zobrazí úvodní obrazovka s rozklikávacím menu v levé horní části. Zde může zvolit varianty „připojení“ nebo „odpojení“ databáze. Po zvolení volby pro připojení se objeví přihlašovací obrazovka pro zadání uživatelského jména a hesla a potvrzovací tlačítko. V případě, že uživatel zadá tyto údaje správně a má příslušná oprávnění, je mu zpřístupněna aplikace. V případě potřeby se uživatel může prostřednictvím stejného rozklikávacího menu odhlásit a znovu přihlásit jako uživatel s jinými právy.



Obrázek 13: Okno přihlášení

Zdroj: vlastní

Aplikace se k databázi připojuje automaticky prostřednictvím konfiguračního souboru, kde jsou uložena všechna důležitá data, jako je adresa databáze, SSH, přihlašovací údaje a podobně. Konfigurační soubor je přístupný uživatelům, a je tedy možné měnit parametry připojení, aniž by bylo nutné vstupovat do kódu aplikace.

Všechny údaje v konfiguračním souboru jsou volně čitelné, až na hesla k SSH a databázi MariaDB. Tyto dva údaje jsou zašifrovány pomocí AES. Aplikace interně uchovává klíč pro zašifrování a skutečné heslo databáze nebo SSH je vlastně heslo z konfiguračního souboru zašifrované aplikací. Jinými slovy – aplikace se vždy automaticky připojí k databázi a následně nahlédne do tabulky s uživateli, porovná přihlašovací údaje a následně provede příslušné operace. Tím je uživateli otevřena aplikace skládající se ze 4 modulů, které jsou postupně popsány v dalších kapitolách.

Ukázka principu získání hesla k databázi z konfiguračního souboru

```
// vytvoření klíče z daného pole bajtů - SecretKeySpec(byte[] key, String algorithm)
Key keyDatabaze = new SecretKeySpec(new byte[]{1, 1, 0, 1, 1,
0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1}, "AES");
// vytvoření instance šifry a její inicializace pro šifrování s typem šifry, módem a výplní
Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
cipher.init(Cipher.ENCRYPT_MODE, keyDatabaze);
//kopírování textu do bajtového pole
byte[] sourceData = vstupniHeslo.getBytes("UTF-8");
//zašifrování textu
byte[] encrypted = cipher.doFinal(sourceData);
//cyklus pro převod zašifrovaného pole bajtů na číslo → číslo se násobí, aby mělo více řádů
for (int i = 0; i < encrypted.length; i++) {
  int cislo = Byte.toUnsignedInt(encrypted[i]);
  cislo = cislo * 12345;
  vystupniHeslo += cislo;
}
return vystupniHeslo;
```

Načtené heslo aplikací z konfiguračního souboru (vstupniHeslo): hesloDatabaze

Vygenerované skutečné heslo pro databázi (vystupniHeslo): 28023150

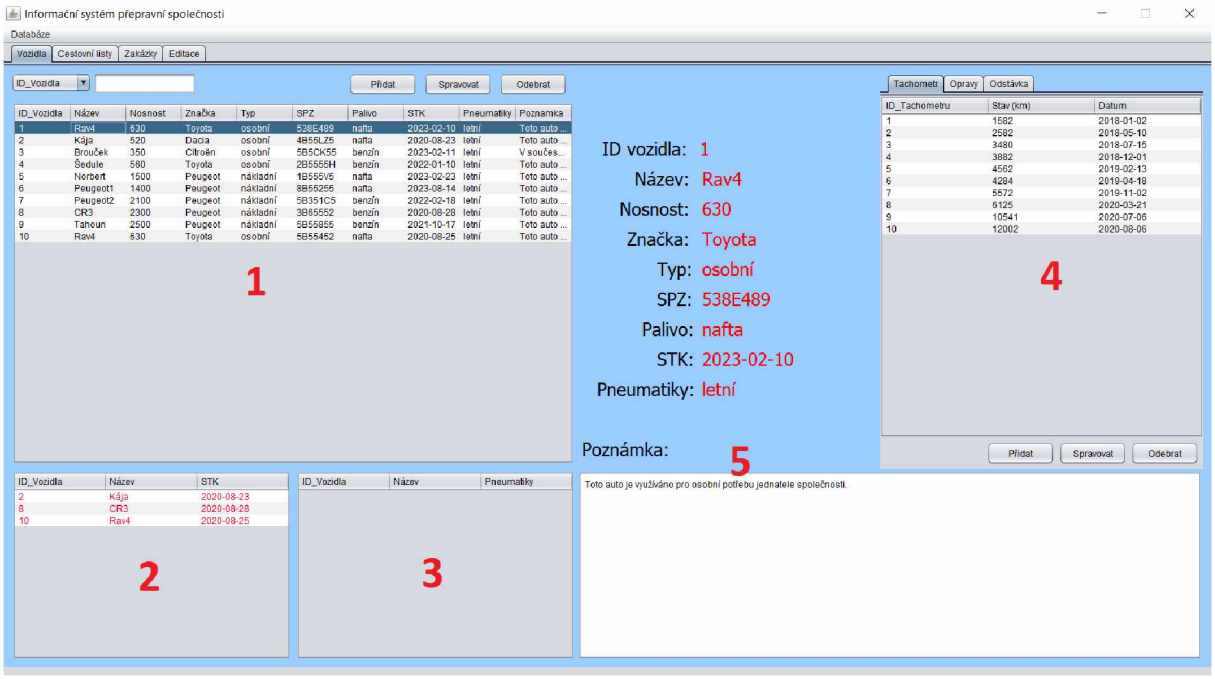
Aplikace rozlišuje 4 základní práva

- Bez práva – v případě, že práva nabývají hodnoty NULL. V tomto případě dojde k odhlášení od databáze;
- Prohlížení – v případě, že práva nabývají hodnoty 0 nebo 1. Uživatel má právo prohlížet záznamy;
- Úprava – v případě, že práva nabývají hodnoty 2 a více. Uživatel má právo prohlížet, upravovat a vytvářet záznamy;
- Správa uživatelů – v případě, že práva nabývají hodnoty 3. Uživatel má právo prohlížet, upravovat, vytvářet záznamy a uživatele.

