UNIVERZITA PARDUBICE
FAKULTA EKONOMICKO-SPRÁVNÍ

Tvorba aplikace pro logistickou firmu

Miroslav Pásler

Bakalářská práce

2011

Univerzita Pardubice
Fakulta ekonomicko-správní
Akademický rok: **2010/2011**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Miroslav PÁSLER**
Osobní číslo:	**E08261**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Informatika ve veřejné správě**
Název tématu:	**Tvorba aplikace pro logistickou firmu**
Zadávající katedra:	**Ústav systémového inženýrství a informatiky**

Zásady pro vypracování:

1) Zkoumání současného stavu, potřeb a požadavků dispečera ve vybrané logistické firmě
2) Možnosti zpracování a evidence objednávek přeprav pomocí IS, návrh řešení
3) Implementace a zhodnocení softwarové aplikace pro efektivnější zpracování této agendy

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

[1] KOSEK, Jiří. PHP : Tvorba interaktivních internetových aplikací. [s.l.], Grada Publishing, 1999. 492 s. ISBN 80-7169-373-1.

[2] Rozlívka Transport s.r.o. [online]. 2010-06-22 [cit. 2010-06-22]. Dostupný z www <http://www.jr-cz.eu>

[3] Spediční databanka RaalTrans. [online]. 2010-06-22 [cit. 2010-06-22]. Dostupný z www <http://www.raal.cz/cs>

[4] KOMÁRKOVÁ, Jitka. Úvod do informačních systémů. Pardubice : Univerzita Pardubice, 2006. 85 s. ISBN 80-7194-870-5.

Vedoucí bakalářské práce: **Ing. Milan Tomeš**
Ústav systémového inženýrství a informatiky

Datum zadání bakalářské práce: **4. října 2010**

Termín odevzdání bakalářské práce: **6. května 2011**

L.S.

doc. Ing. Renáta Myšková, Ph.D.
děkanka

doc. Ing. Jiří Křupka, Ph.
vedoucí ústavu

V Pardubicích dne 4. října 2010

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

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 24. dubna 2011

Miroslav Pásler

Poděkování

Velmi rád bych poděkoval Ing. Milanu Tomešovi za čas, který mi při tvorbě práce věnoval a za připomínky a rady, kterými přispěl k vypracování této bakalářské práce.

Dále bych rád poděkoval pracovníkům firmy ROZLÍVKA TRANSPORT s.r.o. za jejich spolupráci při získávání informací a podkladů k této práci.

Anotace

Práce se zabývá návrhem a následnou implementací informačního systému pro konkrétní firmu podnikající v oboru logistiky. Systém má být navržen tak, aby usnadnil práci zaměstnancům firmy a podpořil základní práci managementu firmy.

Klíčová slova

aplikace, informační systém, programování, Java, PHP MyAdmin, databáze, logistika, životní cyklus, SQL

Title

Creating of aplication for logistic company

Annotation

The work deals with the design and subsequent implementation of an information system for a particular company engaged in the field of logistics. The system should be designed to facilitate the work of employees of the company and support the essential work of management.

Keywords

information system, programming, Java, PHP MyAdmin, database, logistics, life cycle, SQL

Obsah

Seznam obrázků

Seznam tabulek

Seznam příloh

Úvod

V dnešní době, kdy informatika a informační a komunikační systémy prostupují veškerou činností člověka a pro většinu konkurenceschopných firem jsou nepostradatelným nástrojem, stále existují firmy, které ICT využívají nedostatečně nebo dokonce vůbec. S rozvojem internetu vzrostly i požadavky na alternativní způsoby prezentace firem při zachování konkurenceschopnosti na daném trhu. Neexistuje dnes prakticky firma, která by se nějakým způsobem neprezentovala na internetu. Vyhledatelnost a viditelnost v prostředí internetu se pro některé firmy stává existenční nutností. Stejně tak i využívání speciálních prostředků z oblasti informačních technologií, různého programového vybavení a komunikačních technologií se postupem času stává nezbytnou součástí kvalitního chodu firem na všech úrovních velikosti i specializace. S ohledem na to, využití prostředků ICT může být rozhodujícím faktorem pro dynamiku práce firmy, usnadnění rozhodovacích procesů a její vnější prezentaci ve výsledku vedoucí k dlouhodobě udržitelné konkurenceschopnosti firmy.

Úkolem této práce je poskytnout podklad pro řešení problému nedostatečného využití možností použití ICT v konkrétní firmě zabývající se spedicí a logistikou, navrhnout počítačovou aplikaci jako součást IS, která by usnadnila spedičním pracovníkům firmy sepisování objednávek a evidenci a management firmy podpořila v jeho rozhodovacích procesech poskytnutím automatických součtů některých finančních ukazatelů. Na základě návrhu pak zvolit vhodný implementační nástroj a tuto aplikaci vytvořit.

Návrh informačního systému je složitý teoretický proces vyžadující jistou míru specializace, proto je této problematice věnována první část této práce. Na základě těchto teoretických faktů je poté uskutečněna samotná realizace návrhu včetně stanovení požadavků a systémového vymezení problému. Dále jsou v práci popsány významné problémy související s implementací na základě zvoleného nástroje.

1. Teorie vývoje informačního systému

V této kapitole bude obecně pojednáno o vývoji IS, tedy o aspektech, postupech a dalších sounáležitostech souvisejících s tvorbou IS. Budou zde vysvětleny vybrané pojmy týkající se teorie informačních a komunikačních systémů.

1.1. Systém

Dříve než budou blíže vysvětleny vybrané pojmy z teorie informačních systémů, je třeba obecně definovat pojem systém. Pro pojem systém existuje velké množství definic. Pro účely této práce se jeví jako vhodná tato definice: „*Systémem rozumíme obecně soubor prvků, mezi nimiž existují vzájemné vztahy a jako celek má určité vztahy ke svému okolí.*“ [1] Systém odděluje od jeho okolí **hranice systému**. Pomocí hranice systému je systém jasně prostorově vymezen.

1.2. Informační systém

Pojem informační systém můžeme chápat jako konkretizaci pojmu systém, jež byl definován. Pro pojem informační systém zná vymezení i naše legislativa.

„*Informačním systémem se rozumí funkční celek zabezpečující cílevědomé a systematické shromažďování, zpracovávání, uchovávání a zpřístupňování informací. Každý informační systém zahrnuje informační základnu, technické a programové prostředky, technologie a procedury a pracovníky.*“

[2]

„*Pro účely tohoto zákona se rozumí informačním systémem funkční celek nebo jeho část zabezpečující cílevědomou a systematickou informační činnost. Každý informační systém zahrnuje data, která jsou uspořádána tak, aby bylo možné jejich zpracování a zpřístupnění, a dále nástroje umožňující výkon informačních činností.*“ [3]

Informačním systémem můžeme rozumět funkční propojení lidí, dat, procesů (softwaru) a technologií (hardwaru) navzájem spolupracujících tak, aby usnadňovali a podporovali každodenní operace a podporovali řešení problémů a rozhodovací procesy. [4]

1.3. Životní cyklus vývoje IS

Jedná se o posloupnost po sobě následujících fází - etap. Životní cyklus vývoje IS jednoznačně definuje všechny jednotlivé etapy vývoje od plánování až po ukončení provozu IS.

„*Informační systém je produkt jako kterýkoliv jiný, má proto také svůj životní cyklus podchycující celý jeho „život“ od počátku až do konce.*“ [4]

Existuje mnoho různých způsobů vymezení fází životního cyklu IS. Stejně tak existuje několik modelů životního cyklu IS, které popisují jednotlivé etapy a jejich náplň.

Nejstručněji lze jednotlivé fáze definovat následujícím výčtem:

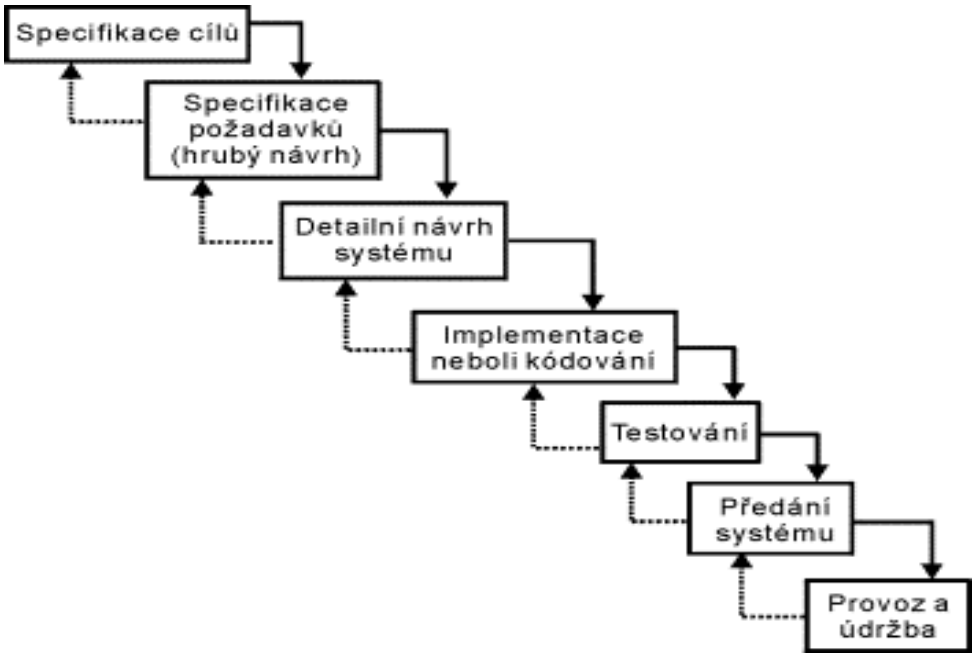
- plánování
- analýza a design
- zavádění do provozu (implementace)
- provoz a údržba

Vymezení ŽC IS v předchozích čtyřech bodech nám nenabízí veškeré etapy, které lze do vývoje IS zařadit. Jako samostatnou fázi lze jistě vymezit fázi vyřazení IS z provozu. [4]

Životní cyklus vývoje IS lze znázornit pomocí několika modelů, každý s těchto modelů přistupuje k popisu etap, k jejich náplni a především k jejich návaznosti odlišným způsobem. Mezi nejběžnější modely ŽC IS patří **vodopádový model** (Obrázek 1), V-model, spirálový model, objektově-orientovaný model a další. V rámci rozsahu této práce se jeví jako nejvhodnější uvést a podrobněji rozebrat vodopádový model, který je také někdy nazýván jako tradiční nebo kaskádový model.

„*Základní charakteristikou modelu vodopád je, že při návrhu IS se provádí postupně jednotlivé etapy životního cyklu, které na sebe navazují a vzájemně se neprotínají. Etapy se provádí podle přesného plánu realizace a zpětně se k nim nevrací, dokončená etapa je vstupem etapy následující.*“ [5]

Tento model je typický tím, že se testuje správnost jednotlivých fází a v případě zjištění chyb, se lze vrátit do fáze předchozí a chyby napravit. Tento systém zajišťuje vyšší kvalitu a menší pravděpodobnost přehlédnutí důležitých záležitostí. Na druhou stranu při použití tohoto modelu je konečný výsledek znám až po poslední fázi.



Obrázek 1 - Vodopádový model ŽC IS. Zdroj [5]

1.3.1. Specifikace cílů a specifikace požadavků

„*Základem celkového návrhu, vývoje i jakékoli úpravy stávajícího systému jsou požadavky uživatelů a cíle organizace. V této části se musí dané požadavky shromáždit, v hrubých rysech rozebrat a odhadnout dobu realizace a náklady.*“ [5]

Cílem této fáze je tedy sestavit základní rámec požadavků a vytyčit základní cíle. Specifikace cílů a požadavků na IS se dá souhrnně nazvat jako **analýza** systému. Rozdílem mezi specifikací cílů a specifikací požadavků je v tom, že specifikace cílů předchází v rámci vodopádového modelu specifikaci požadavků. Specifikace cílů je tedy jakási předběžná analýza systému. Samotná fáze analýzy je pro vývoj IS v rámci vodopádového modelu ŽC jedna z klíčových, protože chyby, kterých se dopustíme v této fázi, a neodhalíme je, se později velice těžko napravují a jejich odstranění bývá velice časově náročné a v neposlední řadě i velice nákladné. [5]

Součástí projektu specifikace cílů tedy dle [5] jsou:

- Časový plán projektu.
- Zdroje nutné k řešení (HW, SW, lidé, finanční prostředky).
- Odhad funkčnosti, rozsahu systému, ekonomické efektivnosti a návratnosti investice.

Pro specifikování daných cílů se použijí nástroje:

- Analýza současného stavu
- Získání požadavků uživatelů a zjištění požadovaných vstupních a výstupních informací.
- Seznam problémů, které jsou známy.

Specifikace požadavků:

- Formální specifikace
- Neformální specifikace

1.3.2. Návrh systému

Návrh systému představuje popis způsobu realizace systému. Podkladem pro návrh IS jsou výsledky z předcházející fáze ŽC tedy z analýzy. S návrhem systému velice úzce souvisí pojem **procesní modelování**. Modely vzniklé při procesním modelování můžeme rozdělit na **formální** (přirozený jazyk) a **neformální** (pomocí dané syntaxe). Pro účely procesního modelování nám slouží nástroje jako je UML diagram, Use-case diagram, vývojový diagram, EPC diagram a další.

1.3.3. Implementace systému

Implementace představuje převedení návrhu IS do podoby reálného výrobku pomocí nějakého programovacího jazyka nebo nástroje. Někdy bývá jako implementace označována ta etapa ŽC, kdy je již hotový IS zaváděn do provozu.

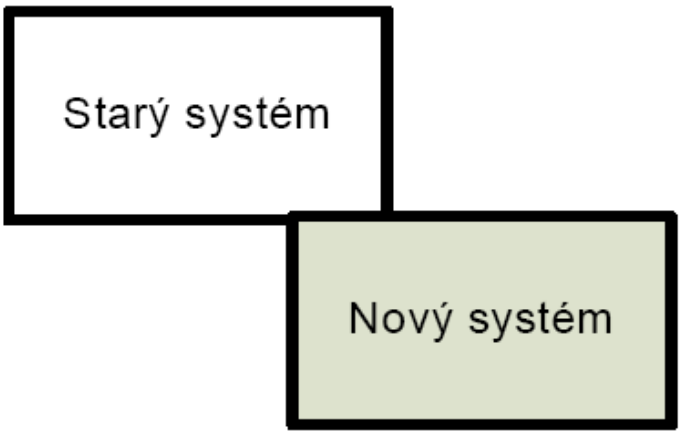
1.3.4. Testování

V této etapě se provádí připravené testy na hotovém IS. Je nutné vyzkoušet veškeré možné reakce systému na zadávaná data a zjištěné nedostatky opravit. Testování se často provádí na systému, který ještě není v reálném prostředí, neboť případné selhání by mohlo mít rozsáhlé následky. [5]

1.3.5. Zavádění do provozu

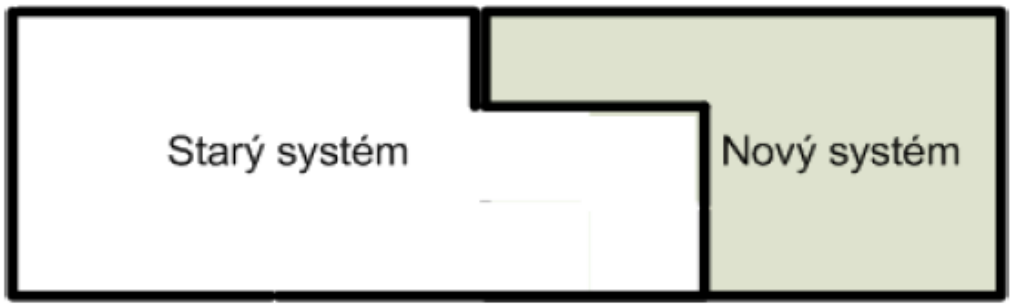
Tato etapa představuje uvedení nového IS do provozu. Při zavádění nového systému do provozu lze dle [4] a dle [5] použít jednu z uvedených strategií:

- **Souběžné zavádění** – po jistou dobu pokračuje provoz starého IS za současného provozu systému nového. Tato metoda je bezpečná a spolehlivá, ale náročná pro zaměstnance (Obrázek 2).



