UNIVERZITA PARDUBICE
ÚSTAV ELEKTROTECHNIKY A INFORMATIKY

SYSTÉM PRO ŘÍZENÍ SKLADU VE VÍCE LOKALITÁCH

BAKALÁŘSKÁ PRÁCE

AUTOR PRÁCE: Jan Podlešák
VEDOUCÍ PRÁCE: Ing. Lukáš Čegan

2007

UNIVERSITY OF PARDUBICE
INSTITUTE OF ELECTRICAL ENGINEERING AND INFORMATICS

SYSTEM FOR INVENTORY CONTROL IN MORE LOCALITIES

BACHELOR WORK

AUTHOR: Jan Podlešák
SUPERVISOR: Ing. Lukáš Čegan

2007

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 Univerzity Pardubice.

V Pardubicích dne 18. 5. 2007

Jan Podlešák

PODĚKOVÁNÍ

Za mnoho podnětných připomínek bych rád poděkoval panu ing. Lukáši Čeganovi, vedoucímu práce.

ABSTRAKT

Tato práce se zaměřuje na problematiku vývoje webových aplikací. Představuje a zhodnocuje současné technologické trendy. Dále prezentuje teorie řízení zásob, práce skladových systémů a technologie přepravy kusových zásilek. Porovnává také dostupné systémy řízení skladových zásob a spediční systémy. Součástí práce je vytvoření vlastní aplikace pro sledování a převod zboží mezi jednotlivými sklady. V aplikaci se klade důraz na propojení správy vozového parku se správou skladů.

OBSAH

1 Úvod

Cílem této bakalářské práce je vytvořit webovou aplikaci pro řízení skladů ve více lokalitách. Toto téma jsem si vybral, protože mne zajímá tvorba webových stránek a práce s databázemi. A také proto, že na trhu neexistuje žádný podobný systém, který by zahrnoval správu skladů a vozového parku, a byl běžně dostupný.

Aplikace umožní vytvoření soustavy skladů podle technologie přepravy kusových zásilek. Především se bude zaměřovat na propojení, typ skladů a způsob dopravy mezi nimi. Dále bude umět založit a spravovat vozový park. V rámci programování aplikace budu implementovat algoritmus na optimalizovaný přesun zboží mezi sklady.

V teoretické části práce dojde k seznámení s technologiemi používající se při vývoji webových aplikací. Bude také nastíněna problematika teorie řízení zásob a technologie přepravy kusových zásilek. V práci budou představeny a zhodnoceny některé ze spedičních systémů.

V implementační části bude navrhnuta a realizována výše zmíněná aplikace.

2 Pojem webové aplikace

Jako výsledný produkt této práce jsem vybral webovou aplikaci, protože ta nabízí efektivnější a levnější řešení zadání než klasická aplikace. Pro vývoj webové aplikace stačí teoreticky pouze textový editor a klientská konzolová aplikace, která je dodávána spolu s databázovým serverem. Webová aplikace je poskytovaná uživatelům z webového serveru a jako klient se využívá webový prohlížeč. Proto se hotový program nemusí, oproti běžné aplikaci, instalovat na každý počítač, na kterém chceme program používat. Podstatnou výhodou je tedy schopnost pracovat podle určení bez ohledu na operační systém či jeho verzi instalovanou na daném klientském počítači. Nevýhodou webové aplikace je, že musíme být připojeni k síti. V tomto případě to ale nevadí, protože v systému řízení skladů ve více lokalitách se počítá s komunikací pomocí sítě. Nyní bych chtěl blíže specifikovat pojem webové aplikace.

2.1 Definice

Termín webová aplikace je dnes používán velmi frekventovaně, definic lze najít mnoho. Např. v dokumentaci k Microsoft Windows 2000 Serveru je uvedeno, že „webová aplikace je softwarový program, který jako svůj základní komunikační protokol používá HTTP a uživateli doručuje data v jazyce HTML.“ Wikipedie říká, že „Webová aplikace je aplikace doručená uživateli z webového serveru pomocí služby WWW.“ Aplikace je přitom definována jako program nebo skupina programů, které mají koncovému uživateli poskytovat nějakou funkcionalitu. Firma Sun zase pro svou platformu J2EE používá následující definici: „Webová aplikace je aplikace napsaná pro Internet, ať už pomocí javovských technologií, jako jsou např. JavaServer Pages a servlety, nebo jako aplikace vytvořené nejavovskými technologiemi, jako jsou např. CGI nebo Perl.“ [1]

2.2 Vlastní vymezení

Jelikož se tato práce zabývá tvorbou webové aplikace, je potřeba si tento termín definovat. Webová aplikace zde bude chápána jako aplikace, která

- je postavena na modelu klient/server
- pro komunikaci mezi klientem a serverem používá protokol http
- rozhraním pro koncového uživatele je webový prohlížeč
- požadavky klienta zpracovává webový server
- zachovává třívrstvou strukturu – datová (databáze), logická (aplikační a webový server), prezentační (webový prohlížeč)

2.3 Priority při programování webových aplikací

V praxi existuje několik zásad, pro programování webových aplikací. V první řadě je to bezpečnost. Vždy totiž hrozí riziko ztráty či zničení dat. Velké nebezpečí je v podobě krádeže informací nebo nabourání webového serveru skrze aplikaci.

Další vlastností je rozšiřitelnost a přehlednost zdrojového kódu. Jakmile se webová aplikace osvědčí v praxi, začíná se obvykle pracovat na jejích dalších úpravách, vylepšeních a nadstavbách. Pokud bylo s těmito modifikacemi počítáno už při návrhu aplikace, je jejich zapracování mnohem jednodušší. Z tohoto důvodu je dobré řešit většinu složitějších webových aplikací modulárním způsobem.

Důležitá je také rychlost. Pomalá webová aplikace je jen velmi málo použitelná, problémy s ní mají i vyhledávače. Proto je třeba optimalizovat všechny skripty na rychlost.

Všechny komponenty a služby hotové aplikace musí být funkční a pracovat správně tak jak mají. Výsledný design by měl být pokud možno jednoduchý a přehledný [2].

3 Prostředky vývoje webových aplikací

Prostředků pro vytvoření webové aplikace je několik. Základním je značkovací jazyk (např. HTML, XHTML,…). Tyto však umožňují tvorbu pouze statických stránek bez interaktivních prvků a bez přístupu do databáze. Spolupráci s databází a interaktivní prvky nabízejí skriptovací jazyky. V následujících kapitolách stručně charakterizuji některé z nich. Důležitým krokem při vytváření aplikace je také výběr vhodné databázové platformy, se kterou budeme prostřednictvím skriptů komunikovat [3].