V případě zadání práv větších než 3 má uživatel práva úpravy, ne však správu uživatelů. Důvodem je připravení aplikace na rozšíření o další práva dle budoucí potřeby.

Z prostředí aplikace je pak možné nastavení práv pouze na hodnoty 1, 2 a 3.

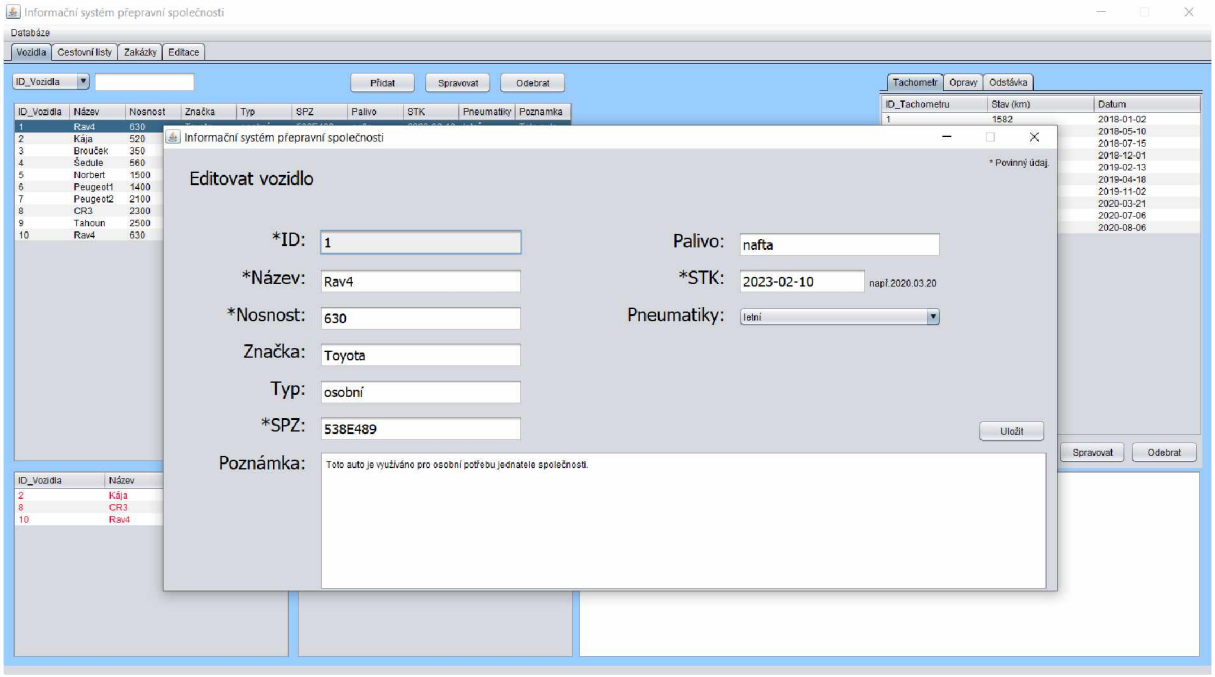
2.2.3 Vozidla



Obrázek 14: Modul Vozidla

Zdroj: vlastní

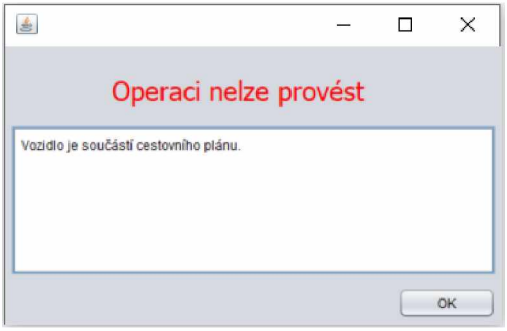
Prvním modulem je modul Vozidla. Zde se uživatel stará o údržbu svého vozového parku. Nachází se zde hlavní tabulka vozidla (1), která je využívána pro výpis vozidel a základních informací o nich. Po kliknutí na záznam tabulky (1) dojde k výpisu údajů do předpřipravené části (5) aplikace. V případě dvojkliku na záznam dojde k vygenerování nového okna s podrobným výpisem. Toto stejné okno se vytvoří i v případě, že uživatel využije dvojici tlačítek v pravém horním rohu pro přidání záznamu nebo jeho správu. Dle zvolené varianty se mění údaje ve vygenerovaném okně. V případě dvojkliku vypíše okno všechny informace bez možnosti editace, při využití tlačítka „přidat“ se zobrazí okno bez údajů, pouze s předvyplněnou kolonkou pro ID vozidla reprezentující první volné ID v databázi pro nový záznam. V případě zvolení možnosti „spravovat“ je okno vygenerováno s daty a možností editace vyjma kolonky ID. Všechna ID v celé aplikaci jsou vždy generována automaticky dle prvního aktuálního volného místa v aplikaci a nelze je měnit.



Obrázek 15: Okno editace vozidla

Zdroj: vlastní

V případě, že chce uživatel záznam o automobilu odstranit, využije zbývající tlačítko „odeber“. V takovém případě aplikace nejdříve zkontroluje, zda auto není spojeno s některým ze záznamů v jiných tabulkách, například zda nevlastní záznam o opravě nebo nefiguruje na existujícím přepravním listě. Pokud je vše v pořádku, aplikace odebere záznam z databáze. V opačném případě je generováno chybové okno s popisem vzniklé situace.



Obrázek 16: Okno chybové zprávy

Zdroj: vlastní

Záznamy v tabulce vozidla (1) je také možno libovolně filtrovat pomocí zadávacího pole v levém horním rohu aplikace. Zde si uživatel zvolí, podle jakého kritéria chce vyhledávat a po zadání textu do pole aplikace fulltextově vyhledává patřičné záznamy v databázi.