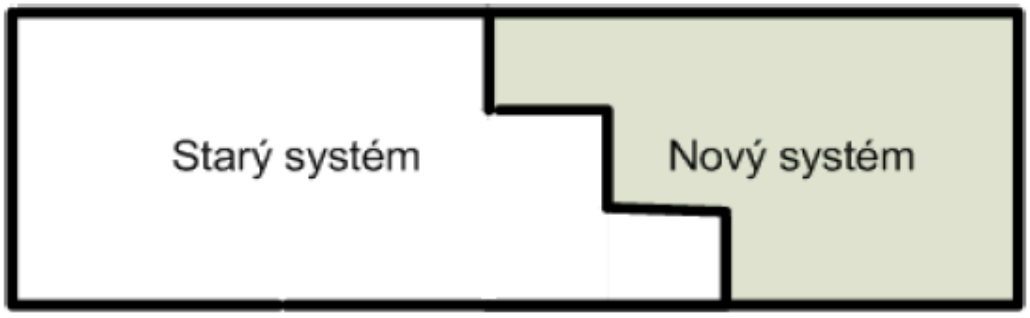
Obrázek 2 - Souběžný systém zavádění IS. Zdroj [4]

- **Pilotní zavádění** – nový systém je zaveden jen na některých pracovištích podniku a teprve po jeho odzkoušení je zaveden i ve zbytku podniku. U této strategie mohou nastat problémy v souvislosti s komunikací mezi pracovišti, kdy jedna část podniku pracuje již s novým IS a zbylá se starým. Jinak je tato forma zavádění bezpečná (Obrázek 3).



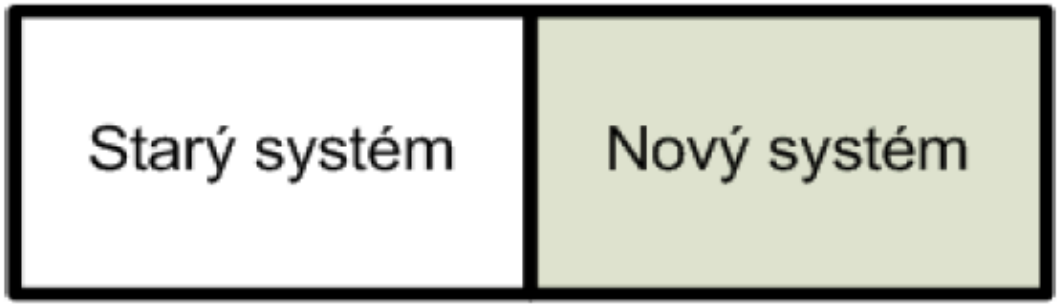
Obrázek 3 – Pilotní systém zavádění IS. Zdroj [4]

- **Postupné zavádění –** používá se u složitých systémů. Postup zavádění je od částí systému, na kterých zbylé části závisí na klíčová pracoviště k systémům závislým na sekundární pracoviště (Obrázek 4).



Obrázek 4 – Postupný systém zavádění IS. Zdroj [4]

- **Nárazové zavádění** – jedná se o úplné nahrazení starého systému systémem novým v jednom okamžiku. Tento postup je velice riskantní, ale přináší významnou úsporu času (Obrázek 5).



Obrázek 5 – Nárazový systém zavádění IS. Zdroj [4]

1.3.6. Provoz a údržba systému

Tato etapa představuje zajištění správného provozu systému. Spadá sem zabezpečení systému a ochrana dat před neoprávněným přístupem, chránění dat před ztrátou pomocí archivace apod. Chyby, které nastaly v předešlých fázích a jsou odhaleny až ve fázi provozu, se jen velmi těžko napravují a tato náprava bývá velice nákladná.

1.4. Požadavky na IS

Na informační systémy jsou dle [4] kladeny různé požadavky. IS by měl být:

- **otevřený** - v závislosti na vnějším prostředí. Pokud se systém nazývá otevřeným, musí existovat možnost doplňování všech komponent systému od různých dodavatelů, kteří potom mají možnost systém upravovat (i programově), v reakci na příslušné změny v místě.

- **dynamický** – systém tzv. „půjde s dobou“. Obvykle se problém řeší formou garance vývoje na několik let.
- **podporovaný** - podpora českého prostředí (komunikuje s uživatelem v češtině).
- **komplexní** - tj. systémy, které systematicky zabezpečují informacemi veškeré složky řízení a organizace.
- **kompaktní** - neboli vnitřně propojené. Takovýto systém má všechny požadované (odůvodněné) vnitřní vazby mezi jednotlivými subsystémy i jednotlivými daty. Má vytvořené jak vazby horizontální (na stejné rozlišovací úrovni), tak vazby vertikální (na hierarchicky odlišných rozlišovacích úrovních).
- **standardizovaný** - respektující všeobecně platné technické i datové předpisy, české obzvláště. Tato vlastnost umožňuje realizovat vazby na vnější okolí, zajišťuje, aby byl systém kompatibilní s dalšími systémy.
- **stavebnicový** - kdy jednotlivé softwarové komponenty lze vyměňovat po blocích.
- **chráněný** - jak před zneužitím tak před úmyslným i neúmyslným poškozením techniky, dat, a softwarové části. Zajištění bezpečnosti dnes patří mezi významné požadavky na IS. Jde přitom o komplexní a nikdy nekončící proces, který vyžaduje systematický přístup začínající definováním strategie informační bezpečnosti a analýzou rizik, pokračující definováním bezpečnostní politiky organizace a bezpečnostními směrnicemi a standardy, po nichž následuje jejich zavedení do praxe včetně školení pracovníků a následuje neustálá kontrola a monitorování, která zajišťuje zpětnou vazbu.
- **kompatibilní** - neboli slučitelný. Jde o to, aby jednotlivé systémy bylo možno vzájemně propojovat – interoperabilita systémů (schopnost spolupráce bez ohledu na platformu jednotlivých systémů) patří dnes ve veřejné správě (a nejen v ní) k významným požadavkům při tvorbě nových systémů
- **minimalizovat datové redundance** – data, která se vyskytují na jednom místě, by se neměla vyskytovat nezávisle i na jiných místech, ale pouze ve formě propojení.

2. Vybrané pojmy z logistiky

Informační systém vytvořený v rámci praktické části této práce je zaměřený pro potřeby logistické firmy. Z tohoto důvodu v této kapitole budou popsány vybrané pojmy z logistiky a oborů s ní úzce souvisejících. Protože daná logistická firma se zaměřuje především na automobilovou dopravu a spedici, budou v této kapitole popsány především pojmy související s touto problematikou.

2.1. Logistika

Pro pojem logistika existuje několik možných definic. Jedna z nich zní:

„*Logistika je disciplína, která se zabývá pohybem zboží a materiálu z místa vzniku do místa spotřeby*.“ [6]

Pojem logistika dále souvisí s mnoha dalšími pojmy, pro účely této práce byly k podrobnějšímu popisu vybrány následující a popsány dle [7] a [8]:

- **Spedice** – dle [7] je spedice zajištění přepravy. V praxi tento pojem znamená poskytnutí služby distribuce nebo zajištění přepravy zboží z jednoho místa na místo jiné a velice úzce souvisí s pojmem logistika, protože na ni navazuje i vykládka, nakládka a skladování zboží, které už do samotné spedice nepatří, ale firmy, které se spedicí zabývají, často poskytují i tyto služby a hranice mezi spedicí a logistikou (jako širší nabídkou služeb než je pouze spedice) se smazává.
- **Dopravce –** poskytovatel přepravních služeb.
- **Vozidlový park** – soubor dopravních prostředků trvale patřící do systému silniční nebo železniční dopravy.
- **Skladování** – je část logistického systému, která zabezpečuje uskladnění produktů a informace o stavu, podmínkách a rozmístění skladovaných produktů.
- **Zásobování –** jedná se o užší pojem nežli logistika. Jedná se o zajišťování hmotných statků a služeb.

2.2. Informační systémy a informační technologie v logistice

2.2.1. Spediční databanka RaalTrans

Databanka RAALTRANS slouží pro zadávání, vyhledávání a třídění nabídek přeprav a volných vozů. Jedná se o centrální databázi vytížení vozidel. V praxi to znamená, že uživatelé této databanky skrze programové rozhraní RAALTRANS Editor zveřejňují, kde se nacházejí jejich volné vozy popřípadě zboží, které chtějí přepravovat a naopak v databázi hledají volné vozy, jež by mohly uskutečnit požadovanou přepravu. Databanka RAALTRANS je tedy v podstatě virtuální trh, kde se setkává nabídka a poptávka po volných vozech a zboží, jež je třeba přepravit za účelem maximálního vytížení vozů a maximálního urychlení a usnadnění přepravy. [9]

V rámci databanky RAALTRANS se jednotliví dopravci rozlišují pomocí kódu RAAL.

3. Vymezení problému a stanovení požadavků na IS

V této kapitole je vymezen problém řešený v této práci. Dále popsány vybrané informace o firmě ROZLÍVKA TRANSPORT s.r.o., průzkum tamního prostředí a na jejím základě stanoveny požadavky na IS a aplikaci.

3.1. Vymezení problému

Byl stanoven požadavek, na vytvoření počítačové aplikace jako součásti IS pro potřeby konkrétní logistické firmy ROZLÍVKA TRANSPORT s.r.o., konkrétně pro usnadnění práce spedičního oddělení. Tato firma se zabývá automobilovou nákladní dopravou vnitrostátní i mezinárodní a dalšími logistickými službami.

3.2. Informace o firmě

Informace o firmě ROZLÍVKA TRANSPORT s.r.o. byly čerpány jednak přímo na místě od pracovníků firmy a jednak z webových stránek firmy [10].

Firma ROZLÍVKA TRANSPORT s.r.o. vznikla v roce 2004 převedením z firmy AUTODOPRAVA Jiří Rozlívka, která byla založena v roce 1991. Stěžejní činností firmy je vnitrostátní a mezinárodní nákladní doprava. Firma nabízí spektrum logistických služeb, mezi které patří zajištění přepravy nákladů v rozmezí 1kg až 24t formou expresních zásilek, sběrných služeb či celovozovou dopravu, skladování a vykládka a nakládka. Zásilky firma dopravuje po celém světě zejména pak po tuzemsku, do Německa, Polska, Maďarska a na Slovensko.

Firma sídlí na adrese Semanínská 2094 Česká Třebová. Areál firmy se nachází v bezprostřední blízkosti hlavního vlakového nádraží města Česká Třebová a rozkládá se na rozloze 11600 m2, na tomto prostoru se nachází 1500m2 skladových prostor a zařízení uzpůsobená k nakládkám a vykládkám nákladních aut i železničních vagonů.

Firma zaměstnává celkem 12 zaměstnanců, které pod sebou zaměstnávají dva ředitelé, tedy jednatelé firmy p. Rozlívka a jeho otec. Mezi tyto zaměstnance patří dva spediční pracovníci, jedna účetní a devět řidičů. Firma má základní kapitál 200 000 Kč a přibližný roční obrat 20 – 25 mil. Kč.

Spediční pracovníci firmy zajišťují zprostředkování přepravy zásilek vnitrostátních i mezinárodních především automobilovou dopravou, ale i železniční námořní a leteckou. K těmto účelům firma využívá buď vlastních vozidel, nebo pronájem vozidel jiných firem na základě spolupráce skrze spediční databanku RAALTRANS. Firma má k dispozici 8 vlastních vozů a využívá služeb přibližně stovky dalších dopravců.

O spediční služby firmy se starají dva spediční pracovnicí. Pokud k přepravě zboží není využito vlastních vozidel, zpravidla spediční pracovníci k přepravě pronajímají služby jiného dopravce, přičemž cena této služby je obvykle nižší než suma, kterou firmě platí za službu zákazník, rozdíl těchto cen pak představuje marži. Spediční pracovníci na základě údajů o přepravě vypisují objednávky (smlouvy o přepravě zboží), které elektronicky nebo faxem zasílají danému dopravci. K tomuto úkolu využívají pouze omezené programové vybavení, jak je popsáno v následující kapitole.

3.3. Průzkum prostředí

Přes svou velikost (viz předchozí kapitola) firma ROZLÍVKA TRANSPORT s.r.o. nevlastní žádný sofistikovaný software, který by usnadňoval proces vypisování objednávek a poskytoval rychlou zpětnou vazbu o stavu práce spedičního oddělení z finančního hlediska. Pro vypisování spedičních zakázek využívají pracovníci firmy nástrojů, které jim nabízí balík MS Office, zejména pak aplikace MS Word. Dále využívají služeb aplikace pro přístup do databáze RAALTRANS a aplikace Route66 pro vyhledávání tras, tyto však nijak neusnadňují proces vypisování objednávek. V dokumentu MS Word mají spediční pracovníci firmy před-vytvořenu tabulku, do které vyplňují patřičné údaje o objednávce. Těmito údaji jsou:

- **informace o dopravci** – název dopravce (tím je myšlen dopravce, u kterého spediční pracovníci firmy objednají danou přepravní zakázku), kontaktní telefon dopravce, fax dopravce, a RAAL kód dopravce.
- **číslo objednávky (smlouvy)** – představuje pořadové číslo objednávky za daný rok. Každý nový rok se objednávky číslují znovu od jedné.
- **údaje o nakládce** – místo nakládky (tj. adresa, kde je zboží naloženo), náklad (tj. informace o druhu zboží, váze, počtu atp.), datum a čas nakládky. Na jedné smlouvě mohou být až 3 místa nakládky a další informace na ně navázané.

- **údaje o vykládce** – místo vykládky (ekvivalentně k místu nakládky), datum a čas vykládky. Na jedné smlouvě mohou být až 3 místa nakládky a další informace na ně navázané.
- **poznámka –** v poznámce bývá zpravidla uvedeno kupříkladu, jak se má naložit s paletami po složení zboží atp.
- **cena** – představuje cenu, za kterou poskytne daný dopravce svoje přepravní služby.
- **spz** – představuje státní poznávací značku vozidla, jímž bude přeprava uskutečněna.
- **další položky** – objednávka obsahuje další prvky, mezi které patří kontaktní údaje na firmu ROZLÍVKA TRANSPORT s.r.o., její RAAL kód a údaje o podmínkách smlouvy.

Podoba objednávky vyplněné pomocí MS Word je uvedena v příloze.