Obr. č. 1: Schéma webové aplikace[1]

3.1 Jazyky pro vývoj webových aplikací

3.1.1 PHP

PHP je skriptovací jazyk, skripty jsou prováděny na straně serveru, tzn. že k uživateli je přenášen až výsledek jejich činnosti. Syntaxe jazyka kombinuje hned několik programovacích jazyků (Perl, C, Pascal a Java). PHP je nezávislý na platformě, skripty fungují bez úprav na mnoha různých operačních systémech. Obsahuje rozsáhlé knihovny funkcí pro zpracování textu, grafiky, práci se soubory, přístup k většině databázových serverů (mj. MySQL, ODBC, Oracle, PostgreSQL), podporu celé řady internetových protokolů (HTTP, SMTP, SNMP, FTP, IMAP, POP3, LDAP, …) PHP je rekurzivní zkratkou pro PHP: Hypertext Preprocessor (kde původní význam zkratky je Personal Home Page) [4].

[1] Zdroj: http://www.adaptic.cz/novy-web/programovani-webovych-aplikaci.htm

3.1.1.1 Historie

Základy PHP položil Rasmus Lerdorf v roce 1995, kdy pro své potřeby sepsal několik Perlovských skriptů, které nazval Personal Home Page Set. Jak rostla potřeba pro nové a nové funkce, PHP/FI (jak se technologie až do verze 2.0 jmenovala) se rozrůstalo a bylo přepsáno do jazyka C.

PHP velmi podobné tomu, které známe dnes, vzniklo ke konci roku 1997 jako kompletní přepsání PHP/FI 2.0 pány Andi Gutmansem a Zeevem Suraskim. Rasmus Lerdorf se k této iniciativě přidal a v polovině roku 1998 vzniklo PHP 3.0. Hlavními přínosy verze 3 byla dobrá infrastruktura pro mnoho databází a různých API, rozšiřitelnost, výkonnější a konzistentnější syntaxe s podporou OOP.

PHP 3.0 bylo už poměrně dobrým nástrojem pro mnoho typových situací, nebylo ale navrženo tak, aby komplexní situace zvládalo efektivně. To bylo hlavní motivací pro vývoj PHP 4.0, jehož základem bylo nové jádro napsané Andi Gutmansem a Zeevem Suraskim. Toto jádro nese název, který byl inspirován jejich křestními jmény – Zend Engine. Kromě zvýšené výkonnosti přineslo PHP 4 několik dalších podstatných změn, jako např. HTTP sessions, podporu pro mnoho web serverů apod.

Nejnovějším přírůstkem je PHP 5.0. Opět přineslo několik vylepšení, ale jedno je mimořádného významu: PHP 5 výrazně zlepšuje objektově orientovanou syntaxi. Přibyly výjimky, konstruktory, destruktory a mnoho dalších jazykových rozšíření. Mezi další podstatná rozšíření patří např. přepracovaná podpora XML, podpora COMu a technologie .NET, lepší podpora MySQL a další [1].

3.1.2 ASP

ASP (Active Server Pages) je technologie vyvinutá společností Microsoft (první verze byla vydána v roce 1996). Je nezávislá na programovacím jazyce a umožňuje vykonávání kódu na straně serveru a následné odeslání výsledku uživateli. To znamená, že webová stránka s

příponou .asp obsahuje kód, který se vykoná na IIS serveru a prohlížeči odešle pouze výsledek, ve značkovacím jazyce HTML, který umí bez problému zobrazit. Programovací jazyky, které se pro ASP používají jsou VBScript a JScript [5].

3.1.3 JSP

JSP (Java Server Pages) je nástroj, pro psaní dynamických HTML stránek, založený na jazyce Java. Funkčností je velmi podobný ASP nebo PHP. Jedná se vlastně o HTML stránky, do kterých je pomocí speciálních značek vložen kód v Javě, který se provádí při vyřizování dotazu na straně serveru.

Na rozdíl od PHP a ASP, které nemají téměř žádnou typovou kontrolu, má JSP typovou kontrolu z Javy. Další odlišností je převádění stránek do tzv. servletů, což jsou speciální třídy v jazyce Java, které pak komunikují s web serverem. To umožňuje rychlý vývoj aplikací a částečně také oddělení designu od výkonného kódu. Díky servletům jsou také JSP nezávislé na platformě. Vytvořené řešení pak můžeme přenést na jiný web server, který má nainstalovanou podporu JSP [6].

3.1.4 ASP.NET

ASP.NET je nadstavba .NET Frameworku firmy Microsoft pro tvorbu webových aplikací a služeb. Je nástupcem technologie ASP a přímým konkurentem JSP (Java Server Pages). V rámci ASP.NET se standardně pracuje s programovacími jazyky podporující CLR (Common Language Runtime), např. Visual Basic.NET, C#.NET, JScript.NET, ale i mutace Perlu nebo Pythonu a další. Prostředí CLR je sdíleno všemi aplikacemi postavenými na .NET Frameworku. Aplikace založené na ASP.NET jsou také rychlejší, neboť jsou předkompilovány do jednoho či několika málo DLL souborů, na rozdíl od ryze skriptovacích jazyků, kde jsou stránky při každém přístupu znovu a znovu parsovány [7].

Tři základní stavební kameny ASP.NET jsou Web Forms, Web Services a Mobile Internet Toolkit. Web Forms slouží k vytváření webových stránek pomocí komponent podobně, jako je tomu zvykem u

„okénkových“ aplikací. Důležitou součástí je rovněž koncept serverových ovládacích prvků. To, že jsou do ASP.NET integrovány webové služby, v podobě Web Services, je velkým plusem. Celý .NET je inspirován snahou usnadnit vývoj distribuovaných systémů, a webové služby jsou jasnou cestou, kudy se pravděpodobně integrace heterogenních systémů bude ubírat. Mobile Internet Toolkit je sada nástrojů pro generování výstupu pro mobilní zařízení [1].

3.2 Databázové systémy

Každá aplikace pracuje s určitým objemem dat, přičemž některá z nich jsou získávána přímo od uživatelů. Tato data je nutno zpracovat a většinou také ve vhodné formě uchovávat a archivovat. K tomuto účelu slouží databázové systémy. Pro svou práci jsem vybral MYSQL.

3.2.1 MYSQL

MYSQL je zkratka z angl. My Structured Query Language. Je to databázový systém, od švédské firmy MYSQL AB. Do MYSQL lze ukládat různá data (texty, obrázky atd.), s nimiž lze dále jednoduše pracovat (třídit, řadit, filtrovat apod.). Databáze MYSQL je jeden z prvních hojně rozšířených systémů. Práce s tímto systémem se dá využít v C, C++, Java, Perl, PHP, Python, Tcl, Visual Basic nebo .NET.