Pod tabulkou vozidla (1) se nacházejí pomocné tabulky „STK“ (2) a „pneumatiky“ (3), které slouží k výpisu upozornění. Tabulka „STK“ (2) vypisuje vozidla, kterým bude v následujících 30 dnech končit platnost STK, a tabulka „pneumatiky“ (3) vozidla, u kterých je potřeba zajistit přezutí pneumatik, například z letních na zimní.

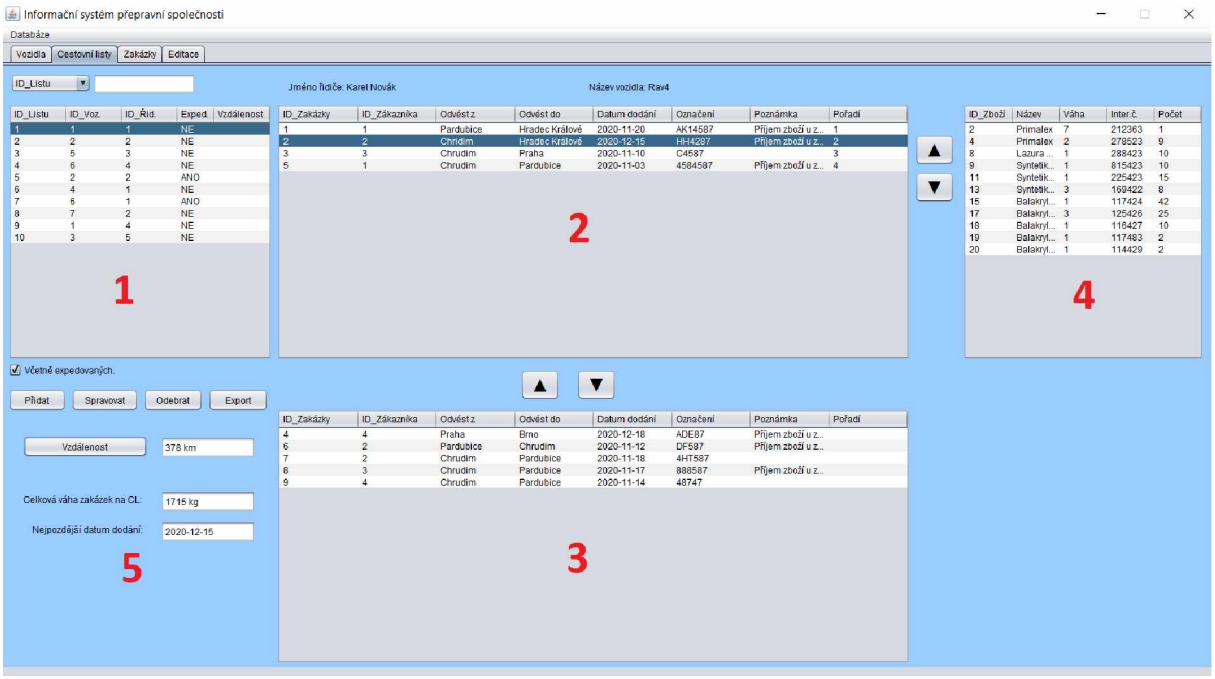
V pravé části aplikace (4) se pak nachází trojice tabulek sloužících k výpisům záznamů o tachometru daného vozidla, provedených opravách a odstávkách. Práce s tabulkami je pak stejná jako s hlavní tabulkou vozidla (1).

V celé aplikaci se pracuje s tabulkami a základními ovládacími prvky jako jsou tlačítka nebo vyhledávací pole naprosto stejně. Z toho důvodu již nebudou v dalších modulech zvlášť popisovány.

Základní funkce jsou:

- Po kliknutí na záznam tabulky dojde k výpisu do připravených polí, pokud jsou k dispozici, a naplnění ostatních tabulek s ním spojených. Taktéž je od té doby možné záznam spravovat pomocí tlačítek „přidat“, „spravovat“ a „odebrat“;
- Po dvojkliku je vygenerováno okno s podrobným výpisem bez možnosti editace;
- Po stisknutí tlačítka „přidat“ je vygenerováno okno s předem zadaným ID pro vytvoření nového záznamu;
- Po stisknutí tlačítka „spravovat“ je vygenerováno okno s podrobným výpisem s možností editace;
- Po stisknutí tlačítka „odebrat“ je provedena kontrola, zda záznam nemá vazby na jiné tabulky. Pokud ano, je generováno okno o chybě s popisem události. Pokud ne, je záznam odstraněn;
- Vyhledávání probíhá fulltextově, kdy uživatel nejdříve volí vyhledávací kritérium z rozklikávacího formuláře a následně zadává text do pole. Vyhledávání je aktivováno zadáním textu do pole.

2.2.4 Cestovní list



Obrázek 17: Modul Cestovní listy

Zdroj: vlastní

Modul Cestovní listy zajišťuje správu a kompletaci cestovních listů. V levé horní části se nachází tabulka (1) pro výpis jednotlivých listů s údaji o vozidle a řidiči k záznamu přiřazených. Dále s informací, zda již došlo k expedici a celkové vzdálenosti. Zde stojí za zmínku, že v případě editace je možno vybrat řidiče a vozidlo z vygenerovaného okna s výpisem všech záznamů a možností filtrování. Tabulku (1) je opět možno libovolně filtrovat a spravovat, jako tomu bylo v předchozím modulu. Navíc je zde umístěno tlačítko „export“ pro exportování cestovního listu do souboru.

V prostřední části modulu se nachází dvě tabulky (2,3) pro správu jednotlivých zakázek na cestovním listě. Po označení vybraného listu se v tabulce (2) přiřazených zakázek zobrazí jednotlivé záznamy. V dolní tabulce (3) jsou vypsány zatím nepřiřazené zakázky k cestovním listům. Přiřazení nebo odebrání zakázky z listu se provede označením záznamu a stisknutím směrové šipky pro přesun v prostřední části modulu.