Tento způsoby vypisování objednávek neumožňuje prakticky žádnou možnost automatizace, přestože z povahy procesu vypisování objednávek existuje vysoký potenciál pro usnadnění a zautomatizování některých prvků této práce při faktu, že v průměru spediční pracovníci za jeden den vypíší mezi 2 a 10 objednávkami.

Dále díky tomuto způsobu vypisování objednávek jsou hodnoty ceny a marže zaznamenávány pouze ručně na papír se všemi nevýhodami, které z toho plynou. Především tento způsob neumožňuje rychlý přehled souhrnných částek s ohledem na období nebo dopravce a veškeré případné žádané informace jsou zdlouhavě získávány pomocí sčítání na kalkulačce. Podoba způsobu vypisování informací o objednávce je uvedena v příloze.

Ve firmě jsou umístěny 3 pevné počítače. Tyto počítače jsou propojeny pomocí místní sítě LAN tak, že každý z nich má v rámci vnitřní sítě nastavenou pevnou IP adresu. Počítače v rámci firmy byly označeny jako PC 1, PC 2 a PC X. PC X představuje abstraktní počítač, tj. jakýkoli další PC připojený do sítě popřípadě třetí počítač instalovaný ve firmě, který převážně využívá účetní firmy, popřípadě ke kontrole stavu i ředitel firmy. Informace o počítačích zobrazuje tabulka (Tabulka 1).

Tabulka 1 - Informace o firemních počítačích. Zdroj Vlastní

Počítač	Uživatel	OS	Rozlišení monitoru	HW parametry
PC 1	Spediční prac. 1	Windows XP	1024x768 px	Procesor: AMD 7750 DUAL-CORE 2,7 GHz Paměť: 2 GB RAM
PC 2	Spediční prac. 2	Windows XP	1680x1050 px	Procesor: AMD SEMPRON 2200+ 1,5 GHz Paměť: 256 MB RAM

3.4. Cíle IS

Společně s pracovníky logistické firmy a na základě předběžného průzkumu prostředí logistické firmy byly stanoveny základní cíle, které má aplikace v rámci IS splnit.

- usnadnění vypisování objednávek přepravy zboží
- usnadnění získávání informací pro podporu manažerského rozhodování pomocí poskytnutí souhrnných i dílčích součtů ceny a marže přepravy zboží za jednotlivé roky, měsíce a dopravce.

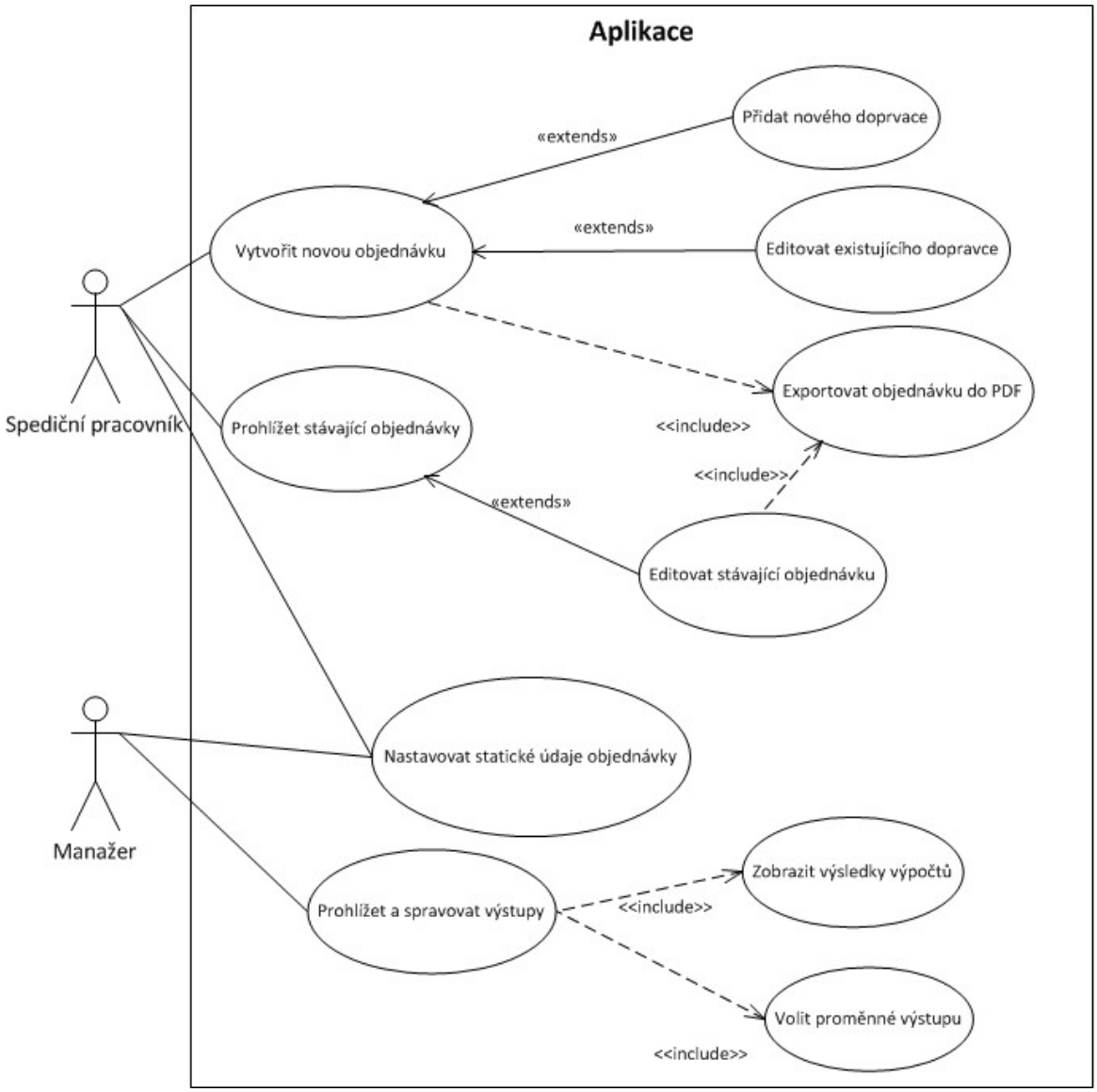
3.5. Požadavky na IS

Společně s pracovníky firmy a zadávajícím práce byly vymezeny základní požadavky na podobu a především funkce aplikace v rámci IS. Funkce aplikace ilustruje use-case diagram (Obrázek 6). Aktér na obrázku pojmenovaný jako „manažer“ symbolizuje především vedení firmy, resp. kohokoli, kdo má zájem na tom získávat souhrnné výstupy systému v souladu s druhým cílem IS.

3.5.1. Požadavky na funkce IS

- možnost částečného automatického vyplnění zadávacího formuláře pro vypisování objednávek přepravy, tj. automatické vyplnění čísla objednávky a údajů o dopravci na základě zvoleného dopravce
- schopnost ukládání informací do databáze
- možnost exportovat vyplněný formulář do tisknutelného souboru (PDF)
- možnost prohlížení údajů uložených v databázi
- schopnost automatického zpracování dat z databáze do souhrnných výstupů a to ve formě číselné i grafické (jedná se o zaznamenanou cenu a marži)

- možnost prohlížení souhrnných výstupů
- schopnost sdílení databáze dvěma paralelně běžícími aplikacemi na dvou PC pomocí síťové komunikace na principu architektury klient/server



Obrázek 6 – Use case diagram – funkce a využití aplikace. Zdroj vlastní

3.5.2. Nefunkční požadavky na IS

Na základě funkčních požadavků na IS a na základě průzkumu prostředí firmy, její personální struktury a technického vybavení byly stanoveny následující nefunkční požadavky na IS:

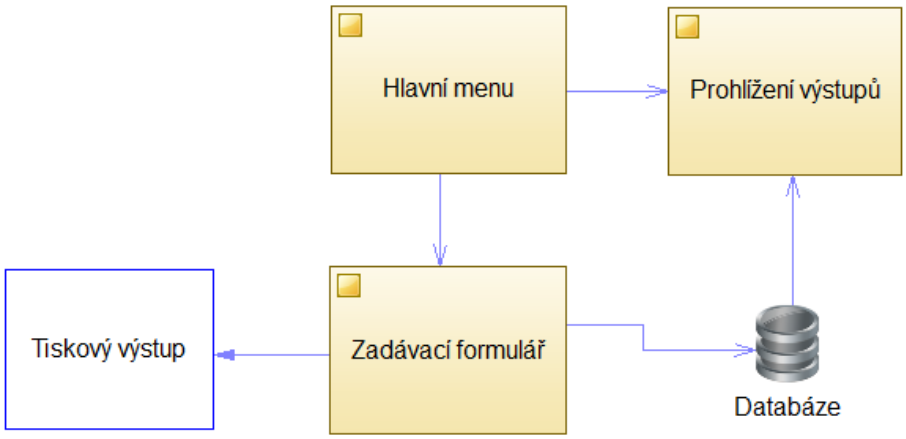
- síťová architektura klient/server
- snadno ovladatelné a přehledné uživatelské rozhraní designově odpovídající standardním desktopovým aplikacím
- výsledná objednávka po exportu nepřesahuje rozsah jedné strany A4
- podpora běhu na operačních systémech MS Windows

4. Návrh aplikace

Tato kapitola se zabývá návrhem aplikace v rámci IS.

4.1. Návrh aplikace a procesu vypisování objednávky

Na základě stanovených požadavků byla vytvořena základní struktura aplikace (Obrázek 7).



Obrázek 7 – Základní struktura aplikace. Zdroj vlastní

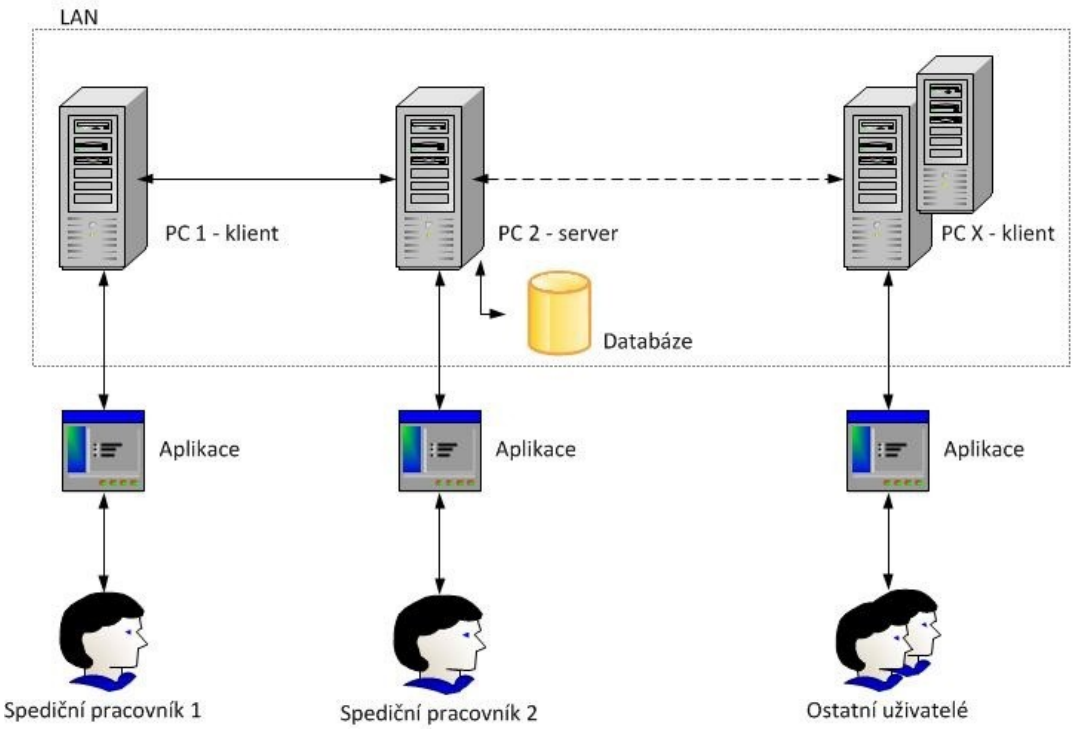
Jak je znázorněno na obrázku (Obrázek 7) struktura aplikace je navržena tak, aby uživatel mohl zvolit, zda chce přistoupit k zadávacímu formuláři nebo k prohlížení souhrnných výstupů. Zadávací formulář je pak možno uložit do databáze nebo vyexportovat do tisknutelného výstupu (PDF). Podoba zadávacího formuláře bude odpovídat položkám, které byly v původním stavu vyplňovány do dokumentu MS Word a které, bude obsahovat vyexportovaný dokument objednávky, k tomu je navíc přiřazena položka „marže“, která na tisknutelné podobě objednávky není, ale ukládá se do databáze za účelem tvorby souhrnných výstupů. Mezi výstupy jsou jednak zahrnuty finanční výstupy a jednak zobrazení položek z databáze. Detailně je práce s aplikací ve formě reprezentace pomocí procesů znázorněna v EPC diagramu (

Příloha 3). Při zobrazení položek z databáze je umožněno jednotlivé položky editovat v zadávacím formuláři a doplňovat nebo opravovat tak informace o objednávce. Část aplikace, která se zabývá poskytováním souhrnných součtů ceny a marže je uzpůsobena tak, aby umožnila vybrat, jaký z těchto dvou ukazatelů má být počítán a dále umožnila

součty a grafické zobrazení vývoje těchto ukazatelů v závislosti na vybraném dopravci (nebo všech dopravcích), měsíci a roce.

Jak bylo uvedeno v kapitole 3.5 Požadavky na IS, aplikace budou paralelně přistupovat ke sdílené databázi. Při tomto pojetí bude jeden z počítačů uchovatelem databáze na svém pevném disku a bude představovat server, ostatní počítače budou na tuto databázi přistupovat po síti jako klient. Do této sítě budou primárně zařazeny PC, na kterých pracují oba spediční pracovníci firmy, popřípadě kdokoli jiný v rámci vnitřní sítě (účetní, vedoucí, atp.), pro přístup k finančním výstupům a dalším funkcionalitám, které bude aplikace poskytovat, tito uživatelé jsou ve schématu zobrazeni jako „ostatní uživatelé“. Tuto architekturu zobrazuje schéma (Obrázek 8). Tento obrázek by se dal v podstatě považovat i jako základní schéma IS také proto, že obsahuje prvky, které patří mezi základní prvky obecného IS, tedy aplikaci (software), uživatele (lidé), počítače (hardware) a databázi (data).

Na základě průzkumu prostředí (kapitola 3.3 Průzkum prostředí) byl jako server zvolen PC spedičního pracovníka p. Vyčítala (v obrázku označen jako „Spediční pracovník 2“) a to především vzhledem k jeho hardwarovému vybavení. Přestože provoz serveru a databáze, při předpokládaném rozsahu několika (4-5) tabulek není hardwarově náročnou činností (s ohledem na tamní vybavení), zdá se jako logické pro tyto účely zvolit výkonnější z obou počítačů. Dalším významným faktem, který hovoří pro volbu tohoto počítače, je fakt (ujištění spedičními pracovníky), že PC spedičního pracovníka p. Vyčítala běží prakticky nepřetržitě celou pracovní dobu a nikdy nenastane situace, kdy by běžel pouze PC jeho kolegyně (v obrázku označena jako „Spediční pracovník 1“), tento fakt prakticky předurčuje použití PC p. Vyčítala jako serveru a úložiště databáze. V neposlední řadě hovoří pro volbu této struktury i fakt, že (podle slov spedičních pracovníků) až 90% objednávek přepravy je vypisováno právě p. Vyčítalem na jeho PC.