Pro jednoduchou správu MYSQL databází se používá nástroj PhpMyAdmin. PhpMyAdmin je Open Source program napsaný v PHP, který umožňuje zálohování, vytváření tabulek, vkládání, editaci a mazání záznamů v tabulkách, vytváření databází apod. PhpMyAdmin je pokročilý nástroj pro kompletní správu MYSQL systému přes webové rozhraní [8].

4 Teorie řízení zásob, práce skladových systémů a technologie přepravy kusových zásilek

Na logistických řetězcích působí vedle informačních toků a dopravy další subsystémy. Jsou to činnosti zabezpečující manipulaci s materiály, komponenty pro výrobu a zbožím, udržování a řízení zásob, provozování skladů a balení. Z hlediska intenzifikace činností na logistickém řetězci má největší význam udržování a řízení zásob.

4.1 Teorie řízení zásob

Cílem řízení zásob je jejich udržování na takové výši a v takové struktuře, aby byla zabezpečena rytmická a nepřerušovaná výroba a pohotovost a úplnost dodávek tak, aby náklady s tím spojené byly minimální.

Řízení stavu zásob a skladového hospodářství je jedním ze základních principů, o které se zajímá logistika. Z hlediska teorie je vypracována metodologie, která se obecně nazývá teorie zásob a je klasifikována jako jedna ze speciálních metod operačního výzkumu.

Pro aplikaci teorie zásob v řízení skladového hospodářství mají význam modely zásob, které jsou dvojího charakteru:

- **deterministické**, tj. takové, které předpokládají dodávky dopředu v určeném čase a velikosti, přičemž se optimalizuje velikost zásoby.
- **stochastické**, u kterých periodu mezi dodávkami, odběrem a velikostí dodávek určujeme pravděpodobnostním modelem a optimalizujeme opět velikost zásob [9].

4.2 Teorie řízení práce skladových systémů

Historicky bylo zboží dodáváno na logistickém řetězci vždy přes sklady. Podle toho, zda se ze skladu odebírají suroviny, materiály nebo montážní komponenty nebo zda se hotové produkty distribuují (expedují), rozeznáváme:

- **sklady předvýrobní**, pro uskladňování surovin, materiálů a komponent pro následnou fázi výroby

- **sklady distribuční**, či expediční pro skladování a distribuci (expedice) hotové produkce pro další fázi výroby, obchod a spotřebu.

4.2.1 Sklad

Sklady jako technická zařízení představují budovy na předem stanovené ploše pro ukládání zásob. V počátcích budování skladů hrály velkou roli pro jejich alokaci dopravní náklady. Bylo zásadou budovat sklady co nejblíže k odběratelům, což umožňovalo rychlý rozvoz zboží na krátké vzdálenosti a s poměrně malými náklady na dopravu

S postupujícím vývojem manipulační techniky, mechanizace úložných prací a automatizace se zjišťovalo, že vysokou techniku je možné s dobrým ekonomickým efektem pořizovat pouze za předpokladu jejího využití určitým výkonem. Takových výkonů se však nedosahovalo u malých, plošně rozptýlených skladů. Bylo účelné budovat větší sklady s rozsáhlejšími atrakčními obvody, i za cenu vyšších nákladů na dopravu. Ekonomická efektivnost takto budovaných koncentrovaných skladů se stanoví tak, že zvýšené náklady na dopravu nesmějí být větší, než celkové úspory vzniklé nasazením vyššího stupně mechanizace. Do těchto úspor je nutné zahrnout i úspory vzniklé tím, že větší sklady umožňují nasazení lepší řídící techniky a informačních technologií, včetně využití optimalizačních metod [9].

4.2.2 Řízení skladů

Moderní skladové hospodářství vyžaduje vyspělý management. Ten působí ve třech základních rovinách:

- strategické řízení
- taktické řízení
- operativní řízení

4.2.2.1 Strategické řízení skladových procesů

Základním strategickým rozhodnutím v oblasti řízení skladových procesů je rozhodnutí o způsobu zásobování atrakční oblasti výroby a distribuce. Zda je účelnější zásobování z plošně rozptýlených skladů nebo ze skladu centrálního. Dále také, zda je vhodná výstavba a provoz vlastních skladových systémů, nebo jestli je vhodnější předat tyto činnosti outsourcingové firmě.

4.2.2.2 Taktické řízení skladových procesů

V taktické fázi se již nerozhoduje o alokaci skladů, může se však měnit plán řízení skladu v souladu s prognózou výroby a koncepcí řízení zásob. Je nutné optimalizovat rozmístění úložných míst jednotlivých položek podle jednotlivých kriterií, jimiž jsou zejména: druh ukládaného zboží, způsob uskladnění a vyskladnění, druh obalové techniky, logistická technologie a další.

4.2.2.3 Operativní řízení skladových procesů

Operativní organizace práce musí plnit zejména dva úkoly: uskladňování a vyskladňování musí probíhat ve stanovených termínech bez poruch a s minimálními náklady a evidence ve skladech musí umožňovat kontrolu stavu zásob podle množství a hodnoty. Systém skladování a řízení skladovacích procesů musí obsahovat:

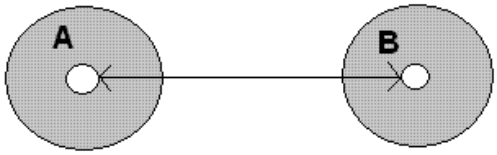
- optimalizaci posloupnosti uskladňovacích a vyskladňovacích procesů
- přiřazení nositelů uskladňování k volným uskladňovacím místům

- přiřazení nositelů vyskladnění k plným skladovacím místům
- bezporuchová a plynulá identifikace uskladňovacích a vyskladňovacích operací
- kontrola stavu uskladněných položek
- aktualizace stavu uskladněných položek [9]

4.3 Technologie přepravy kusových zásilek

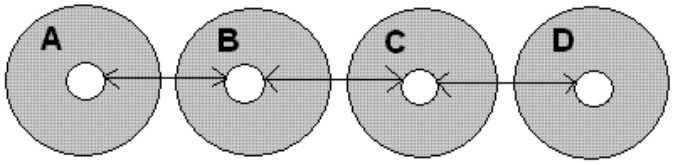
Systém přepravy kusových zásilek je založen na sdružování jednotlivých zásilek, jejich společné přepravě a rozvozu příjemcům v této oblasti. Může být organizován různým způsobem. Základními druhy organizace a technologického provedení systému přepravy kusových zásilek jsou:

a) Přeprava na jedné sběrné lince mezi dvěma místy soustřeďování a rozptylu zásilek (překladiště). Jde o organizačně nejjednodušší způsob.