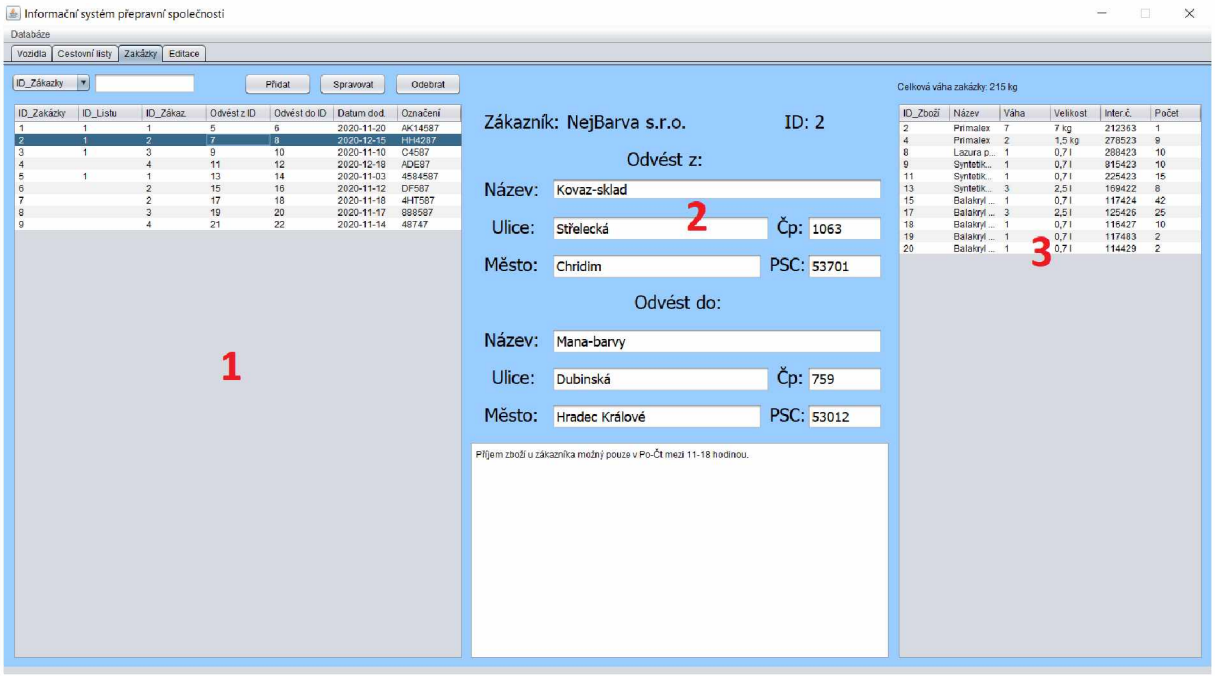
Jednotlivé zakázky na přepravním listu v tabulce (2) je možné řadit podle pořadí v jakém budou rozvezeny. K tomuto účelu slouží opět dvojice směrových šipek v pravé části vedle tabulky (2). Pořadí zakázek je důležité pro následný výpočet plánované trasy. Pořadí zásilky

v rámci cestovního listu je uchováváno v databázi a měněno dle potřeby. V případě odebrání zásilky z listu je tento údaj z databáze odstraněn.

V pravé části modulu se pak nachází tabulka (4) pro výpis zboží na zakázce.

Poslední částí modulu je pak levý dolní roh (5), kde se nachází generování vzdálenosti trasy cestovního listu, celkové váhy a nejpozdější termín dodání. Vzdálenost je generována podle pořadí zakázek na cestovním listě. K tomu je využito Google Maps API. Podrobný postup výpočtu je vysvětlen na konci uživatelské příručky. Celková váha je počítána z váhy veškerého zboží na všech zásilkách cestovního listu a slouží jako orientační údaj pro uživatele, aby nedošlo k přetížení přiděleného vozidla. Pokud jedna nebo více zakázek obsahuje datum doručení, je vybrán nejbližší termín a ten je vypsán jako nejzazší termín dodání.

2.2.5 Zakázky



Obrázek 18: Modul Zakázky

Zdroj: vlastní

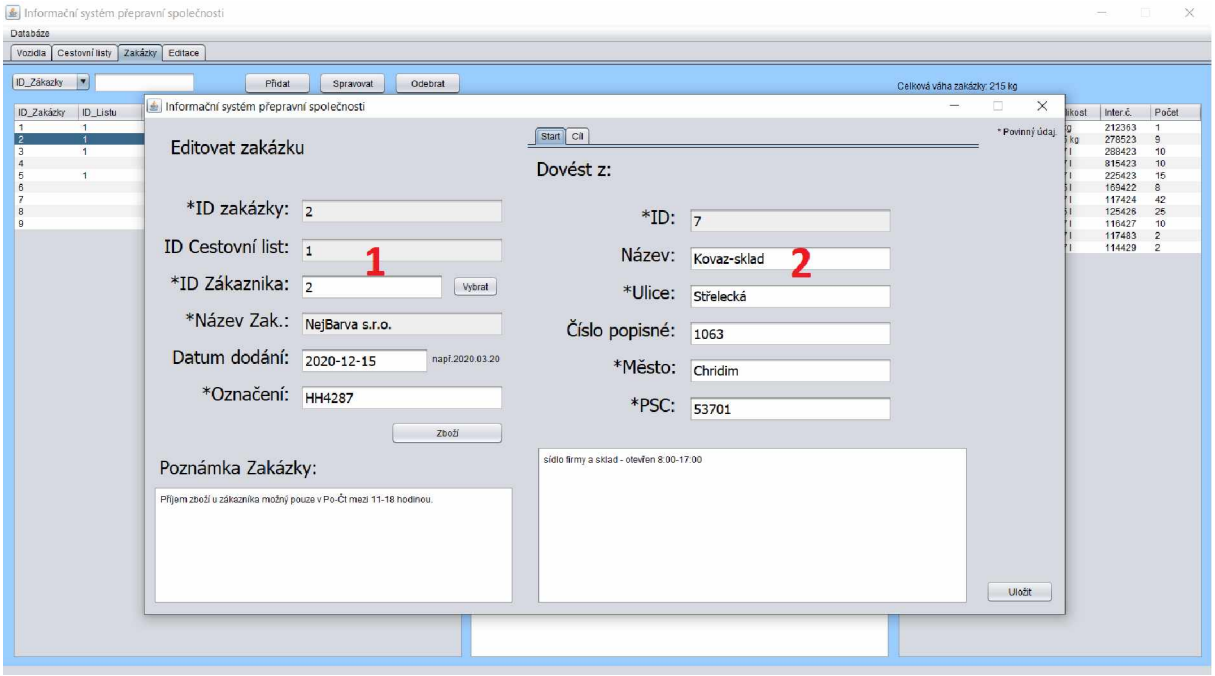
Modul Zakázky slouží ke správě jednotlivých zakázek/objednávek od zákazníků. Na levé straně je opět tabulka (1) pro výpis všech zakázek. Tabulku a záznamy v ní jde opět spravovat a filtrovat obdobným způsobem jako tabulky v předchozích modulech.