Obrázek 8 - Schéma architektury přístupu uživatelů do databáze. Zdroj vlastní

4.2. Návrh databáze

Databáze je navržena tak, aby v sobě uchovávala informace o objednávce přepravy. Jednotlivé tabulky a sloupce tedy odpovídají prvkům zadávacího formuláře, pro který je předobrazem původní dokument MS Word, ve kterém byly objednávky vypisovány. Položky této objednávky jsou podrobně popsány v kapitole 3.3 Průzkum prostředí. Oproti položkám, které obsahuje objednávka je do databáze přiřazeno několik dalších atributů především pro účely tvorby finančních výstupů. Jsou to atributy:

- **marže** – představuje rozdíl mezi cenou, za kterou je objednána přeprava zákazníkem u ROZLÍVKA TRANSPORT s.r.o. a cenou, za kterou spediční pracovníci objednají přepravní službu u jiného dopravce.
- **měsíc** – objednávky přepravy nejsou vystavovány ke konkrétnímu datu, proto je pro účely tvorby souhrnných výstupů použit atribut měsíc, který představuje měsíc, ve kterém byla objednávka vystavena.
- **cena v Kč** – cena, za kterou spediční pracovníci objednávají přepravní služby u jiných dopravců je proměnlivá položka nejen ve smyslu své hodnoty, ale i ve smyslu souvisejících atributů, cena může být vypisována v Eurech nebo v Kč, může být vypisována včetně DPH nebo bez DPH atp. Z tohoto důvodu je

pro účely tvorby souhrnných výstupů zaveden tento atribut. Představuje položku, která byla ručně vypisována na papír (viz kapitola 3.3 Průzkum prostředí).

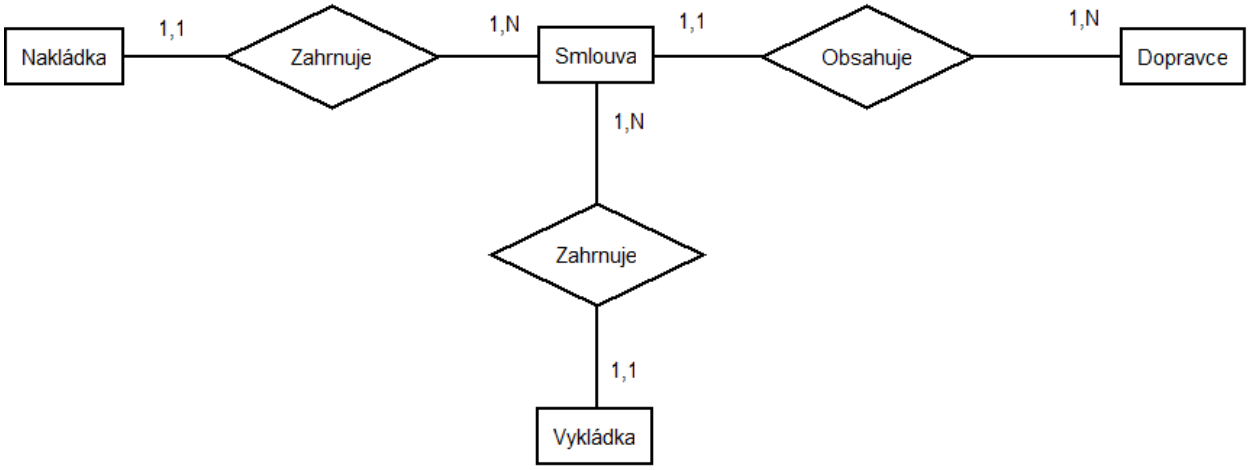
4.2.1. Konceptuální úroveň

Na konceptuální úrovni návrhu databázového modelu byl vytvořen seznam entit. Entita představuje jasně identifikovatelný a odlišitelný objekt reálného světa, který je schopen samostatné existence. [11]

Z návrhu vyplynuly čtyři entity (Tabulka 2). Jejich vzájemné vztahy jsou vyjádřeny pomocí ER diagramu (Obrázek 9). Entity představují části dokumentu smlouvy o přepravě a jejich význam je popsán v kapitole 3.3 Průzkum prostředí.

Tabulka 2 - Seznam entit. Zdroj vlastní

Název Entity	**Klíčový atribut**	**Atributy**
Smlouva o přepravě	Číslo smlouvy, rok	poznámka, cena, spz, měsíc, marže, cena v Kč
Dopravce	RAAL	název, telefon, fax
Nakládka	ID	místo, datum, čas, náklad
Vykládka	ID	místo, datum, čas



Obrázek 9 - ERD návrhu databáze. Zdroj vlastní

Vztahy mezi entitami, které jsou na obrázku (Obrázek 9) představují:

- **Zahrnuje** – Entita *Smlouva* zahrnuje *Nakládku* a *Vykládku*, *Nakládka* a *Vykládka* jsou zahrnuty ve smlouvě, přičemž *Smlouva* zahrnuje 1-N nakládek a 1-N vykládek.
- **Obsahuje** – Entita *Smlouva* obsahuje *Dopravce*, přičemž jeden dopravce může být obsažen až v N smlouvách.

4.2.2. Logická úroveň

Z návrhu vzniklého v konceptuální úrovni pomocí transformačních pravidel transformujeme ERD do relačního modelu databáze (RMD). Počet a struktura vzniklých relací je závislá na vzájemných vztazích entit, jejich kardinalitě a parcialitě, které udávají povinnost a četnost členství ve vztahu. První ze dvou čísel u dané entity vyjadřuje povinnost nebo nepovinnost členství ve vztahu (1 nebo 0) a druhé číslo stupeň četnosti (1 nebo N) (Obrázek 9). [11]

Po transformaci se entity rozpadly do čtyř relací (Tabulka 3). Primární klíče jsou vyznačeny tučně, cizí klíče kurzívou.

Tabulka 3 - Seznam relací. Zdroj vlastní.

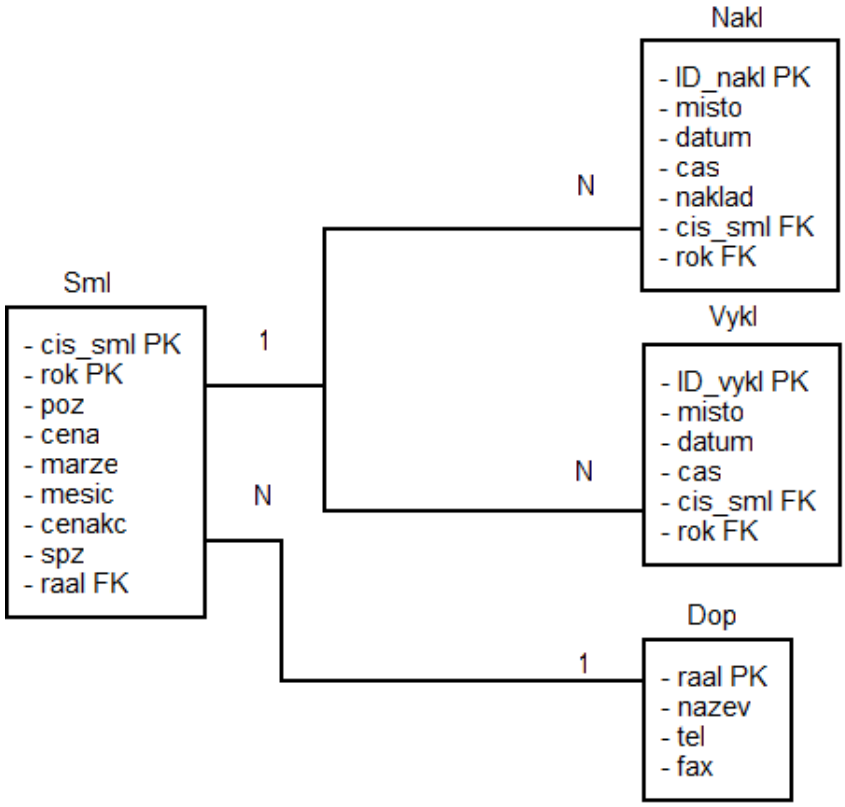
Název relace	**Atributy**
Sml	**cis_sml PK, rok PK**, poz, cena, mesic, marze, cenakc, spz, *raal FK*
Dop	**raal PK**, nazev, tel, fax
Nakl	**ID_nakl PK**, misto, datum, cas, naklad, *cis_sml FK, rok FK*
Vykl	**ID_vykl** PK, misto, datum, cas, *cis_sml FK, rok FK*

4.2.3. Normalizace

- 1NF – říká, že žádný atribut v relaci není vícehodnotový [11]. Pokud vzniklé relace podrobíme normalizaci 1NF. zjistíme, že 1NF splňují. Atribut **datum** by se sice dal dekomponovat na atributy **měsíc** a **den**, ale většina implementačních nástrojů relačního modelu databází poskytuje datový typ **datum**.

- 2NF – je splněna za předpokladu splnění 1NF a dále říká, že každý neklíčový atribut včetně kandidátních klíčů musí být plně funkčně závislý na celém primárním klíči [11]. Tato normální forma se tedy týká pouze relací se složeným primárním klíčem. V navrhovaném modelu má složený primární klíč pouze relace **sml**. Všechny atributy této relace jsou plně funkčně závislé na celém primárním klíči. Relační model tedy splňuje i 2NF.
- 3NF - splňuje relace, jestliže je v 2NF a každý neklíčový atribut je závislý na primárním klíči přímo, tedy není na primárním klíči závislý přes jiný atribut. Všechny relace splňují i tuto normální formu.

Po provedení transformace a normalizace byl navrhnut výsledný relační model dat (Obrázek 10).



Obrázek 10 - RMD databáze. Zdroj vlastní

4.3. Implementace databáze

Na základě návrhu byl jako implementační nástroj pro vytvoření databáze zvolen nástroj PHP MyAdmin. Jedná se o nástroj umožňující správu databáze MySQL pomocí webového rozhraní. Výhodou databázového systému MySQL je, že se jedná o volně

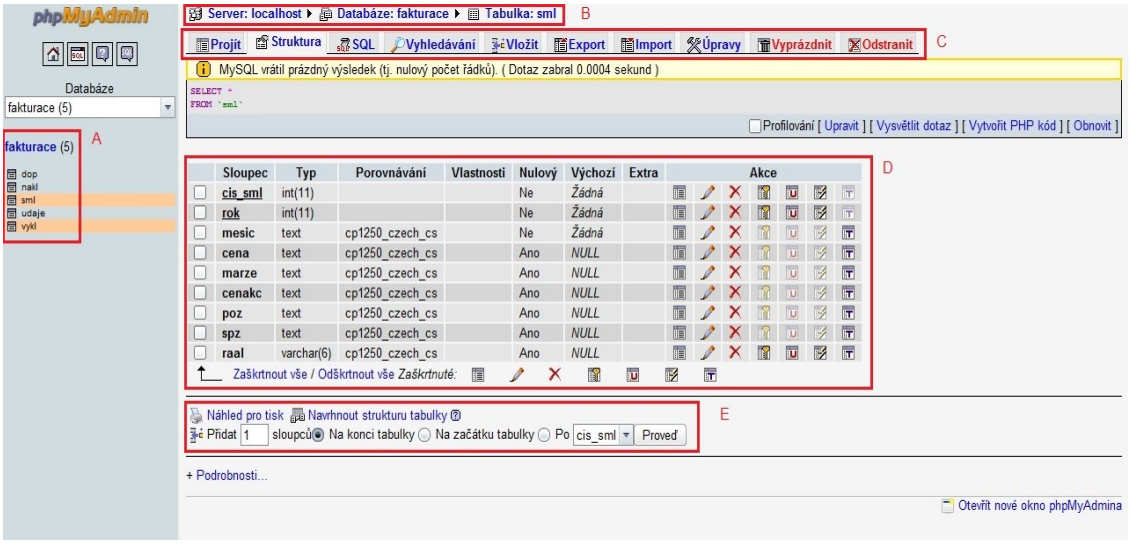
šiřitelný software. Pro rozsah databáze, která byla navrhnuta v předchozích krocích je nástroj PHP MyAdmin bohatě dostačující.

Při implementaci databáze v PHP MyAdmin se nastavují vlastnosti jednotlivých tabulek i atributů, nastavují se zde integritní omezení odpovídající implementační úrovni, které nabízí nástroj PHP MyAdmin tj. primární a cizí klíče, datové typy, maximální možnou délku, výchozí hodnoty a další. Strukturu a vlastnosti databáze v rámci implementační úrovně zobrazuje tabulka (Tabulka 4).

Tabulka 4 - Struktura databáze na úrovni implementace. Zdroj vlastní

Sloupec	Datový typ	Maximální délka	Nulový	Index
Tabulka sml				
cis_sml	INT		NE	PRIMARY
rok	INT		NE	PRIMARY
cena	DOUBLE		ANO	
marze	INT		NE	
cenakc	INT		NE	
poz	TEXT	1000	ANO	
spz	TEXT	12	ANO	
mesic	INT		NE	
raal	VARCHAR	6	NE	
Tabulka nakl				
id_nakl	INT		NE	PRIMARY
misto	TEXT	500	ANO	
naklad	TEXT	250	ANO	
datum	TEXT	20	ANO	
cas	TEXT	20	ANO	
cis_sml	INT		NE	
rok	INT		NE	
Tabulka vykl				
id_vykl	INT		NE	PRIMARY
misto	TEXT	500	ANO	
datum	TEXT	20	ANO	
cas	TEXT	20	ANO	
cis_sml	INT		NE	
rok	INT		NE	
Tabulka dop				
raal	VARCHAR	6	NE	PRIMARY
nazev	TEXT	250	ANO	
tel	TEXT	12	ANO	
fax	TEXT	12	ANO	

Integritní omezení uvedená v tabulce (Tabulka 4) musí být ošetřena i na úrovni práce s databází v rámci vytvořeného programu, aby se docílilo zpětné vazby na konání uživatele s možností napravení chyby. Uživatel musí mít kvalitní informace o chybě, která nastala, aby jí mohl řešit. Například pokud zadá velké množství znaků, které není pro danou položku do databáze možné uložit, musí o tom být v rámci programu informován, aby tento fakt mohl napravit. Integritní omezení na úrovni aplikace jsou popsány v kapitole 5.2.8 Integritní omezení a ošetření výjimečných událostí. Implementaci databáze pomocí nástroje PHP MyAdmin ilustruje obrázek (Obrázek 11).



Obrázek 11 - Prostředí nástroje PHP MyAdmin. Zdroj vlastní

Popis obrázku (Obrázek 11):

- A – Název databáze a vypsané jednotlivé tabulky.
- B – Cesta k aktuální tabulce
- C – Menu
- D – detailní struktura vybrané tabulky, položka „Sloupec“ představuje jednotlivé sloupce tabulky, v této tabulce v tuto chvíli není žádný záznam. Položka „typ“ představuje datový typ proměnné (int, text,…). Položka „Porovnávání“ představuje použitou znakovou sadu. Položka „Výchozí“ představuje výchozí hodnotu pro hodnoty daného sloupce. Pod položkou „Akce“ je pole tlačítek umožňující editaci daného sloupce.
- E – Menu pro přidávání sloupců do tabulky a další nástroje.