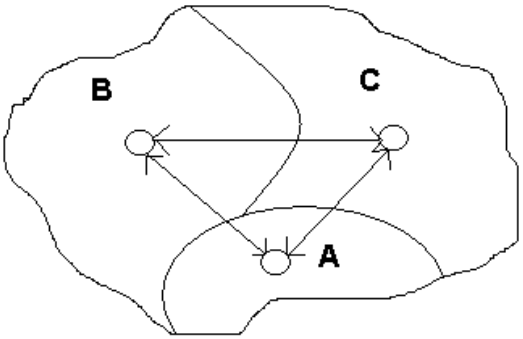
Obr. č. 2: Přeprava kusových zásilek na jedné lince mezi dvěma místy

b) Přeprava na určité sběrné lince s několika místy soustřeďování a rozptylu zásilek



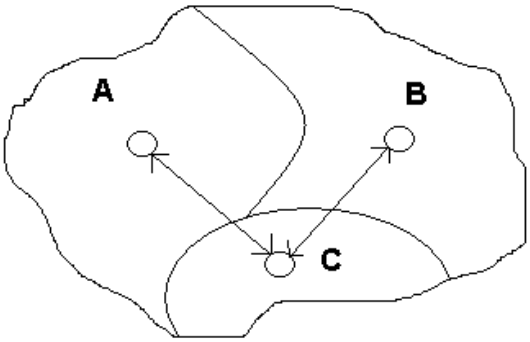
Obr. č. 3: Přeprava kusových zásilek na jedné lince mezi několika místy

c) Přeprava v systému sběrných linek pokrývající určité území podle zásady, že každé sběrné středisko má se všemi ostatními středisky samostatnou sběrnou linku.



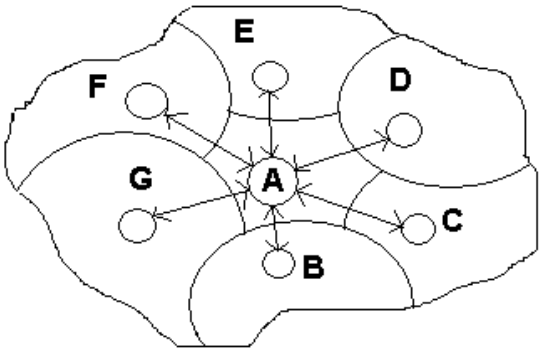
Obr. č. 4: Přeprava kusových zásilek systémem přímých linek

d) Systém obdobný schématu c), ve kterém však jsou spojovací sběrné linky mezi sběrnými středisky s malým vzájemným obratem zásilek vedeny přes vhodná tranzitní překladiště.



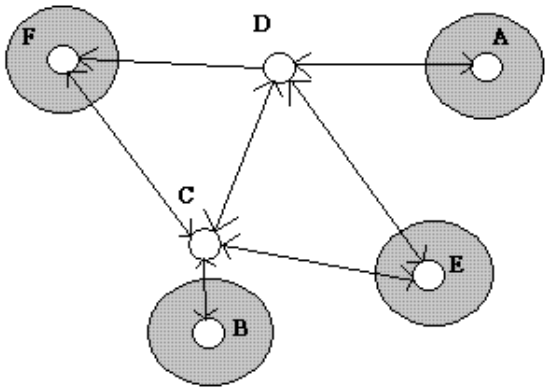
Obr. č. 5: Přeprava kusových zásilek s použitím tranzitního překladiště (C)

e) Systém centrálního sběrného místa, ve kterém jsou všechna sběrná střediska propojena sběrnými linkami výhradně s centrálním sběrným místem. Tam přepravují všechny zásilky směřující mimo jejich sběrný obvod a v opačném směru putují zásilky ze všech ostatních středisek směřující k nim.



Obr. č. 6: Přeprava kusových zásilek přes centrální překladiště (A)

f) Systém podle jakéhokoliv předchozího schématu, ve kterém je využit jiný druh dopravy, např. železniční, letecké nebo vodní.



Obr. č. 7: Kombinovaná přeprava kusových zásilek [10]

5 Srovnání ostatních programů

Aplikací zabývající se správou, evidencí a vedením skladů je velké množství. Všechny, které jsem nalezl, jsou však pouze klasické programy pracující až na výjimky jen pod operačním systémem MS Windows. Například skladová evidence TRELL[1] od firmy TRELLSOFT nabízí správu neomezeného počtu skladů, správu příjmů, výdeje a převodů, přepočítávání cen a další pokročilé funkce. Přes síť může být propojeno až deset nezávislých poboček.

Dále mne zaujal program Skladík[2], který obsahuje skladovou evidenci materiálu a zásob menšího rozsahu pro malé a střední firmy. Umožňuje provádět příjem, výdej, tisk dokladů o pohybu, přehled o pohybech v daném období, inventur a různých přehledů, za období, zaúčtování do účetnictví. Je zde možnost práce s více sklady a práce v síti. Připojen je i modul fakturace pro fakturování výdeje i tvorbu nezávislých faktur.

Všechny další programy (jako např. Warehouse EU[3], Tellus[4] a Sklady[5]) pracují velmi podobně a nabízejí podobné funkce, jedna zásadní však všem chybí. Je to možnost, spolu se skladem, evidovat a spravovat vozidla. Tento nedostatek lze vyřešit instalací dalšího specializovaného programu, kterým je např. Dispatcher[6] nebo Plantour[7], zabývající se logistikou dopravy, spedicí, evidencí a administrativní správou silničních vozidel a strojů. Toto řešení ale není tak efektivní jako možnost pracovat v jednom prostředí, které nabízí správu skladů i vozového parku.

[1] http://trell.hyperlink.cz/skladyg.htm
[2] http://www.mbsw.cz/produkty/skladik/index.htm
[3] http://www.greg.cz/
[4] http://www.grebestudios.com
[5] http://www.ph-software.com/
[6] http://www.dispatcher.cz/
[7] http://www.plantour.cz

6 Specifikace cílů, možností a problémů

Aplikace umožní vytvoření soustavy skladů, které budou umístěné do struktury s jedním centrálním překladištěm (viz. kapitola 4.3 - technologie přepravy kusových zásilek). Jednotlivé sklady ponesou informace o počtu zboží, vozidlech a skladnících. Všechny údaje bude možno přidávat, mazat a různě upravovat.