Veprostřed modulu (2) se pak nachází formulář pro výpis nejdůležitějších informací o zásilce, jako je například název zákazníka, počáteční a cílová adresa nebo celková váha zásilky.

V pravé části se pak nachází tabulka (3) pro výpis zboží na zvolené zásilce.

Zajímavostí tohoto modulu je pak vytváření a správa zásilek. Skládá se z několika generovaných oken a 2 nejdůležitější z nich budou níže popsány.

Prvním oknem je přidání nebo správa zásilky. Na rozdíl od předchozích případů se zde zobrazuje mnohem více informací získávaných z mnoha tabulek napříč tabulkami databáze.



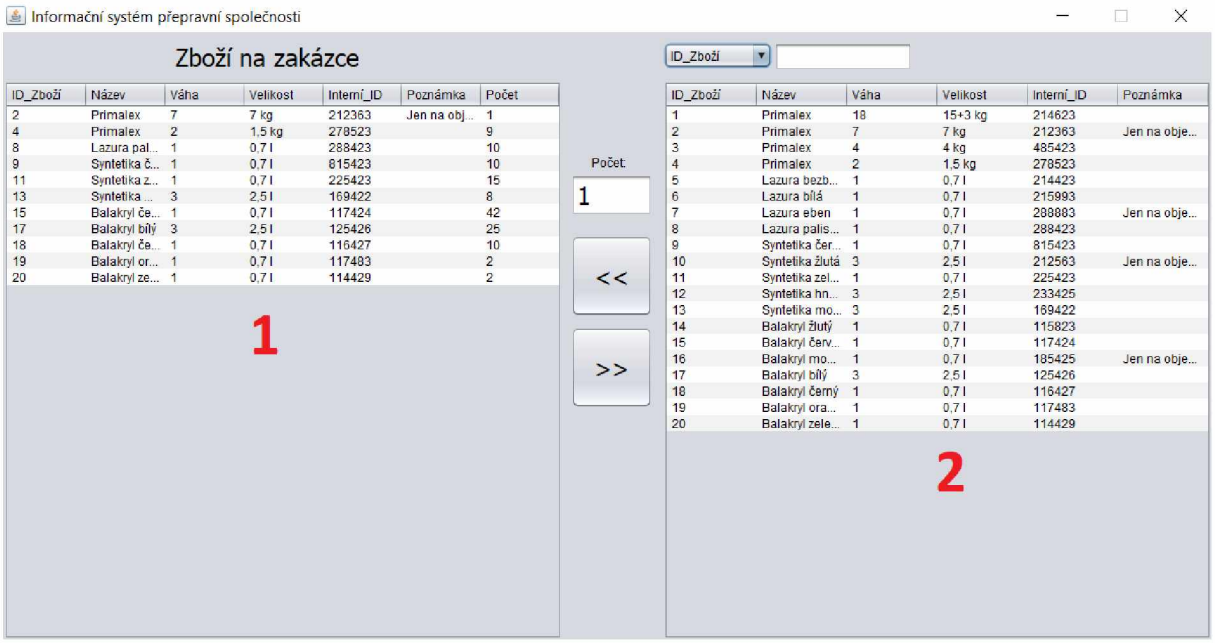
Obrázek 19: Okno editace zakázky

Zdroj: vlastní

Editační okno nám zobrazuje, tak jako v předchozích případech, základní informace o zakázce. Rozdíl je však v tom, že zatímco v levé části (1) jsou vypsány základní údaje o zakázce, v pravé části (2) má uživatel možnost přímo vytvořit nebo spravovat adresy počátečních a cílových míst. Nevybírá tedy z tabulky adres, ale vytváří rovnou nové záznamy v databázi. Logika v aplikaci je nastavena tak, že nejdříve jsou v databázi vytvořeny adresy, a až poté zakázka. Pro uživatele se to však jeví jako jeden krok po stisknutí tlačítka „uložit“ v pravém dolním rohu. V případě, že dojde k chybě a operaci „uložit“ není možné dokončit, dojde k vymazání již vytvořených záznamů z databáze. To znamená, že pokud dojde k vytvoření adres, ale nepodaří se vytvořit zásilku, jsou tyto adresy vymazány a uživateli

je generováno chybové okno s výpisem události. Těmto případům by měla zabránit kontrola vstupních dat pomocí regulárních výrazů, ovšem nikdy nemůžeme vyloučit například chybu na straně databáze.

V levé spodní části modulu (1) se nachází tlačítko „zboží“. Toto tlačítko slouží k vložení položek na zakázku. Toto tlačítko se stává aktivním až po vytvoření zakázky, jelikož není možné vložit zboží na neexistující zakázku. Po stisku tlačítka se uživateli vygeneruje okno pro přidání zboží na zakázku.