Do databáze „fakturace“ byla kromě výše uvedených tabulek začleněna i tabulka „udaje“. Jedná se o tabulku s pouze jedním záznamem, která uchovává statické informace exportované na objednávku jako je adresa firmy, kontaktní údaje atp. Pro uchování těchto informací mohlo být zvoleno i jiné formy, například textového souboru, nebo souboru XML, toto řešení však bylo vyhodnoceno jako nejsnazší, přičemž funkčnost je u všech řešení srovnatelná.

5. Implementace aplikace

Na základě průzkumu prostředí zejména pak hardwarových a softwarových prostředků firmy (kapitola 3.3 Průzkum prostředí) a požadavků na IS (kapitola 3.5 Požadavky na IS), byl jako implementační nástroj zvolen programovací jazyk Java. Těmto požadavkům by jako implementační nástroj vyhovovalo více jazyků, jazyk Java byl však z těchto jazyků vybrán i na základě dalších vlastností jazyka popsaných v této kapitole. Jedná se o moderní objektově orientovaný programovací jazyk, který v současnosti patří k nejpoužívanějším programovacím jazykům na světě.

5.1. Vybrané vlastnosti jazyka Java

V této podkapitole budou uvedena fakta o jazyce Java zejména taková, která jsou nezbytná pro porozumění použitých programovacích principů.

5.1.1. Objektová orientovanost

Programovací jazyk Java je plně objektově orientovaný (vyjma osmi základních datových typů). To v praxi znamená, že program vytvořený v jazyce Java se skládá z jedné nebo více tříd. [12] Třídy obsahují atributy a metody. Třídy, stejně jako u jiných objektově orientovaných programovacích jazyků, slouží jako forma pro vytvoření objektů. Objekty se někdy označují jako instance dané třídy nebo také referenční proměnné (protože je jejich hodnota předávána odkazem do paměti). V Javě jsou všechny třídy (ať už implicitně nebo explicitně) děděny ze třídy s názvem *Object*. [13] Třída *Object* je tedy přímým nebo nepřímým předkem všech tříd vytvořených v jazyce Java. Pokud vytvořená třída není zděděna z jiné třídy explicitně pomocí klíčového slova **extends** je implicitně děděna z třídy *Object*.

Jak z předchozího textu vyplývá, Java nabízí nástroje jako je dědění nebo polymorfismus. Podrobnější popis principů těchto nástrojů je však nad rámec této práce.

5.1.2. Základní datové typy

V jazyce Java rozeznáváme 8 základních (primitivních) datových typů. Jejich výčet a vlastnosti zobrazuje tabulka (Tabulka 5).

Tabulka 5 - Základní datové typy jazyka Java. Zdroj vlastní upraveno na základě [13]

název	klíčové slovo	velikost v paměti	hodnoty	typ
byte	byte	1B	od –128 do +127	celočíselný
short	short	2B	od –32768 do +32 767	
integer	int	4B	od –2147483648 do +2147483647	
long	long	8B	od –9223372036854775808 do +9223372036854775807	
float	float	4B	7-8 platných cifer	floating point
double	double	8B	15-18 platných cifer	
character	char	2B	65536 různých znaků	znakový
boolean	boolean	1b	2 hodnoty - true a false	logický

Datové typy **float** a **double** patří mezi datové typy s plovoucí desetinou čárkou (floating point).

5.1.3. Řídící struktury

Mezi řídící struktury zařazujeme řídící a iterační příkazy (podmínky a cykly), tedy příkazy, které umožňují měnit pořadí vykonávaných instrukcí a větvení programu. [12]

Nejdůležitějšími řídícími příkazy v Javě jsou jednoduchá podmínka a přepínač. Jednoduchá podmínka umožňuje větvení programu do dvou větví na základě vyhodnocení logického výrazu. Příkaz je volán klíčovým slovem **if** a jeho obecná syntaxe dle [13] je:

if(podmínka)příkaz_1
nebo
if(podmínka)příkaz_1 **else** příkaz_2

Přepínač je volán klíčovým slovem **switch** a umožňuje vícenásobné větvení programu.

V Javě stejně jako v mnoha dalších programovacích jazycích rozeznáváme tři druhy cyklů: cyklus s podmínkou na začátku, cyklus s podmínkou na konci a cyklus s pevným počtem opakování. Pro cyklus s podmínkou na začátku používáme klíčové slovo **while**,

pro cyklus s podmínkou na konci klíčových slov **do while**, a pro cyklus s pevným počtem opakování klíčové slovo **for**.

5.1.4. Grafické uživatelské rozhraní

Pro GUI (graphical user interface) existuje v předdefinovaných knihovnách Javy velké množství tříd, představujících komponenty běžně používané v operačních systémech založených na práci s okny, jako je MS Windows. Děděním z před-vytvořených tříd nebo vytvářením jejich instancí lze v Javě snadno docílit přívětivého grafického rozhraní pro uživatele zvyklého na prostředí MS Windows.

5.1.5. Další klíčové vlastnosti jazyka Java

- Java je case-sensitivní, to znamená, že rozlišuje malá a velká písmena v názvech proměnných i v klíčových slovech. Proměnná s názvem *okno* není stejná jako proměnná s názvem *Okno*.
- Pro textové řetězce existuje v Javě datový typ *String*. Přestože se nejedná o základní datový typ, patří třída String do balíčku, který je automaticky importován, proto deklaraci textové proměnné můžeme provádět explicitně.
- Java také umožňuje práci s poli a to vícerozměrnými, statickými i dynamickými (ArrayList).
- Java disponuje propracovaným systémem zachycování výjimek. V praxi to znamená, že pokud nějaká část kódu může vyvolat chybu, je programátor nucen ji předem ošetřit.

5.2. Vybrané problémy řešené při tvorbě apliakce

V této kapitole jsou podrobněji popsána řešení vybraných problémů nastalých při tvorbě aplikace v programovacím jazyce Java.

5.2.1. Spojení s databází

Způsob komunikace s databází, byl v jazyce Java navrhnut tak, aby byla možná komunikace se všemi často používanými SŘBD (Systém Řízené Báze Dat) jako jsou kupříkladu Oracle, MySQL, MS Access nebo MS SQL Server. Z toho důvodu bylo firmou Sun Microsystems (tvůrce jazyka Java) vytvořeno jednotné rozhraní pro přístup k relačním databázím s názvem JDBC (Java Database Connectivity). [12]

Ovladač JDBC je překladač, který převádí zprávy specifické pro každý databázový systém na zprávy, kterým rozumí interpret jazyka Java a naopak.

Při vytváření apliakce, jež je předmětem této práce, vyvstal také problém propojení programu s databází. Jak bylo uvedeno v kapitole 4.3. Implementace databáze, pro vytvoření databáze byl použit nástroj PHP MyAdmin, tedy databázový systém MySQL, postupů propojení databáze s aplikací bylo čerpáno s elektronického zdroje [15].

Pro spojení s databází MySQL se používá JDBC rozhraní určené pro databáze MySQL. Ovladač JDBC pro spojení s databází zpravidla poskytuje výrobce daného databázového systému, v tomto případě je to firma MySQL AB, ovladač je dostupný k volnému stažení na webových stránkách této firmy [14]. V této práci byl použit ovladač *mysql-connector-java-5.1.6*. Daný ovladač stačí stáhnout, umístit do složky se souborem *Databaze.java* (nebo jiným, který zabezpečuje spojení s databází) a v editoru jej přidat do *classpath*, tedy zaregistrovat jako používaný.

Předtím, než mohou být provedeny jakékoli operace nad databází, je nutné ovladač nahrát a registrovat. K tomu slouží příkaz *Class.forName*(”jméno_JDBC_ovladače”);. V konkrétním případě připojení k databázi MySQL může daný kód vypadat například tak, jak ilustruje vrchní část obrázku (Obrázek 12) řádek 18-25.

```
14  public class Databaze {
15
16      public static ResultSet spojeni(String query) {
17
18          try {
19              Class.forName("com.mysql.jdbc.Driver");
20          } catch (Exception e) {
21              System.out.println("Chyba driveru");
22              JOptionPane.showMessageDialog(null, "Nepodařilo se zaregistrovat ovladač databáze!",
23                                            "Chyba!", JOptionPane.ERROR_MESSAGE);
24              System.out.println(e.toString());
25          }
26          try {
27              Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/fakturace",
28                                                            "Finrod", "heslo");
29              Statement st = conn.createStatement();
30              ResultSet rs = st.executeQuery(query);
31              return rs;
32          } catch (Exception e) {
33              System.out.println("Chyba volani, neprihlaseno");
34              JOptionPane.showMessageDialog(null, "Nepodařilo se navázat spojení s databází!",
35                                            "Chyba!", JOptionPane.ERROR_MESSAGE);
36              System.out.println(e.toString());
37              return null;
38          }
39      }
40  }
```

Obrázek 12 - Ilustrace zdrojového kódu pro připojení k databázi MySQL. Zdroj vlastní

Volání metody *forName()* třídy *Class* musí být, jako většina operací nad databází umístěno v bloku **try{}**, který zachycuje případné nastání výjimky. O řešení výjimky se pak stará kód uvedený v následném bloku **catch{}**. V tomto případě bude mít výskyt chyby za následek zobrazení chybového okna s hlášením: „Nepodařilo se zaregistrovat ovladač databáze!“

Ve chvíli, kdy je ovladač nahrán a zaregistrován, je možné navázat spojení s databází. Registrace ovladače JDBC je vázána na třídu, která obsahuje metody pro práci s JDBC, tato třída se jmenuje *DriverManager* a obsahuje také metodu *getConnection*(), která zajišťuje připojení k příslušné databázi. Jako parametry metody *getConnection*() předáme adresu databáze, ke které se chceme připojit. V případě této práce je to databáze pojmenovaná **fakturace** a je umístěná na serveru, tedy na lokálním disku PC 2.

K tomu, abychom po připojení do databáze nad ní mohli provádět datové operace, slouží metoda *createStatement*(), která vytvoří instanci třídy *Statement*. Objekty třídy *Statement* disponují metodami *executeQuery*() a *executeUpdate*(), těm se jako parametr předává příslušný SQL dotaz. Metoda *executeQuery*() je určena pro dotazování databáze a výsledek dotazu vrací jako objekt rozhraní *ResultSet*, jak je vidět na obrázku (Obrázek 12). Oproti tomu metoda *executeUpdate*() slouží k zapisování do databáze a jako parametr se jí předávají SQL dotazy typu INSERT nebo UPDATE. Z rozdílnosti povahy dotazování databáze a zápisu do databáze vyplývá, že bylo nutno vytvořit dvě metody, které by zajišťovaly propojení s databází. Vedle metody *spojeni*(), jejíž zdrojový kód je zobrazen výše (Obrázek 12), byla vytvořena metoda *zapis*(), která se stará o zapsání do databáze. Tato metoda je procedurálního typu (tedy nevrací žádnou hodnotu), což se v jazyce Java provádí klíčovým slovem **void** v deklaraci metody. Argumentem metody *zapis*() stejně jako metody *spojeni*() je SQL dotaz ve formě textového řetězce. Těla obou metod jsou prakticky totožná, obě obsahují jak registraci ovladače tak připojení k databázi. Rozdíl je až v tom, že je volána metoda *executeUpdate*() instance třídy *Statement*, která se stará o zápis do databáze, tudíž nic nevrací. Klíčový rozdíl mezi oběma metodami je vidět na obrázku (Obrázek 13), kde je zobrazena část kódu metody *zapis*().

```
try {
    Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/fakturace",
                                                  "Finrod", "heslo");
    Statement st = conn.createStatement();
    st.executeUpdate(query);
```

Obrázek 13 - Demonstrace zápisu do databáze. Zdroj vlastní

Takto zapsané argumenty metody *getConnection*() (Obrázek 12 a Obrázek 13), tedy uvedení slova „localhost“ by umožňovali pouze připojení k databázi umístěné na lokálním disku, nikoli přístup k databázi umístěné na serveru, tedy na PC 2 v rámci vnitřní sítě LAN. V rámci zvoleného způsobu přístupu k databázi (kapitola 4.1 Návrh aplikace a proces) bylo tedy nutno adresu databáze volit dynamicky s ohledem na to z jakého počítače bude do databáze aplikace přistupovat. Toho bylo dosaženo vytvořením textového dokumentu, který obsahuje adresu serveru, z tohoto dokumentu je pak pomocí metody *getIP*() tato adresa načítá a metoda *getIP*() ji ve formě textového řetězce vrací jako svou návratovou hodnotu. Tento textový dokument je pak na každém PC, ze kterého je aplikace spouštěna, pojmenován stejně, ale jeho obsah může být různý.

Stejného efektu by bylo možno dosáhnout nahrazením slova „localhost“ přímo IP adresou serveru v argumentu metody *getConnection*(), protože zadání IP adresy vlastního počítače je ekvivalentní k zadání slova „localhost“. V takovém případě by bylo připojení řešeno staticky. Tento způsob řešení by však s sebou přinášel několik nevýhod z hlediska pružnosti a bezpečnosti, kupříkladu pokud by byla serveru přidělena jiná IP adresa, server byl přesunut na jiný počítač atp., vyžadovalo by to zásah do kódu programu. Z bezpečnostních důvodů pak kupříkladu PC 2 obsahuje v textovém dokumentu nikoli IP adresu serveru (tedy svou), ale nadále slovo „localhost“, to zaručuje spojení s databází i v případě, že by bylo znemožněno připojení k síti.

Poslední úpravou v argumentech metody *getConnection*() je pak explicitní nastavení znakové sady pomocí příkazu „characterEncoding=Cp1250“ připojeného přes otazník k názvu databáze. Tento postup je nutno provést, aby bylo možno do databáze zapisovat některé problematické znaky české abecedy.

Výslednou podobu metody *getConnection*() pak ilustruje obrázek (Obrázek 14)

```
Connection conn = DriverManager.getConnection("jdbc:mysql://"+Databaze.getIP()+
                                              ":3306/fakturace?characterEncoding=Cp1250",
                                              "Finrod", "heslo");
```

Obrázek 14 - Výsledná podoba metody GetConnection(). Zdroj vlastní

Rozhraní *ResultSet* disponuje mnoha metodami pro zpracování výsledku SQL dotazu. Mezi tyto metody patří kupříkladu metoda *next*(). Použití této metody je spojeno s případem kdy návratem SQL dotazu je větší počet položek (řádků, sloupců), příkladem takového dotazu může být dotaz: „SELECT * FROM dop“, který vrací celý obsah tabulky *dop*. Pokud je třeba přistupovat k jednotlivým položkám, využijeme metodu *next*() jako argument cyklu s podmínkou na začátku.

Metoda *next*() přesune kurzor o jeden řádek vpřed z jeho aktuální pozice. Implicitně je kurzor umístěn před první řádek, první volání metody nastaví první řádek jako aktuální, druhé volání nastaví jako aktuální druhý řádek, a tak dále. Metoda *next*() vrací hodnotu **false** ve chvíli, kdy je kurzor nastaven za poslední řádek. [16]

Pro přístup k meta-datům databáze slouží metoda *getMetaData*(), s jejíž pomocí se dá kupříkladu zjistit počet řádků nebo počet sloupců v tabulce.