Při objednávání zboží si klient bude moci vybrat výdejní prodejnu. Pokud dané zboží nebude na skladě, systém ho objedná z nejbližšího skladu a zadá požadavek na jeho přesun. Na konci dne potom vytvoří rozpis jízd pro jednotlivá vozidla. Cílem bude co nejefektivněji využívat přepravní možnosti vozidel.

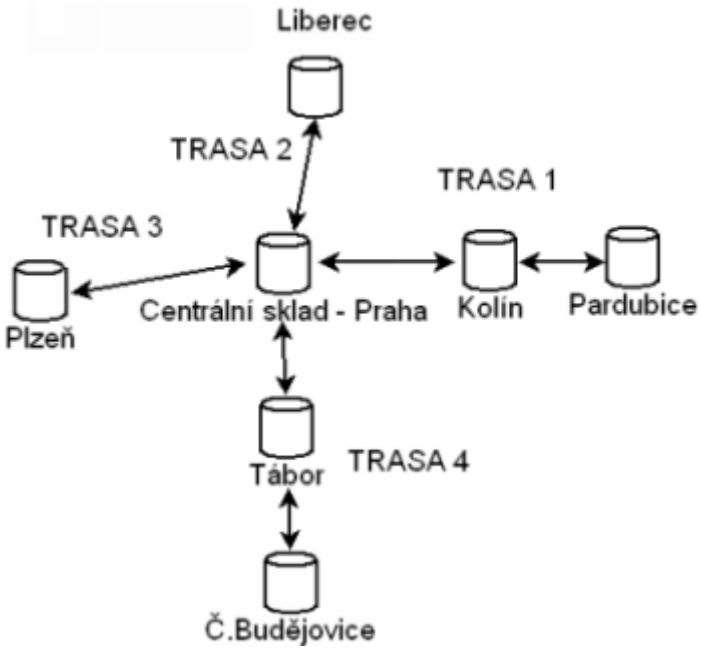
Aplikace bude mít klientskou, skladnickou a administrátorskou část. Klientům bude umožňovat objednávání zboží a sledování stavu objednávky. Skladníkům editaci údajů a kontrolu nakládky, vykládky a výdeje zboží. Administrátor bude mít vládu nad celým systémem, tzn. vytváření skladů a jejich umisťování do struktury, spravování zboží, vozidel, skladníků a klientů.

6.1 Prvotní návrh

Možností jak může vypadat struktura skladů je více, já jsem vybral systém s jedním centrálním překladištěm. Sklady budou sdružovány do tras a jednotlivé trasy budou napojeny na centrální překladiště.

6.1.1 Problém tras

Při vytváření tras je nejprve nutno zvolit, který sklad bude plnit funkci centrálního střediska. Další pobočky se potom zařadí do tras podle uživatelových potřeb. Když se vytvoří nový sklad, přidá se k nějaké již vytvořené trase nebo se pro něj vytvoří zcela nová trasa. Na dalším obrázku je vidět, jak může taková struktura skladů v trasách vypadat.



Obr. č. 8: Příklad schématu tras

6.1.2 Problém vozidel

Vyvstává problém jak budou řešeny jízdy vozidel po jednotlivých trasách. Auta se budou pohybovat pouze z domovského skladu do centrálního a zpět. Cestou budou nakládat a vykládat zboží, jak bude potřeba. Jakmile vozidlo přijede do centrálního skladu, vyloží vše co veze a naloží zboží, které směřuje z jiných tras do jeho trasy. Nakládá však pouze to zboží, které necestuje dál než do domovského skladu auta. Vycházíme-li z předchozího obrázku, znamená to, že vozidlo z Kolína nikdy nepoveze zboží směřující do Pardubic. Z toho plyne podmínka pro řádný přesun zboží – v každé trase musí být alespoň jedno vozidlo a to v nejvzdálenějším skladu od centrálního překladiště.

6.1.3 Algoritmus jízd

Algoritmus, který bude vypočítávat přesun zboží musí řešit několik problémů.

Jeden z nich je vyřešit, za jakých podmínek má auto ze skladu vyjíždět. Jestli pouze když něco poveze z domovské pobočky a nebo jestli i když nic neveze přímo ze svého skladu, ale cestou do a z centrálu by naložilo zboží, které by se jinak, např. z kapacitních důvodů, nemohlo převést.

Dále se musí zhodnotit, zda přesun zboží nelze rozdělit mezi ostatní vozidla na trase, které už mají výjezd naplánován. Tím by se ušetřilo jedno vozidlo, avšak nesmí to být za cenu toho, že by se nějaké zboží nedoručilo.

6.1.4 Problém hmotnosti a objemu zboží

Algoritmus musí také řešit omezenou kapacitu vozidel. Každé auto má svou nosnost a objem přepravního prostoru. Dále musí počítat s tím, že při nakládce vzniká mezi zbožím volný prostor. Objem vozidla, se kterým se bude počítat, je tedy nutno snížit v průměru o 30 %.

Když se všechno požadované zboží do vozidla nevejde, přenechá se zbytek dalším vozidlům. Není-li již žádné auto k dispozici, musí algoritmus přesunout zbývající zboží na další den.

Tímto se zajistí co nejrychlejší přeprava i velkého počtu zboží s málo auty. Požadavek je aby od objednávky k doručení zboží uplynulo co nejméně času. To samozřejmě záleží na kapacitách a počtu vozidel. V praxi by to nemělo trvat déle než 2-3 dny.

6.2 Další možnosti návrhu

Nyní bych rád zmínil některé další eventuality návrhu, které ale z časových možností a k rozsahu zadání práce nebudu implementovat.

6.2.1 Clark – Wrightův algoritmus

Technologií přepravy zboží mezi sklady je několik. Vědní obor, který se touto problematikou zabývá, se nazývá VRP (Vehicle Routing Problem)[1]. Jeden z prvních a nejznámějších heuristických algoritmů, který byl vyvinut pro řešení VRP, je Clark – Wrightův algoritmus z roku 1964.

Jak tento algoritmus funguje. Předpokladem je, že vozidla jsou uspořádané sestupně vzhledem k jejich kapacitě. Algoritmus úspor je pak následující:

[1] více na http://neo.lcc.uma.es/radi-aeb/WebVRP/main.html

1. Pro každou dvojici zákazníků Z_i a Z_j vypočteme úsporu $s_{ij} = c_{i0} + c_{0j} - c_{ij}$. Tedy kolik ušetříme, když nahradíme dvě trasy $(Z_0;Z_i;Z_0)$ a $(Z_0;Z_j;Z_0)$ jednou trasou $(Z_0;Z_i;Z_j;Z_0)$.