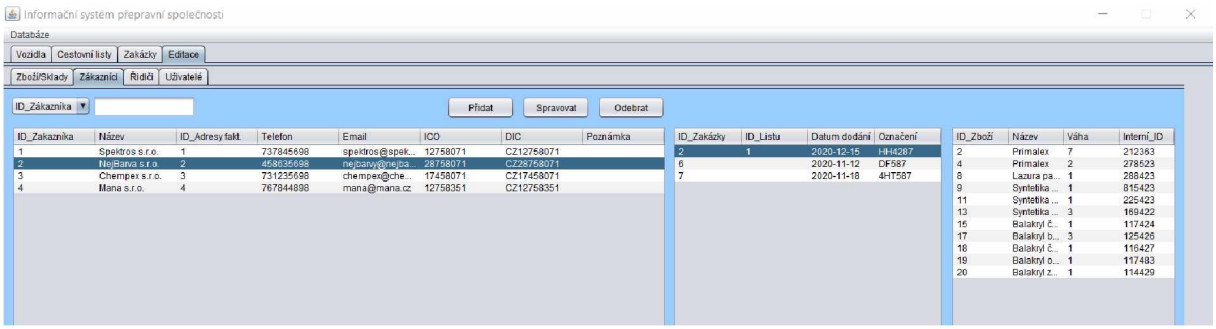


Obrázek 20: Okno přiřazení zboží

Zdroj: vlastní

V pravé části je zobrazena tabulka (1) pro výpis zboží na dané zakázce a v levé části pak tabulka (3) předuložených typů zboží. Uživatel provede vložení nebo odebrání zboží pomocí směrových šipek uprostřed se zadáním příslušného počtu kusů daného zboží. V případě opakovaného přidání stejného typu zboží není vytvořen nový záznam, ale upraven pouze počet kusů v již existujícím. To samé platí i při odebrání nižšího počtu kusů zboží, než je celkové množství na zakázce. Uživatel může také pro lepší orientaci v seznamu typů zboží libovolně fulltextově filtrovat, jako tomu je v jiných modulech. Změny při přidání nebo odebrání zboží jsou okamžitě promítnuty do databáze.

2.2.6 Editace



Obrázek 21: Modul Editace

Zdroj: vlastní

Modul Editace se skládá ze 4 podmodulů a slouží ke správě zboží, skladů, zákazníků, řidičů a uživatelů aplikace.

Všechny 4 podmoduly umožňují stejné funkcionality jako moduly předchozí a jejich ovládání je taktéž stejné. Podmodul „zákazník“ navíc umožňuje pro lepší orientaci ještě výpis jednotlivých zakázek pro vybraného zákazníka a zboží na těchto zakázkách.

Podmodul „uživatel“ umožňuje správu všech uživatelů aplikace včetně nastavení jejich přístupových práv a hesel. Data v tomto podmodulu jsou přístupná, pouze pokud má přihlášený uživatel práva pro správu uživatelů. Uživatel s příslušnými právy může kromě ostatních uživatelů manipulovat i s vlastním profilem a změnit si tak vlastní přístupová práva. Změna práv u právě přihlášeného uživatele je sice okamžitě zapsána do databáze, na aplikaci to však v daný okamžik nemá žádný vliv.

Je to z toho důvodu, aby nedošlo k nechtěnému přepsání vlastních práv. Mohla by totiž nastat situace, kdy daný uživatel je jediný, kdo může měnit práva uživatelů, a v případě, že by si toto právo odebral, nebylo by možné již upravit tato práva přímo z aplikace. Aplikace aktuální práva uživatele kontroluje a získává při připojení do databáze a následně pouze pracuje s uloženou hodnotou. Proto může uživatel svou chybu napravit, dokud nedojde k odpojení od databáze.

2.2.7 Výpočet vzdálenosti

Pro výpočet vzdálenosti v aplikaci je využito, jak už bylo zmíněno, Google Maps API. Výpočet je prováděn tak, že jsou postupně načtena všechna počáteční a cílová města zakázek, a to v pořadí, v jakém jsou seřazena na cestovním listě. Následně jsou postupným

dotazováním na Google Maps API zjištěny a sečteny jednotlivé vzdálenosti a vráceny jako celková trasa. V rámci aplikace nás zajímá pouze vzdálenost na úrovni měst. To znamená, že pokud zakázka má počáteční a cílové město stejné, nebo je stejné cílové město a počáteční město dvou zásilek po sobě jdoucích, je vypočtená vzdálenost nulová. Důvodem je, že není cílem řešit několika kilometrové přejíždění v rámci města, ale minimalizace nákladů na delších trasách například Praha–Brno.

2.3 Instalační příručka

Samotná aplikace se instalovat nijak nemusí a stačí pouze spustit příslušný jar soubor. Aplikace si po zapnutí sama načte všechny potřebné informace z konfiguračního souboru. Je však nutné vytvořit databázi a zapsat do konfiguračního souboru potřebné připojovací informace pro připojení k ní.

Níže je popsán kompletní způsob nasazení na skutečný virtuální server poskytovatele cesky-hosting.cz, na kterém běží databáze MariaDB, a to včetně zakoupení domény.

Zakoupení virtuálního serveru a domény

Prvním krokem je zakoupení virtuálního serveru na stránkách cesky-hosting.cz. Pro účel vytvářeného systému postačí nejlevnější nabízená varianta, a to Virtuální server VMS Mini. Po zakoupení virtuálního serveru je potřeba ještě dokoupit nebo zaparkovat již existující doménu, pro kterou bude v budoucnu vytvořena databáze;

Vytvoření databáze

Prostřednictvím uživatelského rozhraní na stránkách cesky-hosting.cz je nutné vytvořit pro příslušnou doménu novou MariaDB databázi. Pro následnou správu databáze pak provozovatel nabízí bezplatný nástroj phpMyAdmin;

Zavedení skriptu Adminer

Pro lepší správu databáze je výhodné, aby uživatel zavedl ještě open source skript Adminer. Zavedení je jednoduché, stačí si ze stránek tvůrců adminer.org stáhnout tento nástroj v podobě jediného skriptu. Následně ho nakopírovat na webserver do prostoru příslušné domény. Skript poté jednoduše spustíme zadáním příslušné URL. Například v tomto případě lubosvojacek.eu/adminer-4.7.7-cs.php.