5.2.2. Architektura klient/server

Jak bylo uvedeno v kapitole 3.5 Požadavky na IS a popsáno v kapitole 4.1 Návrh aplikace a proces aplikace přistupuje k datům uloženým v databázi na principu architektury klient/server. Dále tamtéž bylo uvedeno, že vnitřní síť (LAN) firmy je poskytovatelem internetu koncipována tak, že v rámci LAN má každý počítač přidělenou pevnou IP adresu. Tento fakt umožňuje na úrovni aplikace přistupovat do databáze umístěné na serveru ekvivalentním způsobem, jako kdyby byla umístěna na lokálním disku (podrobně v kapitole 5.2.1 Spojení s databází). Jediným rozdílem pak, je, že v textovém souboru, který obsahuje adresu serveru není uvedeno klíčové slovo „localhost“, ale IP adresa serveru, v tomto případě tedy PC 2.

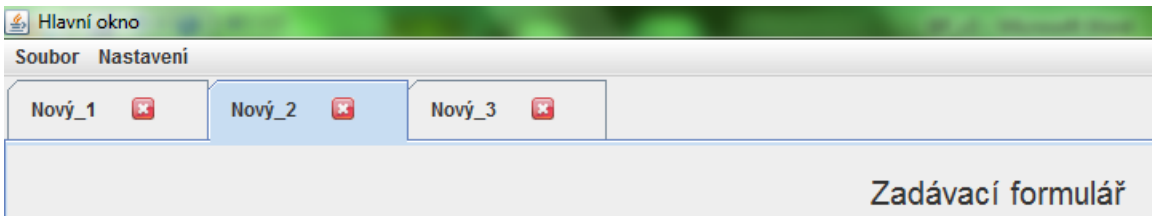
5.2.3. Hlavní okno

Po spuštění aplikace se otevře okno aplikace pojmenované „Hlavní okno“. Toto okno obsahuje pouze horizontální menu a obrázek s logem firmy ROZLÍVKA TRANSPORT s.r.o. Toto okno funguje pouze jako jakási úvodní stránka, tedy brána k dalším funkcím

aplikace. Horizontální menu funguje klasicky jako u většiny desktopových aplikací a obsahuje dvě položky: „Soubor“ a „Nastavení“. Položka „Nastavení“ obsahuje pouze jedinou položku a to „Údaje o výstupu“ skrze ni se nastavují statické údaje tiskového výstupu, jako je adresa firmy, e-mail apod. Položka „Soubor“ pak zahrnuje následující položky:

- **„Nový...“** – zvolení této položky zobrazí malé okno, ve kterém uživatel volí, jestli chce otevřít nový zadávací formulář, nové okno pro správu databáze nebo nové okno finančních výstupů
- **„Uložit jako**...“ – tato volba je přístupná až ve chvíli kdy je aktivní nějaký zadávací formulář a umožňuje uživateli vytvořit z vyplněného formuláře výstup v podobě souboru PDF
- **„Otevři z databáze“** – volba této položky je ekvivalentní k volbě nového okna pro správu databáze
- **„Uložit do databáze“** – tato volba je opět přístupná až pro otevřený zadávací formulář a umožňuje jeho uložení do databáze.

Při otevření nového zadávacího formuláře je v hlavním panelu okna otevřen zadávací formulář jako záložka, přičemž takto může být otevřeno i více panelů, ty jsou pak dynamicky pojmenovávány, jak ilustruje obrázek (Obrázek 15).



Obrázek 15 - Otevírání panelů zadávacího formuláře. Zdroj vlastní

5.2.4. Zadávací formulář pro výpis objednávek

Jedním ze dvou stěžejních cílů systému, tak aby byl žádaným přínosem, byl stanoven požadavek na usnadnění práce vypisování objednávek přepravy. Jako jeden z prvků aplikace byl proto vytvořen zadávací formulář pro výpis objednávek.

Strukturu tohoto formuláře a jeho grafickou podobu určuje především podoba tabulky v dokumentu MS Word, která byla používána spedičními pracovníky pro vypisování objednávek. Vyplněná a vytištěná podoba takto vyplněné objednávky je uvedena v příloze.

Formulář obsahuje všechny položky tak, jak již byly popsány v kapitole 3.3 Průzkum prostředí. Navíc obsahuje položky „Marže“ a „Cena v Kč“, ty nejsou součástí tisknutelné verze objednávky, ale jsou na formulář zařazeny, protože tyto údaje jsou použity pro tvorbu souhrnných výstupů.

Pro vyplnění jednotlivých položek je použito základních grafických komponent jazyka Java, které nabízí knihovna Swing. Těmito komponentami jsou:

- **JTextField –** neboli textové pole je klasický formulářový prvek. Jedná se o jednořádkové pole libovolné délky umožňující uživateli zadávat text.
- **JLabel –** slouží jako popisek a jeho text je uživatelem needitovatelný.
- **JTextArea –** textová oblast je podobná jako textové pole, ale může zabírat libovolný počet řádků.
- **JComboBox-** rozevírací seznam. Umožňuje výběr jedné z předem vyplněných položek. Skládá se z textové části a malého tlačítka se šipkou nalevo od textu. Po kliknutí na tuto komponentu se rozevře seznam vybratelných položek. Tato komponenta není uživatelem přímo editovatelná.
- **JSpinner** – jedná se o komponentu, která se skládá z textového pole a dvou tlačítek se šipkami směřujícími nahoru a dolu nalevo od textového pole. Slouží k záznamu číselných hodnot. Pomocí tlačítek je možno snižovat nebo zvyšovat číslo v textové části o předem daný krok, horní a spodní číselná hranice je také předem nastavitelná. Textová část je přímo editovatelná uživatelem.
- **JButton** – tlačítko. Tato komponenta představuje klasické tlačítko, na které je typicky vázána nějaká událost.

Jaké komponenty byly použity pro zobrazení a editaci položek formuláře shrnuje tabulka (Tabulka 6). Sloupec „Výchozí hodnota“ představuje výchozí hodnotu pouze prázdného formuláře, formulář otevřený z databáze je vyplněn údaji z databáze.

Tabulka 6 - Komponenty zadávacího formuláře. Zdroj vlastní

Položka	Komponenta	Výchozí hodnota
číslo smlouvy	JTextField	dynamicky z databáze
rok smlouvy	JSpinner	aktuální rok
dopravce	JComboBox	první dopravce z databáze dle abecedy
telefon dopr.	JTextField	dynamicky dle dopravce
fax dopr.	JTextField	dynamicky dle dopravce
RAAL dopr.	JTextField	dynamicky dle dopravce
místo nakládky	JTextArea	prázdné textové pole
den nakládky	JSpinner	aktuální den
měsíc naklákdy	JSpinner	aktuální měsíc
rok nakládky	JSpinner	aktuální rok
čas nakládky	JTextField	prázdné textové pole
náklad	JTextField	prázdné textové pole
místo vykládky	JTextField	prázdné textové pole
den vykládky	Jspinner	aktuální den
měsíc vykládky	JSpinner	aktuální měsíc
rok vykládky	JSpinner	aktuální rok
čas vykládky	JTextField	prázdné textové pole
poznámka	JTextArea	prázdné textové pole
cena	JTextField	prázdné textové pole
marže	JTextField	hodnota "0"
cena v Kč	JTextField	hodnota "0"
spz	JTextField	prázdné textové pole

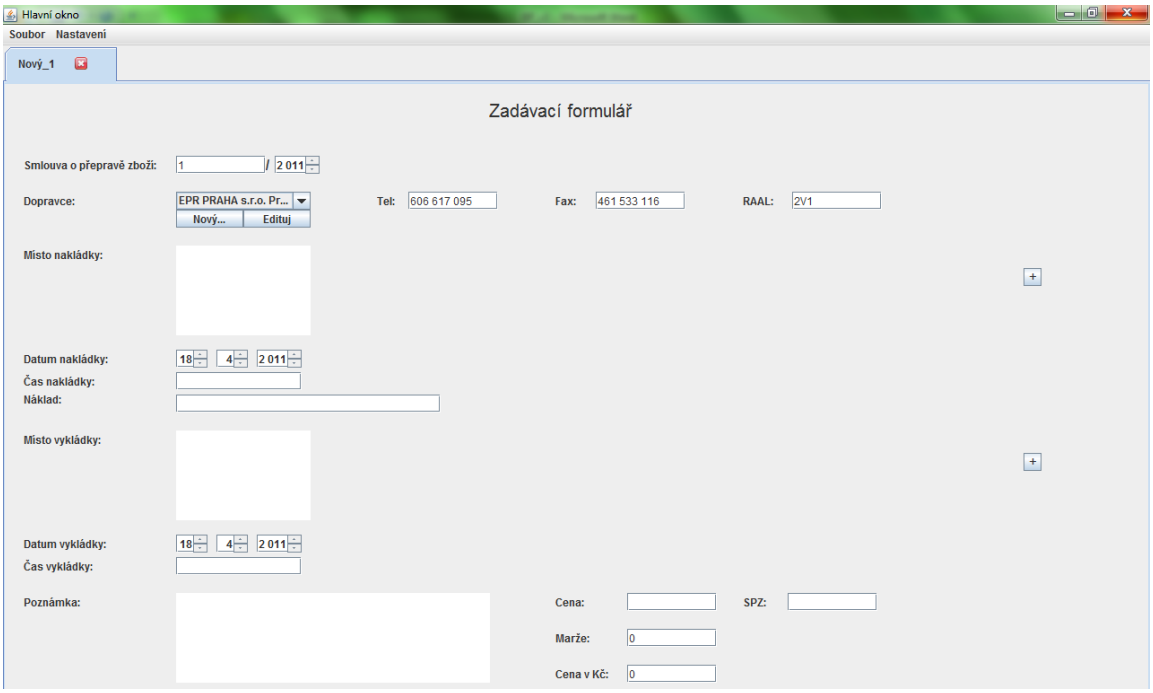
U komponent, kde je vyplňován aktuální časový údaj (rok, měsíc, den), je použito datového typu třídy *Date*. Z něho je pak pomocí textových funkcí separován aktuální den, měsíc nebo rok.

Seznam dopravců do rozevíracího seznamu je načítán z databáze z tabulky *dop* a abecedně seřazen. Při výběru konkrétního dopravce, se do textových polí telefon, fax a RAAL automaticky z databáze načtou příslušné hodnoty dle zvoleného dopravce. Při otevření nového formuláře je implicitně nastaven první dopravce dle abecedního řazení. Pro urychlení volby příslušného dopravce (vzhledem k jejich počtu) je poté možno použít počátečního písmene dopravce pro rychlé přesunutí v seznamu.

Položky „marže“ a „cena v Kč“ mají přednastavenou hodnotu nula. Důvod, proč zrovna tyto dvě položky, mají přednastavenou hodnotu na „0“, když jiné položky jsou prázdné, je takový, že v době vypisování objednávky spediční pracovník tyto hodnoty zpravidla nezná a doplňuje je později (což umožňuje editace objednávky z databáze).

Zároveň pokud by do databáze byla uložena u těchto položek prázdná hodnota, nebylo by možno z nich pak počítat automatické součty. Aby spediční pracovník pokaždé, když v danou chvíli nezná hodnotu těchto položek, nemusel do nich vpisovat nulu, je tato hodnota před-vyplněna. V databázi pak existence této hodnoty nijak neznemožní tvorbu souhrnných výstupů.

Pokud uživatel nenalezne v rozevíracím seznamu požadovaného dopravce (ještě nikdy nebyl použit), je třeba ho přidat do databáze. K tomuto účelu je pod komponentou rozevíracího seznamu umístěno tlačítko s nápisem „Nový...“. Po kliknutí na toto tlačítko otevře se okno s formulářem pro vyplnění údajů nového dopravce. Toto okno obsahuje položky pro vyplnění údajů o dopravci, tedy jeho název, RAAL kód, telefon a fax. Poté, co uživatel tyto údaje vyplní, klikne na tlačítko s nápisem „Uložit do Databáze“a údaje o dopravci jsou poté uloženy do databáze. Při tomto procesu je testováno, zda dopravce se stejným názvem nebo RAAL kódem již neexistuje. V takovém případě by na tento fakt byl uživatel upozorněn dialogovým oknem, přičemž dostane na výběr, zda chce údaje o takovém dopravci přepsat, nebo nikoli a pokračovat v editaci. Na podobném principu funguje i proces úpravy údajů o vybraném dopravci. K tomuto účelu slouží tlačítko s nápisem „Edituj“ umístěné vedle tlačítka „Nový...“. Po kliknutí na toto tlačítko se otevře stejné editační okno, jako v prvním případě, pouze jeho položky, tedy údaje o dopravci, jsou předvyplněny na základně vybraného dopravce (aktuálně zvoleného dopravce v rozevíracím seznamu). Při změně údajů o dopravci nebo při přidání nového dopravce se tato změna propíše do všech otevřených panelů zadávacích formulářů. Výslednou podobu formuláře zobrazuje obrázek (Obrázek 16).



Obrázek 16 - Výsledná podoba zadávacího formuláře. Zdroj vlastní

Aby si vzhled zadávacího formuláře zachoval přehlednost i při rozdílných rozlišeních monitorů jednotlivých počítačů, jsou komponenty do okna umisťovány relativně s ohledem na rozlišení monitoru v pixelech. K zjištění rozlišení monitoru slouží metoda *getScreenSize*().

Protože (jak bylo uvedeno v kapitole 3.3 Průzkum prostředí) údajů o nakládce a údajů o vykládce může být 1-3, je tomuto faktu uzpůsoben i zadávací formulář. Po otevření nového formuláře jsou na něm zobrazeny pouze údaje o jedné nakládce a vykládce. Pokud by uživatel potřeboval vypisovat údaje o více nakládkách nebo vykládkách, je při pravém okraji formuláře umístěno tlačítko „+“ vždy pro vykládku i nakládku. Po kliknutí na toto tlačítko se zobrazí další skupina údajů o nakládce (resp. vykládce). Pokud by uživatel na toto tlačítko klikl omylem, nebo si svou volnu rozmyslel, nic se neděje, do tiskového výstupu se tento fakt neprojeví, dokud uživatel nevyplní pole „místo nakládky“ (resp. vykládky). Volbu nového místa nakládky (resp. vykládky) zobrazuje obrázek (Obrázek 17).

Obrázek 17 - Přidání dalšího místa nakládky a vykládky do formuláře. Zdroj vlastní

5.2.5. Okno správy databáze

Aby měl uživatel možnost prohlížet přehledně zobrazené objednávky přepravy a mohl je zpětně editovat zejména pak za účelem doplnění marže a ceny v Kč, obsahuje aplikace okno správy databáze.

Jedná se o okno, v němž jsou vypisovány objednávky seřazené podle svého čísla v jednotlivých letech a základní údaje, tak aby výsledný soupis nesl potřebné informace a zároveň si zachoval přehlednost. Po pravé straně každé položky je pak tlačítko „Editace“. Po stisknutí tohoto tlačítka se otevře panel zadávacího formuláře a jeho položky se vyplní odpovídajícími daty z databáze. Zobrazení položek ilustruje obrázek (Obrázek 18).