2. Úspory setřídíme sestupně a začínáme na začátku seznamu. Postupujeme následovně.

Paralelní verze

3. Pro úsporu s_{ij} procházíme vozidly $V_1, ..., V_m$. Pro vozidlo V_k a zákazníky Z_i,Z_j, kteří přísluší úspoře s_{ij} , může nastat jedna z následujících možností:

4. Z_i a Z_j nejsou v cestě žádného z vozidel a vozidlo V_k nemá ve své trase žádného zákazníka. Když $i \neq j$ pak do trasy V_k vložíme $Z_0\ Z_i\ Z_j\ Z_0$, když $i=j$ tak vložíme $Z_0\ Z_i\ Z_0$

5. Z_i je v trase vozidla V_k a Z_j není v trase žádného vozidla a když trasa vozidla V_k je ve tvaru $(Z_0,Zi,...,Z_0)$ nebo $(Z_0,..., Z_i, Z_0)$. Potom když náklad trasy vozidla V_k spolu s požadavkem q_{ij} nepřekročí kapacitu vozidla V_k, tak Z_j přidáme do trasy mezi Z_i a Z_0. Trasa vozidla V_k potom bude ve tvaru:

   buď $(Z_0;Z_j;Z_i;...;Z_0)$ nebo $(Z_0;...;Z_i;Z_j;Z_0)$

6. Jestliže nenastala žádná z přecházejících možností, přejdeme na vozidlo V_{k+1} [11]

7. Přejdeme na další úsporu a opakujeme krok 3 dokud neprojdeme celý seznam úspor.

Sekvenční verze

3. Pro vozidlo V_k procházíme seznam úspor od začátku. Pro vozidlo V_k a zákazníky Z_i,Z_j, kteří přísluší úspoře s_{ij} , může nastat jedna z následujících možností:

4. Z_i a Z_j nejsou v cestě žádného z vozidel a vozidlo V_k nemá ve své trase žádného zákazníka. Když $i \neq j$ pak do trasy V_k vložíme Z_0 Z_i Z_j Z_0, když $i=j$ tak vložíme Z_0 Z_i Z_0

5. Z_i je v trase vozidla V_k a Z_j není v trase žádného vozidla a když trasa vozidla V_k je ve tvaru (Z_0,Zi,...,Z_0) nebo (Z_0,..., Z_i, Z_0). Potom když náklad trasy vozidla V_k spolu s požadavkem q_{ij} nepřekročí kapacitu vozidla V_k, tak Z_j přidáme do trasy mezi Z_i a Z_0. Trasa vozidla V_k potom bude ve tvaru:

   buď (Z_0;Z_j ;Z_i; ...;Z_0) nebo (Z_0; ...;Z_i;Z_j ;Z_0)

6. Jestliže nenastala žádná z předcházejících možností, tak přejdeme na úsporu, která následuje v seznamu.

7. Krok 3 opakujeme pro každé vozidlo.

Přepokládám, že by se tento algoritmus mohl později naprogramovat a připojit jako modul k této práci.

6.2.2 E-shop

Cílem vytvářené aplikace je, aby se stala podpůrným systémem pro řízení a chod e-shopu. E-shop by měl poskytovat aplikaci hlavně informace o objednávkách. Naproti tomu aplikace by měla vracet údaje o stavu objednávek a stavu zboží na skladech.

Propojení obou odlišných systému by bylo možné pomocí tzv. webservices. Webservices je technologie pro vzdálené volání procedur pomocí přenosu zpráv v jazyku XML protokolem HTTP. Pro definici typů dat využívá standard XML Schema. Pro identifikaci objektů (dat, jmen operací, atd.) využívá standard XML Namespaces. Technologii webových služeb tvoří tři části:

- protokol pro vzdálené volání procedur, zvaný SOAP
- jazyk pro popis poskytovaných služeb, zvaný WSDL
- mechanismy pro nalezení služeb, zvané UDDI a WSIL [12]

Vzhledem k náročnosti tohoto úkolu, budu pro testování používat pouze klientskou část, jejíž výstupy budou suplovat výstupy z e-shopu.

7 Definování informačních potřeb

Motivací k vytvoření práce je neexistence podobného univerzálního řešení na trhu. Tato aplikace má široké spektrum použití např. v zasilatelství, ve firmách, kde se pravidelně rozváží zboží, v zásobování, svoz materiálu ve výrobě a další.

Aplikace je také určená pro napojení na e-shop, kde poskytuje silnou podporu pro chod a spravování firemních poboček a vozidel. Je vhodná zvláště pro středně velké firmy s několika prodejnami. Těmto společnostem nabízí vyřešení problému s optimalizací přesunu zboží mezi jednotlivými pobočkami a s tím spojenou minimalizaci nákladů na přepravu.

Protože je to webová aplikace, běžící na jednom serveru, umožňuje spravovat celý systém najednou na dálku bez nutnosti fyzicky objíždět všechny sklady. To je dobré např. při údržbě nebo upgradu aplikace. Tím šetří firmám finanční i lidské prostředky.

Rozšířením aplikace o další funkce (jako např. posílání zpráv mezi uživateli nebo sdílení souborů) lze tuto také zavést jako firemní intranet.

Výhody použití počítačové aplikace pro řízení skladu a přepravy kusových zásilek jsou především:

- zlepšení řízení
- poskytnutí včasných a přesných informací
- zvýšení spolehlivosti
- zlepšení finančních výsledků podniku

8 Analýza systémových potřeb

V této kapitole bych rád představil technologie a nástroje, které budu používat při vývoji webové aplikace na správu skladů. Dále pak analyzuji nároky na její provoz.

8.1 Technologie použité pro tvorbu aplikace

Program bude napsaný ve značkovacím jazyce HTML. Jako skriptovací jazyk použiji PHP, který bude využívat databázi MYSQL. Řešení PHP + MYSQL jsem si vybral, protože je zadarmo, je nezávislé na platformě, má velkou podporu a je obecně velice rozšířené. Podrobně jsou vlastnosti PHP, MYSQL a dalších prostředků pro vývoj webových aplikací popsány v kapitole 3.

Na grafickou úpravu stránek systému využiji prostředků kaskádových stylů CSS (Cascading Style Sheets), které umožňují efektivně sjednotit vzhled všech částí.

8.2 Nástroje použité pro tvorbu aplikace

Zdrojové kódy aplikace budu psát v programu PSPad 4.3[1]. Tento program je freewarový editor textů, který zvýrazňuje syntaxi programovacích jazyků. Poskytuje také různé užitečné nástroje pro tvorbu internetových stránek, jako je např. kontrola syntaxe, pokročilá editace textu (vyhledávání, kontrola pravopisu) a šablony pro PHP, SQL, HTML a další jazyky.