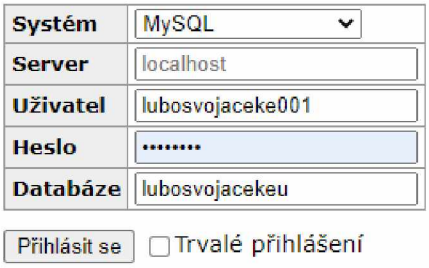
Do webserveru pro nakopírování skriptu se uživatel dostane prostřednictvím odkazu webftp.cesky-hosting.cz;

Obrázek 22: Přihlášení na FTP

Zdroj: vlastní, snímek webového rozhraní [9]

Vytvoření tabulek v databázi

Pro vytvoření tabulek je nejdříve nutné se přihlásit do databáze prostřednictvím Adminer skriptu. Uživatel tedy zadá do prohlížeče adresu lubosvojacek.eu/adminer-4.7.7-cs.php a vyplní přihlašovací údaje do databáze získané při vytvoření databáze.



Obrázek 23: Přihlášení do databáze

Zdroj: vlastní, snímek webového rozhraní Adminer

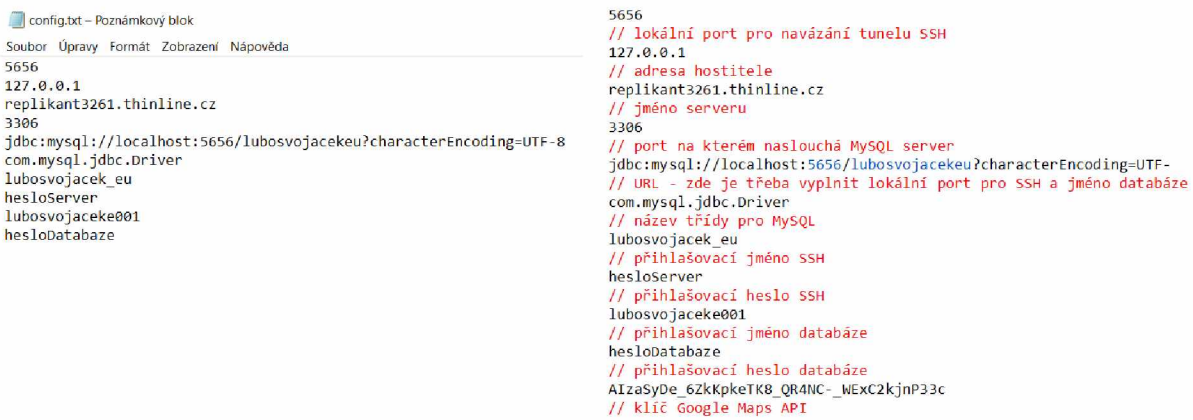
Po připojení se hned zobrazí zadávací pole pro SQL příkazy. Zde buď může uživatel ručně nakopírovat schéma z přiloženého souboru schemaDB.sql, nebo využít variantu importu v levé části stránky. Po provedení importu Adminer automaticky vytvoří všechny tabulky a vazby. V případě potřeby může uživatel tuto operaci opakovat pro vložení vzorových dat ze souboru vzordataDB.txt;

Nastavení konfiguračního souboru a generování hesla

Posledním krokem je zapsání potřebných údajů do konfiguračního souboru včetně vygenerování hesla pro SSH a databázi.

K vyvíjené aplikaci je vytvořen i jednoduchý program pro generování hesel. Tento program obsahuje stejný klíč jako diplomovou prací popisovaná aplikace. Do tohoto programu zadá uživatel jím zvolené heslo databáze (dále jen ADB) a SSH (dále jen ASSH), které později zapíše do konfiguračního souboru. Program mu následně vygeneruje skutečná hesla pro databázi (dále jen BDB) a SSH (dále jen BSSH). Uživatel pak v uživatelském rozhraní stránek cesky-hosting.cz nastaví heslo pro databázi na BDB a pro SSH na BSSH.

Po získání a nastavení hesel otevře uživatel konfigurační soubor programu a vyplní ho dle přiloženého vzoru;



Obrázek 24: Ukázka (zleva) konfiguračního a vzorového souboru

Zdroj: vlastní

Spuštění aplikace

V tuto chvíli již zbývá pouze spustit samotnou aplikaci a poprvé se připojit. Pro umožnění prvního přihlášení je v tabulce uživatelů vytvořen uživatel LubošV s heslem Heslo1 se všemi dostupnými právy.

2.4 Externí knihovny

- AbsoluteLayout.jar;
- dom4j-2.1.1.jar;
- jsch-0.1.55.jar;

- rs2xml.jar;
- mysql-connector.jar;
- okhttp-3.5.0.jar;
- okio-1.11.0.jar;
- google-maps-services-0.1.5.jar;
- gson-2.3.1.jar;
- joda-time-2.4.jar;
- okhttp-2.0.0.jar;

ZÁVĚR

Diplomová práce splňuje v plné míře požadavky, které jsou na ní kladeny, a to jak samotným zadáním, tak stanovenými cíli v průběhu analýzy požadavků zadavatele a rešerší existujících řešení.

Vyvíjený systém skládající se z aplikace v jazyce Java a databáze MariaDB je plně funkční a v současné době reálně nasazený. Celý systém byl dlouhodobě testován v reálných podmínkách firmy zadavatele. Práce je navržena tak, aby jí bylo možné v budoucnu rozšířit, s čímž je i počítáno. Kód databáze je kompatibilní jak s databází MySQL, tak MariaDB. Díky tomu je možné nasazení na většinu virtuální serverů, které jsou v současné době nabízeny.

V případě, že se uživatel rozhodne nasadit databázi na jiný virtuální server, který má stejný princip vzdáleného přístupu, stačí pouze změnit údaje v konfiguračním souboru a není tedy nutno vstupovat do kódu aplikace. Pokud se způsob liší, stačí pouze upravit třídu pro připojení bez nutnosti upravovat celou aplikaci. Díky tomu bude do budoucna možné rozšířit aplikaci o různé varianty připojení.