Obrázek 18 - Výpis objednávek z databáze. Zdroj vlastní

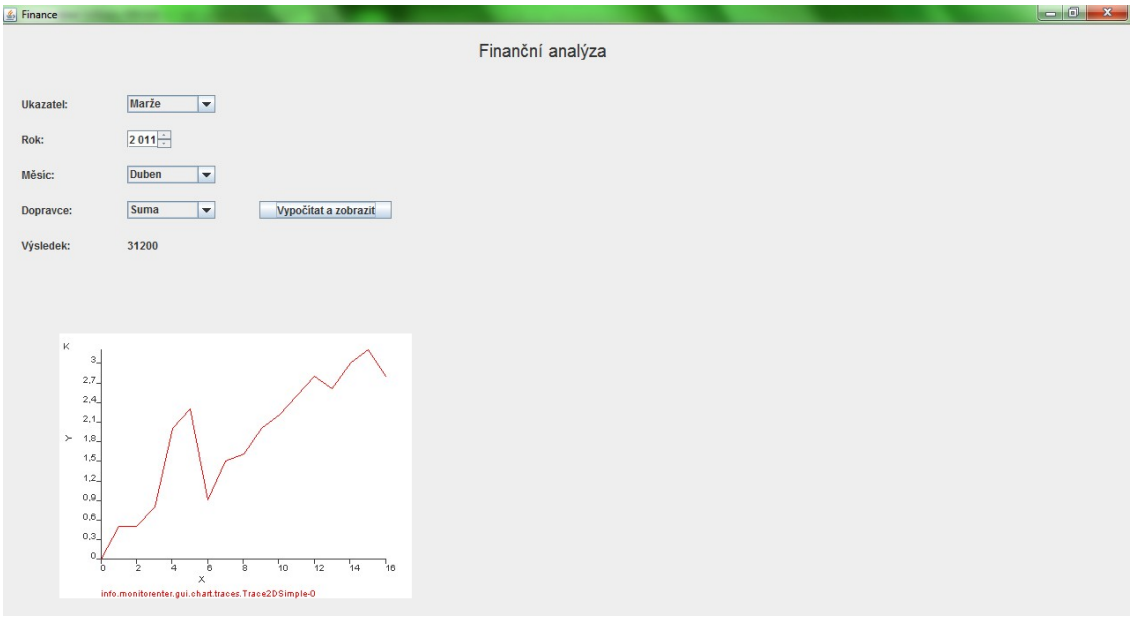
5.2.6. Okno správy finančních výstupů

Druhým z hlavních cílů systému a jeho přínosnosti byla stanovena tvorba finančních výstupů tak, aby byl usnadněn proces sčítání cen a marží a tím byla vytvořena podpora pro pružnější manažerské rozhodování.

K těmto účelům slouží okno „správy finančních výstupů“. Okno se skládá z položek, ve kterých jsou nastaveny atributy pro výpočet ukazatele. V těchto položkách uživatel vybírá, zda chce počítat souhrn ceny nebo marže, za jaký měsíc v daném roce chce výpočet

provést a zda chce výpočet provést pro jednoho konkrétního dopravce nebo souhrnně pro všechny dopravce. Dále okno obsahuje komponentu grafu, která umožňuje grafické zobrazení průběhu vývoje ukazatele v daném měsíci. Tato komponenta nese název Chart2D a jedná se o komponentu externí knihovny JChart2D, která je volně stažitelná na [17].

Ve chvíli, kdy uživatel nastaví položky, stiskne tlačítko „Vypočítat a zobrazit“, poté se v okně zobrazí výsledek v číselné podobě a dále také v grafické podobě prostřednictvím komponenty Chart2D. Podobu výběru atributů výpočtu a zobrazení výsledků ilustruje obrázek (Obrázek 19).



Obrázek 19 – Okno správy finančních výstupů. Zdroj vlastní

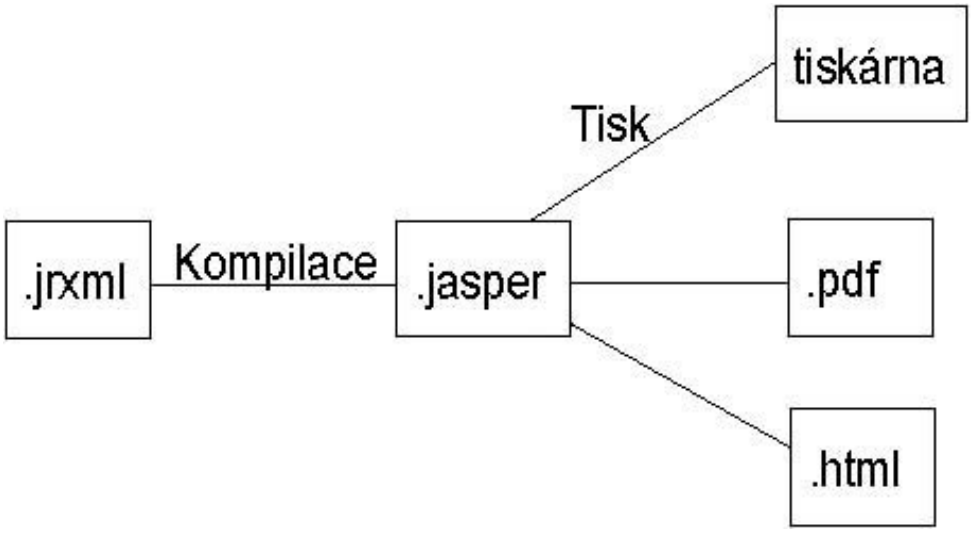
5.2.7. Řešení tiskových výstupů

Mezi funkční požadavky na IS byl zařazen i požadavek, že aplikace musí umožňovat export formuláře do tisknutelného souboru (kapitola 3.5.1 Požadavky na funkce IS). API programovacího jazyka Java nástroj pro přímou tvorbu formátovatelného tiskového výstupu neobsahuje. Z tohoto důvodu bylo pro tvorbu tiskových výstupů využito knihovny funkcí s názvem JasperReports, ta je volně dostupná na internetu na [18] a popis základní práce s touto knihovnou je dostupný na [19].

JasperReports je systém, který podle vstupní šablony produkuje výstup. Vytvoření každé sestavy se dle [19] skládá ze čtyř základních kroků:

- vytvoření vstupní šablony ve formátu .jrxml
- zkompilování šablony do souboru .jasper
- vložení vstupních dat do sestavy
- volání metod z JasperReports API pro tisk sestavy.

Postup vytvoření tiskové sestavy ilustruje obrázek (Obrázek 20).



Obrázek 20 - Postup vytvoření tiskové sestavy pomocí JasperReports. Zdroj [19]

Soubor .jrxml je v podstatě XML soubor, který popisuje to, jak má tisková sestava vypadat a co má obsahovat. Tento soubor se dá editovat klasicky jako každý XML soubor podle daných pravidel, to však vyžaduje značné odborné znalosti, proto je pro účely snadné editace šablony pro tvorbu tiskového výstupu možno použít editační nástroj iReport, který je volně stažitelný na [18].

Šablona vytvořená pomocí nástroje iReport je poté používána k tvorbě tiskového výstupu do souboru PDF. O tento proces se již stará vytvořená aplikace. Jak je tvorba výstupu do PDF řešena v kódu jazyka Java ilustruje obrázek (Obrázek 21).

```
99          JasperReport jr = JasperCompileManager.compileReport(
100                 "strep.jrxml");
101         JasperPrint jp = JasperFillManager.fillReport(jr, params,
102                                                 new JREmptyDataSource());
103         JasperExportManager.exportReportToPdfFile(jp,cesta +".pdf");
```

Obrázek 21 - Tvorba výstupu do PDF. Zdroj vlastní

Nejprve je nutno vytvořit instanci třídy *JasperReport*, které předáváme zkompilovanou šablonu. O kompilaci se stará metoda *compileReport*() třídy *JasperCompileManager*, která

jako svůj argument přebírá cestu a název šablony v tomto případě pojmenované „strep.jrxml“. Dále je třeba vytvořit instanci třídy *JaspePrint*, o to se postará metoda *fillReport*() třídy *JasperFillManager*. Tato metoda přebírá jako argument jednak vytvořenou instanci třídy *JasperReport* a jednak parametry, pomocí kterých je možno do tiskových výstupů promítnout dynamické položky objednávky. Předání instance třídy *JREmptyDataSource* jako posledního argumentu metody *fillReport*() pak představuje fakt, že report používá jako dynamických položek pouze parametry, ale žádné proměnné (proměnné slouží k dodatečným výpočtům v rámci kompilace šablony).

O to aby se provedl export do souboru typu PDF se stará třída JasperExportManager konkrétně její metoda exportReportToPdfFile(), která přijímá dva argumenty. Prvním argumentem je vytvořená instance třídy JaspePrint a druhým je cesta k adresáři, do kterého má být výstup uložen. Tato cesta je volena dynamicky uživatelem pomocí komponenty JFileChooser, kterou nabízí knihovna Swing. Podoba výsledného PDF souboru, tak jak je vyplněn v zadávacím formuláři a poté exportován, je uvedena v příloze.

5.2.8. Integritní omezení a ošetření výjimečných událostí

Integritní omezení na úrovni databáze popisuje Tabulka 4 v kapitole 4.3 Implementace databáze. Aby však byla aplikace bezpečná a nedocházelo k chybám při komunikaci aplikace s databází, je třeba integritní omezení ošetřit i na úrovni aplikace. Díky způsobu ošetření výjimek v jazyce Java, které nutí programátora explicitně v kódu ošetřit místa, která by mohla způsobit pád aplikace, není nebezpečí, že by některá chyba způsobila kolaps programu. To zároveň umožňuje na předpokládané výjimečné události v běhu programu reagovat. Mezi tyto události například patří událost, kdy by uživatel pokusil do databáze uložit smlouvu, která již existuje. Při každém uložení smlouvy je tedy třeba testovat, zda se smlouva s daným číslem a daným rokem již v databázi nenachází. Pokud ano, je uživateli nabídnuta volba, zda chce danou smlouvu přepsat, nebo dál pokračovat v editaci. Ekvivalentním způsobem je řešena i situace, kdy je do databáze zadáván nový dopravce. Také ve chvíli, kdy by z neočekávaných důvodů selhala komunikace s databází, nenastane pád aplikace nebo jiné její selhání, ale o tomto faktu je uživatel informován dialogovým oknem.

Integritní omezení s hlediska databáze jsou jednak řešeny na úrovni testování výstupů kritických komponent, které by mohly způsobit konflikt při zápisu do databáze a jednak

samotným nastavením některých komponent, které neumožňují jiný výstup než, ten který očekává na svém vstupu databáze z hlediska typu proměnné, hodnoty i délky.

Výstup je testován u komponent „číslo smlouvy“, „marže“ a „cena v Kč“. Pokud by do těchto polí formuláře uživatel zadal jinou než číselnou hodnotu v požadovaném tvaru nebo nezadal žádnou hodnotu, aplikace uživateli neumožní smlouvu uložit do databáze a na problém jej upozorní pomocí dialogového okna. Toho je dosaženo pomocí převodu textového výstupu komponent *JTextField* na celočíselný typ *Integer*. V případě, že by uživatel zadal do komponenty číslo v neodpovídajícím tvaru nebo zadal text popřípadě nezadal nic (před-vyplněnou hodnotu smazal), vyvolá se při běhu programu výjimka, která je v programu zachycena a ošetřena zobrazením dialogového okna.

Co se týče délky textových řetězců jako výstupu z komponent formuláře, je komponentám *JTextField* nastavena maximální možná délka textu na číslo odpovídající integritnímu omezení na úrovni databáze.

5.3. Možnosti dalšího rozšíření

Pokud by se odstupem času, po dlouhodobějším používání systému (řádově měsíce), zdálo vhodné do aplikace implementovat další části (dne rozhodnutí pracovníků firmy), byla navržena tato možná rozšíření aplikace:

- **rozšíření možností souhrnných výstupů –** do okna souhrnných finančních výstupů by mohly být přidány i další souhrnné údaje, jako kupříkladu celkový počet vypsaných objednávek na jednoho konkrétního dopravce atp.
- **možnost přímého tisku formuláře** – tato možnost nebyla do aplikace řazena z důvodu toho, že exportovaný PDF soubor je nejen tisknut, ale i odesílán elektronickou formou a popřípadě v této podobě uchováván. Aby tato funkce (přímý tisk) mohlo být efektivně využívána, musela by aplikace patrně umožňovat zobrazení náhledu tisku.
- **možnosti dalšího nastavení** – mezi tyto možnosti by se dalo zařadit kupříkladu nastavení vzhledu aplikace (barvy rámů atp.), možnost dynamické volby zobrazovaných položek databáze (tj. dynamická úprava zobrazované tabulky kupříkladu odebírání a přidávání sloupců) a další. Rozšíření aplikace o tyto možnosti by však příliš nezvyšovalo její efektivitu a přínos v rámci stanovených cílů.

6. Zavedení aplikace v prostředí firmy

Pro zavedení systému do provozu firmy bylo použito strategie, která se nejvíce přibližuje strategii postupného zavádění. S pracovníky firmy byly konzultovány a odzkoušeny dílčí části programu, přičemž bylo navrženo několik úprav aplikace dle požadavků pracovníků a tyto úpravy pak byly v aplikaci realizovány.

Aby mohl být vytvořený systém provozován, bylo nutno programové vybavení počítačů firmy doplnit o některé softwarové komponenty. Na všechny počítače, kde byla aplikace instalována, muselo být nainstalováno (pokud již nebylo) běhové prostředí jazyka Java pro daný operační systém, v případě všech počítačů tedy prostředí pro běh na systému MS Windows. Běhové prostředí jazyka Java (JRE – Java Runtime Environment) je volně dostupné na adrese „http://java.sun.com“ a jeho velikost je 15,7 MB (verze 1.6.0.24), jeho instalace tedy nepředstavovala pro firmu žádnou zátěž.

Pro potřeby běhu systému byl na PC 2 nainstalován programový balík WampServer, což je jedna z mnoha distribucí, která v sobě integruje nástroje Apache server, PHP, MySQL, phpMyAdmin a další rozšíření. Tento program je dostupný jako freeware. Díky tomuto softwarovému prostředku bylo možno na PC 2 implementovat databázi.

Zástupce spouštěcí ikony programu WampServer byl umístěn do složky „Po Spuštění“ lokálního disku PC 2, čímž bylo docíleno stavu, kdy pracovník firmy nemusí explicitně spouštět po zapnutí počítače server, ale tato činnost se provede automaticky.

Přístup na server umístěný na PC 2 byl blokován systémovým firewallem, proto bylo pro zajištění průchodu komunikace povolit komunikaci pro port 3306, na kterém komunikuje aplikace s databází, resp. serverem na kterém je umístěna.

Veškeré zdrojové kódy, exportovaná databáze i zkompilovaná aplikace je umístěna na přiloženém CD.

Při zavádění aplikace se nevyskytly žádné významné problémy, pouze připomínky k podobě zadávacího formuláře, které byly na základě přání pracovníků opraveny. Aplikace začala být ihned po své implementaci pracovníky firmy využívána. Aplikace splňuje všechny vytyčené požadavky na její funkce i požadavky nefunkční. Lze tedy objektivně říci, že je pro pracovníky firmy přínosem a plní svůj účel.