K testování použiji program VertrigoServ 2.10[2]. VertrigoServ kombinuje Apache HTTP server, MySQL server, PHP hypertext preprocesor a PhpMyAdmin do jednoho balíčku. Díky tomu mohu jednoduše spouštět testovanou aplikaci na lokálním počítači bez přístupu na internet.

[1] `http://www.pspad.com`
[2] `http://vertrigo.sf.net`

Rich picture, ERD diagram a obrázek architektury pro návrh systému vytvořím v nástroji Dia 0.95[1]. Dia je vynikající program pro pohodlnou tvorbu všemožných diagramů a strukturních obrázků.

8.3 Nároky na provoz aplikace

8.3.1 Požadavky na hosting

Aby aplikace mohla fungovat, musí být zdrojové soubory nahrané na serveru a musí být vytvořená databáze, se kterou program pracuje. Dále musí být správně nakonfigurován soubor `connect.inc.php`, který obsahuje adresu, jméno a heslo databáze.

Nároky na hosting jsou:

- podpora PHP minimálně verze 4.0
- MySQL databáze

8.3.2 Požadavky na uživatele

Jelikož bude systém běžet na jednom serveru a uživatelé se k němu budou připojovat ze svých počítačů, jsou nároky na ně velice malé. Uživateli stačí pouze počítač připojený k síti Internet, vybavený libovolným webovým prohlížečem.

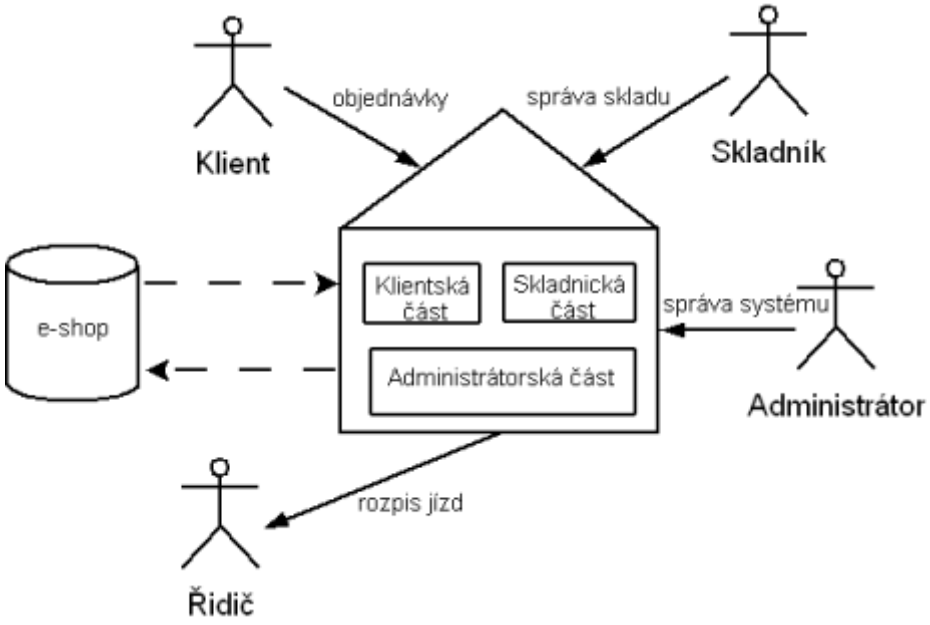
[1] `http://www.gnome.org/projects/dia/`

9 Návrh systému

Předtím, než začnu aplikaci programovat, je třeba vytvořit nákresy toho, jak bude systém fungovat, jaké interakce budou mezi jednotlivými částmi a jak bude vypadat návrh databáze.

9.1 Rich picture

Na následujícím obrázku je znázorněn tzv. rich picture, z kterého je patrné, jak bude aplikace fungovat, co všechno bude její součástí a co bude ovlivňovat.

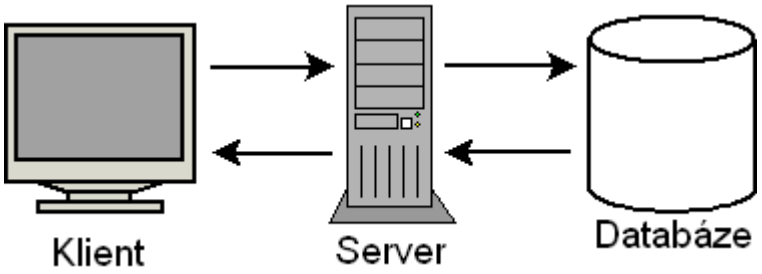


Obr. č. 9: Rich picture znázorňující návrh systému

Z obrázku je vidět, že je systém rozdělen na tři části. V klientské části zadává uživatel objednávky na zboží. V skladnické části se spravuje sklad a v administrátorské části je možno ovlivňovat a kontrolovat celý systém. Aplikace bude také poskytovat denní rozpis jízd pro řidiče. Přerušovanými čarami je naznačeno možné napojení systému na e-shop.

9.2 Architektura

Aplikace bude postavená na architektuře klient-server. Klient-server je síťová architektura, která odděluje klienta (často aplikaci s grafickým uživatelským rozhraním) a server. Jednotlivé instance klientů komunikují se serverem, který obvykle běží na vzdáleném počítači [13]. Data server zapisuje a vybírá z databáze.