Pro účely testování je v současné době databáze nasazena na virtuálním hostingu včetně testovacích dat. Zde je možné si celé řešení vyzkoušet bez nutnosti vytváření vlastní databáze. Stačí pouze spustit aplikaci uloženou v příloze v elektronické podobě. Aplikace si sama načte potřebné informace z konfiguračního souboru, který byl za tímto účelem předvyplněn. Po připojení aplikace k databázi se uživatel přihlásí pomocí testovacího profilu se jménem „LubošV“ a heslem „Heslo1“. Testovací profil má nastavena nejvyšší možná práva, a tedy umožňuje uživateli využívat veškeré funkcionality aplikace. Během testování aplikace je uživateli důrazně doporučeno neměnit práva u testovacího profilu. V případě, že tato práva sníží a odhlásí se, aniž by nastavil práva zpět nebo vytvořil jiného uživatele s nejvyššími právy, nebude již možné práva z prostředí aplikace upravovat a mohou mu být omezeny i některé další funkcionality aplikace.

POUŽITÁ LITERATURA

[1] *TRIFID software Vysoké Mýto* [online]. Vysoké *Mýto: TRIFID software*, [2010] [cit. 2020-07-12]. Dostupné z: https://www.trifid-sw.cz/

[2] *Toptrans* [online]. Praha: TOPTRANS EU, c2003-2019 [cit. 2020-07-12]. Dostupné z: https://www.toptrans.cz/

[3] *Zdravím! Jsem TRAXEE* [online]. Brusel: WEBCO, 2020 [cit. 2020-07-20]. Dostupné z: https://wabco-traxee.com/cs/

[4] *Kompletní řešení správy vozového parku | Mapon* [online]. Riga: Mapon, 2020 [cit. 2020-07-20]. Dostupné z: https://www.mapon.com/cz

[5] *MySQL* [online]. Redwood Shores: Oracle Corporation, 2020 [cit. 2020-07-09]. Dostupné z: https://www.mysql.com/

[6] STEPHENS, Ryan K., Ronald R. PLEW a Arie JONES. *Naučte se SQL za 28 dní: [stačí hodina denně]*. Dotisk 1.vydání. Brno: Computer Press, 2010. ISBN 978-80-251-2700-1.

[7] *Welcome to NetBeans* [online]. USA: Apache Software Foundation, © 2017-2019 [cit. 2020-07-09]. Dostupné z: https://netbeans.org/

[8] SCHILDT, Herbert. *Mistrovství – Java*. Brno: Computer Press, 2014. Mistrovství. ISBN 9788025141458.

[9] *Český hosting: webhosting a registrace domény* [online]. Praha: THINline, 2020 [cit. 2020-07-12]. Dostupné z: https://www.cesky-hosting.cz/

[10] *MariaDB Foundation - MariaDB.org* [online]. Delaware: MariaDB Foundation, c2009 - 2020 [cit. 2020-07-12]. Dostupné z: view-source:https://mariadb.org/

[11] *UML modeling tools for Business, Software, Systems and Architecture* [online]. Creswick: Sparx Systems, c2000-2020 [cit. 2020-07-21]. Dostupné z: https://sparxsystems.com/

[12] *MySQL: MySQL Workbench* [online]. Redwood Shores: Oracle Corporation, c2020 [cit. 2020-07-21]. Dostupné z: https://www.mysql.com/products/workbench/

[13] *ARIS Community Recent Comments* [online]. Darmstadt: Software, c2009 - 2020 [cit. 2020-07-21]. Dostupné z: https://www.ariscommunity.com/

[14] KANISOVÁ, Hana a Miroslav MÜLLER. *UML srozumitelně*. 2., aktualiz. vyd. Brno: Computer Press, 2006. ISBN 80-251-1083-4.

[15] ARLOW, Jim a Ila NEUSTADT. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. 2., aktualiz. a dopl. vyd. Brno: Computer Press, 2007. ISBN 978-802-5115-039.

[16] *BPMN Specification - Business Process Model and Notation* [online]. Needham: Object Management Group, c1997 - 2020 [cit. 2020-07-21]. Dostupné z: http://www.bpmn.org/

[17] *BPMN Specification - Business Process Model and Notation* [online]. Needham: Object Management Group, c1997 - 2020 [cit. 2020-07-21]. Dostupné z: http://www.bpmn.org/

[18] ŘEPA, Václav. *Analýza a návrh informačních systémů*. Praha: Ekopress, 1999. ISBN 80-861-1913-0.

[19] GROFF, James R. a Paul N. WEINBERG. *SQL: kompletní průvodce*. Brno: CP Books, 2005. Programování (CP Books). ISBN 80-251-0369-2.

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

PŘÍLOHY

Příloha I: VOJACEKL_INFORMACNISYSTEM_MM_2020.ZIP

Elektronická příloha ve formátu zip obsahuje vyvíjenou aplikace pro Informační systém přepravní společnosti, aplikace pro generování hesla a testování, textovou část diplomové práce, vytvořené diagramy a další podpůrné soubory. Příloha je rozdělena dle následující struktury.

- Praktická část
  - Aplikace
    - Generator_hesel.zip
    - Hlavni_aplikace.zip
    - Testovaci_aplikace.zip
  - SQL_Scripty
    - schemaDB
    - schemaDB+vzordataDB.txt
    - vzordataDB.txt
- Teoretická část
  - Diagramy
    - EER_Diagram.mwb
    - ISPS.EAP
    - Vojáček_BPMN.adf
    - Vojáček_BPMN.pdf
    - Vojáček_Diagram_případů_užití.pdf
    - Vojáček_Diagram_případů_užití.png
    - Vojáček_ER_diagram.pdf
    - Vojáček_Funkční_požadavky.pdf
    - Vojáček_komunikační_diagram.pdf
    - Vojáček_komunikační_diagram.png
    - Vojáček_Nefunkční_požadavky.pdf
    - Vojáček_sekvenční_diagram.pdf
    - Vojáček_sekvenční_diagram.png

- VojacekL_InformacniSystem_MM_2020.pdf