Závěr

Úkolem této bakalářské práce bylo poskytnout podklad pro řešení problému nedostatečného využití ICT ve firmě ROZLÍVKA TRANSPORT s.r.o., která se zabývá logistikou. Dále bylo úkolem, na základě informací získaných v dané firmě, navrhnout a vytvořit aplikaci jako součást informačního systému, která by usnadnila spedičním pracovníkům firmy sepisování objednávek a evidenci a management firmy podpořila v jeho rozhodovacích procesech poskytnutím automatických součtů některých finančních ukazatelů.

Na základě průzkumu provedeného ve firmě a popsaného v této práci byla navržena a vytvořena aplikace v rámci IS dané firmy dle stanovených cílů. Tato aplikace pak byla ve firmě implementována a v současné době (24. 4. 2011) je plně funkční a je pracovníky firmy využívána ke všem účelům, ke kterým byla vytvořena. Na základě těchto faktů, lze tedy konstatovat, že cíle této práce byly splněny.

Použitá literatura

[1] KŘUPKA, Jiří. Teorie Systémů. Pardubice: Univerzita Pardubice, 2006. 140 s. ISBN 80 – 7194 – 923 – X.

[2] *psp.cz* [online]. 2010 [cit. 2010-12-05]. Zákon č. 256/1992 Sb., o ochraně osobních údajů**.** Dostupné z WWW:
< http://www.psp.cz/eknih/1990fs/tisky/t1465_00.htm >

[3] *pristupnost.cz* [online]. 2010 [cit. 2010-12-05]. Zákon 365/2000 Sb., O informačních systémech veřejné správy**.** Dostupné z WWW:
< http://www.pristupnost.cz/zakon-365-2000-sb-o-informacnich-systemech-verejne-spravy>

[4] KOMÁRKOVÁ, Jitka. Úvod do informačních systémů: pro kombinovanou formu studia. Pardubice: Univerzita Pardubice, 2006. 85 s. ISBN 80-7194-870-5.

[5] *muni.cz* [online]. 2010 [cit. 2010-12-05]. Životní cyklus informačního systému. Dostupné z WWW: < http://www.fi.muni.cz/~smid/mis-zivcyk.htm >

[6] DRAHOTSKÝ, Ivo; ŘEZNÍČEK, Bohumil. Logistika – procesy a jejich řízení. Brono: Computer Press, 2003. 334 s. ISBN 80 - 7226 - 521 - 0.

[7] *slovnik-cizich-slov.cz* [online]. 2011 [cit. 2011-02-07]. Spedice.
Dostupné z WWW: < http://slovnik-cizich-slov.abz.cz/web.php/slovo/spedice>

[8] LIKŠŮ, Vladimír. Logistika 1. Praha: VŠE v Praze, 2001. 269 s.
ISBN 80 - 245 - 0166 - X.

[9] *raal.cz* [online]. 2011 [cit. 2011-02-07]. Spediční databanka RAALTRANS.
Dostupné z WWW: < http://www.raal.cz/cs/popis-raal>

[10] *jr-cz.eu* [online]. 2011 [cit. 2011-02-07]. ROZLÍVKA TRANSPORT s.r.o.
Dostupné z WWW: <http://www.jr-cz.eu/spedice.html>

[11] ŠIMONOVÁ, Stanislava; PANUŠ, Jan. Databázové systémy I. Pardubice: Univerzita Pardubice, 2005. 106 s. ISBN 978 - 80 - 7194 - 988 - 6.

[12] KEOGH, James. Java bez předchozích znalostí. Brno: CP Books, a.s., 2005. 280 s. ISBN 80-251-0839-2

[13] VIRIUS, Miroslav. Java pro zelenáče. Praha: Neocortex spol. s r.o., 2005. 268 s. ISBN 80-86330-17-6

[14] *mysql.com* [online]. 2011 [cit. 2011-02-05]. MySQL Connectors. Dostupné z WWW: < http://www.mysql.com/products/connector/>

[15] ŠEDA, Jan. *interval.cz* [online]. 2011 [cit. 2011-02-05]. Úvod do JDBC. Dostupné z WWW: < http://interval.cz/clanky/uvod-do-jdbc>

[16] *oracle.com* [online]. 2011 [cit. 2011-02-07]. Interface ResultSet. Dostupné z WWW: < http://download.oracle.com/javase/1.4.2/docs/api/java/sql/ResultSet.html>

[17] *sourceforge.net*[online]. 2011 [cit. 2011-02-08]. JChart2D. Dostupné z WWW: <*http://jchart2d.sourceforge.net/index.shtml*>

[18] *jasperforge.org* [online]. 2011 [cit. 2011-02-08]. JasperForge. Dostupné z WWW: < *http://jasperforge.org* >

[19] JEŽEK, Kamil. *java.cz* [online]. 2011 [cit. 2011-05-07]. Úvod do problematiky tisku sestav. Dostupné z WWW: <http://www.java.cz/detail.do?articleId=6795>

Seznam zkratek

ICT	Informační a komunikační technologie (Information and Communication Technologies)
IS	Informační systém (Information System)
ŽC	Životní cyklus
MMDIS	Multidimensional Management and Development of Information Systems
HW	Hardware
SW	Software
UML	Unifikovaný modelovací jazyk (Unified Modeling Language)
EPC	diagram procesu řízeného událostmi (Event-driven Process Chain)
PC	Osobní počítač (Personal Computer)
ERD	Entity Relationship Diagram
RMD	Relační model dat
NF	Normání forma
XML	Extensible Markup Language
SŘBD	Systém Řízené Báze Dat
JDBC	„Javovské“ databázové spojení (Java Database Connectivity)
LAN	Lokální síť (Local Area Network)
API	Aplikační programové rozhraní (Application Programming Interface)
JRE	Běhové prostředí Javy (Java Runtime Environment)

Příloha 1 – Původní podoba objednávky (část1)

ROZLÍVKA TRANSPORT s.r.o.

Semanínská 2094
56002 ČESKÁ TŘEBOV.
RAAL : H41

Tel: +420 465 530 222
Fax: +420 465 530 223

rozlivkatransport@volny.cz

p. Vyčítal
+420 775 202 697
pí Majksnerová
+420 775 202 696

SMLOUVA O PŘEPRAVĚ ZBOŽÍ.	**č. 142/2011**		
DOPRAVCE:	EN-KA Ústí n.O.	***TEL:***	
		FAX :	**4655254562**
SPZ:	Máca	***RAAL:***	***V60***
MÍSTO NAKLÁDKY:	JSS OBAL SERVIS Leština u Zábřeha		
DATUM A ČAS :	**22.4. 2011**		
NÁKLAD :	- kartony 2pal. 200kg		
MÍSTO VYKLÁDKY:	J.Škrkoň-Techplast, a.s. - provoz Benátky 56002 Česká Třebová		
DATUM A ČAS:	**22.4. 2011**		
POZNÁMKA:			

Příloha 1 – Původní podoba objednávky (část2)

SMLOUVA Č: **142/2011** ***FAKTURUJTE NA :*** **ROZLÍVKA TRANSPORT s.r.o.**
Semanínská 2094
56002 Česká Třebová
IČO:26009137 DIC:CZ 26009137

!!!!!!! POZOR ZMĚNA ADRESY !!!!!!

Cena : 1. 200,- Kč + DPH

SPLATNOST JE 60 DNÍ A JE PODMÍNĚNA OBRŽENÍM VEŠKERÝCH DOKLADU

a) **ORIGINÁL CMR POTVRZENÝ ODESÍLATELEM A PŘÍJEMCEM**

b) **U MEZINÁRODNÍ PŘEPRAVY RAZÍTKO CU V CMR,**

d) **POTVRZENOU NAŠI SMLOUVU - POŠLETE OBRATEM !!!**

e) **POTVRZENÉ DODACÍ LISTY !!!!!!!!!!!!!!**

!!! JINAK VÁM BUDE FAKTURA VRÁCENA ZPĚT !!!

Všeobecné podmínky: Pojištění nákladu je kryté Vámi v plném rozsahu. Není povolen přímý kontakt se zákazníkem. Přeprava se uskuteční dle § 610-629 OZ. V případě problémů Vás žádáme o okamžité upozornění. V případě nedodržení této smlouvy budou případné postihy přeneseny na Vás. Po ukončení přepravy zašlete fakturu nejdéle **do 7 dnů**. V ceně jsou zahrnuty veškeré náklady a prostoje spojené s touto přepravou. Při pozdějším zrušení č nepřistavení vozidla budou účtovány veškeré vzniklé více náklady až do výše přepravného.**V případě navrácení faktury pro její neúplnost, bude fakturovaná částka ponížena o 500,- Kč !!**.

FIRMA JE ZAPSANÁ V OBCHODNÍM REJSTŘÍKU, VEDENÉM KRAJSKÝM SOUDEM V HRADCI KRÁLOVÉ ODDIL C, VLOŽKA 19830

ROZLÍVKA TRANSPORT S.r.o.
Semanínská 2094 Česká Třebová
IČO: 26009137

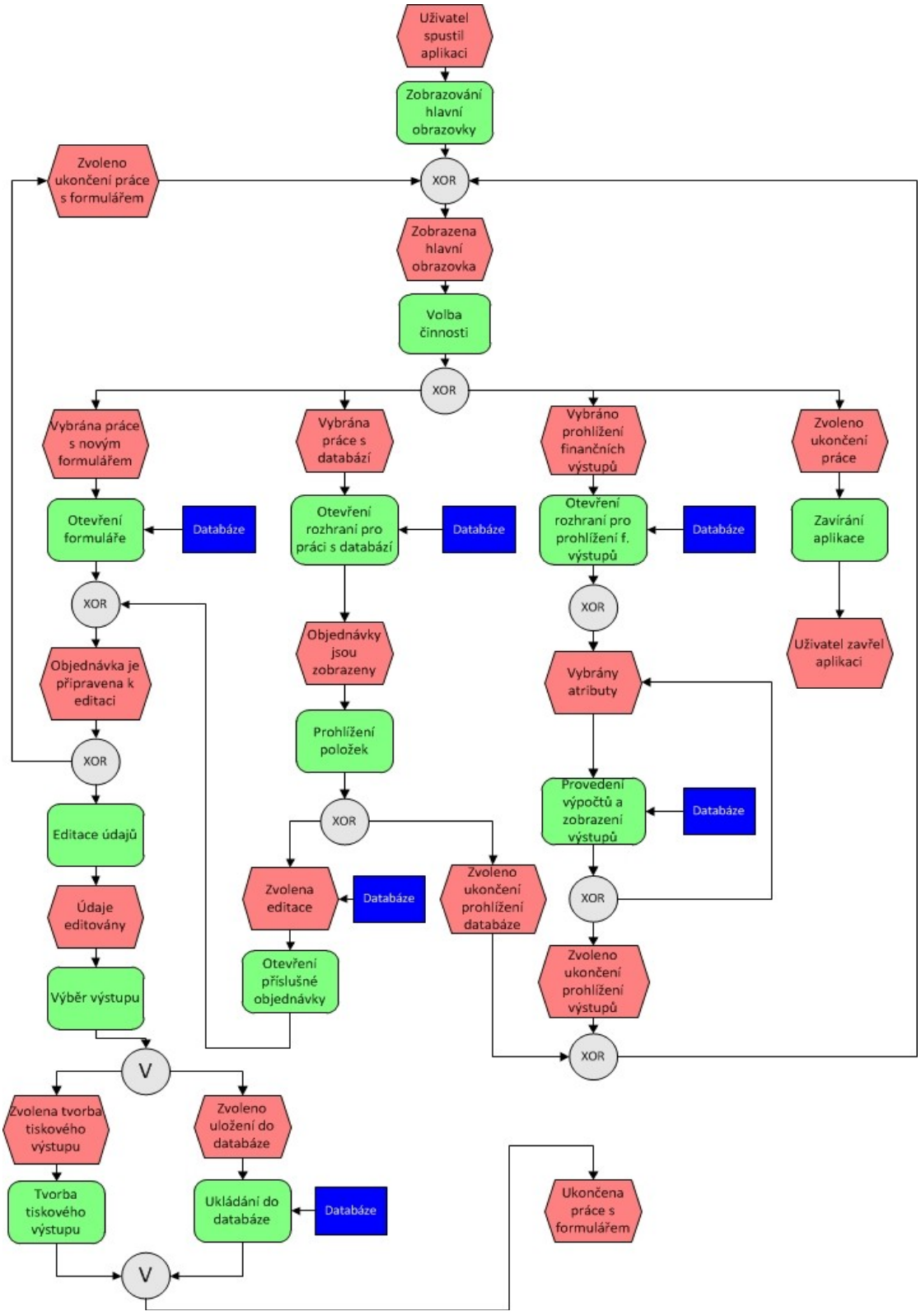
.....
OBJEDNAVATEL

.....
DOPRAVCE

Příloha 2 – Tabulka původního způsobu sepisování informací o objednávkách

DATUM	FIRMA/OBJ.	DOPRAVCE	CENA	ZŮSTATEK

Příloha 3 – EPC diagram práce s aplikací.



Příloha 4 – Nynější podoba objednávky

ROZLÍVKA TRANSPORT s.r.o

rozlivkatransport@volny.cz RAAL: H14

Semanínská 2094
56002 ČESKÁ TŘEBOVÁ

IČO: 26009137 Tel: +420 465 530 222 p. Vyčítal pí Majksnerová

DIČ: CZ 26009137 Fax: +420 465 530 223 +420 775 202 697 +420 775 202 696

Smlouva o přepravě zboží: **č. 2/2011**

Dopravce:	EPR PRAHA s.r.o. Praha 10, p. Štoudek		
TEL:	606617095	FAX:	461 533 116
SPZ:		RAAL:	2V1
Místo nakládky:	JSS OBAL SERVIS Leština u Zábřeha		

Náklad: - kartony 2pal. 200kg

Datum a čas: 22.4. 2011

Místo vykládky: J. Škrkoň-Techplast, a.s.
- provoz Benátky
56002 Česká Třebová

Datum a čas: 22.4. 2011

Cena: 1200 K + DPH

Poznámka:

SPLATNOST JE 60 DNÍ A JE PODMÍNĚNA OBDRŽENÍM VEŠKERÝCH DOKLADŮ

a) ORIGINÁL CMR POTVRZENÝ ODESILATELEM A PŘÍJEMCEM
b) POTVRZENOU NAŠI SMLOUVU - POŠLETE OBRATEM!!!
c) POTVRZENÉ DODACÍ LISTY!!!

!!! JINAK VÁM BUDE FAKTURA VRÁZENA ZPĚT !!!

Všeobecné podmínky: Pojištění nákladu je kryté Vámi v plném rozsahu. Není povolen přímý kontakt se zákazníkem. Přeprava se uskuteční dle § 610-629 OZ. V případě problémů Vás žádáme o okamžité upozornění. V případě nedodržení této smlouvy budou případné postihy přeneseny na Vás. Po ukončení přepravy zašlete faktutu nejdéle do 7 dnů. V ceně jsou zahrnuty veškeré náklady a prostoje spojené s touto přepravou. Při pozdějším zrušení či nepřistavení vozidla budou účtovány veškeré vzniklé více náklady až do výše přepravného. V případě navrácení faktury pro její neúplnost, bude fakturovaná částka ponížena o 500,- Kč !!

..... OBJEDNAVATEL

..... DOPRAVCE