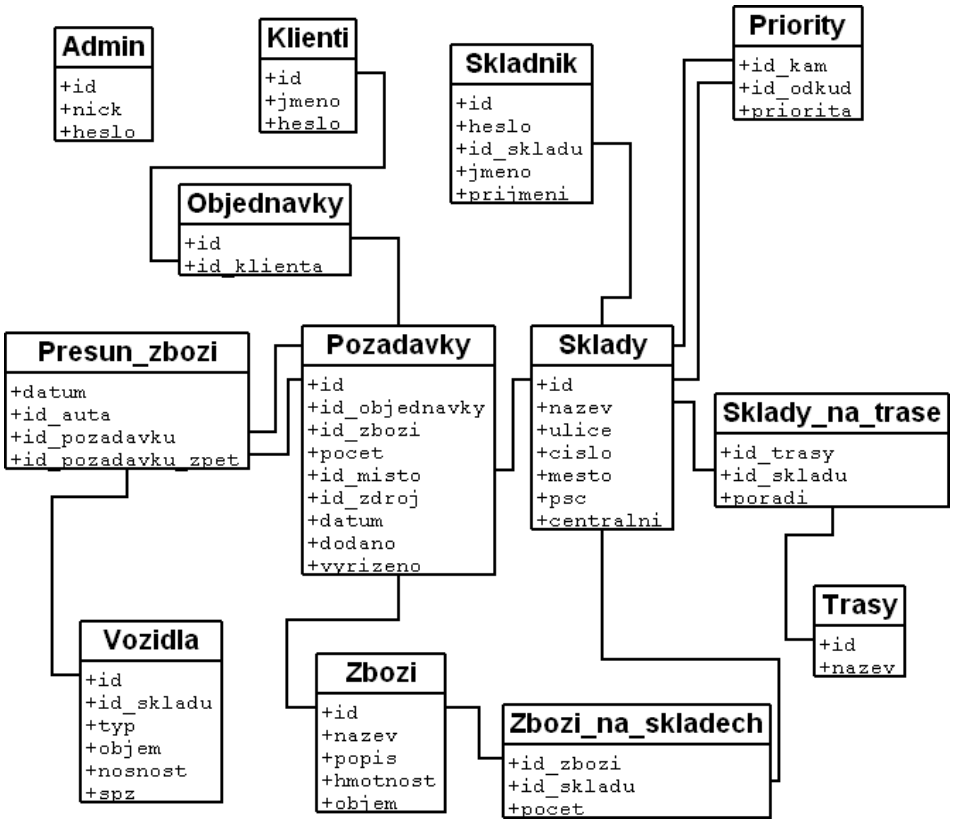
Obr. č. 10: Architektura klient-server

V tomto případě je klientem libovolný webový prohlížeč, já budu k testování používat aplikaci Mozilla Firefox[1] 1.5.0.6. Jako server bude sloužit server Apache[2] verze 2.0.59. Apache HTTP Server jsem si vybral, protože je to nejrozšířenější softwarový webový server. Ten bude přistupovat do databáze MySQL verze 5.0.24.

9.3 ERD

ERD diagram (Entity Relationship Diagram) popisuje statickou strukturu databáze. Entity se v těchto diagramech popisují pomocí obdélníků. Entitou je v tomto případě tabulka databáze. Pod názvem tabulky jsou uvedeny její atributy.

[1] http://cs.start.mozilla.com/firefox
[2] http://httpd.apache.org/



Obr. č. 11: ERD diagram

Na ERD diagramu jsou vidět vztahy mezi tabulkami databáze. Z obrázku je patrné, že hlavní úlohu hrají tabulky *Sklady* a *Pozadavky*, na které jsou napojeny ostatní. V tabulce *Sklady* uchovávám údaje o jednotlivých skladech (id, adresa a jestli se jedná o centrální sklad). Na tuto entitu jsou napojeny *Sklady_na_trase*, kde je uloženo id skladu, pořadí a na které trase se nachází. Seznam tras je uveden v tabulce *Trasy*.

Uložení zboží na skladech řeší tabulka *Zbozi_na_skladech*, kde je uvedeno id skladu, id zboží a počet. Název, popis a vlastnosti zboží (hmotnost, objem) se nalézají v tabulce *Zbozi.* Se *Sklady* ještě souvisí entita *Skladnik* s údaji o jménu, příjmení, heslu a id skladu, do kterého má uživatel přístup, a entita *Priority*, kde jsou uložené priority při vyhledávání nejbližšího skladu.

V tabulce *Pozadavky* uchovávám informace o tom jaké zboží, kolik a odkud (id_zdroj) kam (id_misto) se má převézt. Dále je zde uvedeno v jakém stavu se požadavek nachází. *Dodano* znamená, jestli je zboží na výdejním skladě, a *vyrizeno*, jestli si ho klient už vyzvedl. Z této

tabulky se ještě dozvíme datum a číslo objednávky, do které požadavek patří. Objednávky jsou zaznamenávány do entity *Objednávky* (id objednávky a id klienta, který objednávku vystavil). Údaje o klientech jsou v tabulce *Klienti* (id klienta, jméno a heslo).

Požadavky se vyřizují pomocí *Presun_zbozi*, kde je uveden datum, id požadavku a jaké vozidlo zboží poveze. Je-li číslo požadavku uvedeno v *id_pozadavku*, znamená to, že vozidlo veze zboží cestou k centrálnímu skladu. Je-li číslo požadavku uvedeno v *id_pozadavku_zpet,* znamená to, že vozidlo veze zboží cestou z centrálního do domovského skladu. Údaje o vozidlech se ukládají do tabulky *Vozidla*. Je zde id vozidla, id skladu, do kterého je přiděleno, typ, spz a vlastnosti (nosnost, objem).

Poslední tabulkou je *Admin*, kde se nalézají informace o administrátorovi aplikace (id, nick a heslo).

9.4 Od objednávky k výdeji zboží

V této kapitole popíšu, jak bude aplikace fungovat a jak se bude chovat při běžném provozu.

Při objednávání uvidí klient všechno zboží, které je v celém systému. Zvolí si druh, počet a výdejní místo. Přitom se nestará, jestli je všechno zboží na výdejním skladu. Úkolem aplikace je teď vyhodnotit objednávku. Pokud není dostatek zboží na skladě, vystaví systém požadavek na přesunutí požadovaného množství z nejbližších skladů. Ten zapíše do tabulky *Pozadavky*. Nejbližší sklady systém rozpozná z tabulky *Priority*. Systém se prohledává do té doby, než je splněna objednávka nebo než dosáhne posledního skladu.

Takto si klienti objednávají zboží po celý den. Na konci dne provede administrátor, stiskem tlačítka, výpočet přesunu zboží na další den. Systém vyhodnotí požadavky a připraví rozpis jízd pro jednotlivá auta. Přitom dbá na nosnost, objem vozidel a jejich optimální využití.

Po spočítání přesunu, se ve *skladnické části* objeví aktuální rozpis jízd pro vozidla z příslušného skladu. Bude rozdělen na dvě části, první

bude obsahovat sklady a zboží, které má auto vyřídit během cesty do centrálního skladu. V druhé budou požadavky na přesun zboží směrem od centrálního do domovského skladu.

Dále se skladníkovi vygeneruje seznam zboží, které by daný den mělo do skladu dorazit. Při vykládce z vozu pak označí zboží jako dodané. Ve stejné chvíli se klientovi v kontrole objednávky vypíše, že zboží je na skladě a že si ho může vyzvednout. Jakmile to učiní, označí skladník požadavek za vyřízený.

10 Vývoj aplikace

Nyní se zaměřím na vývoj samotného systému. Budu popisovat, jak jsem postupoval při programování, popíšu strukturu adresářů a souborů, které jsem vytvořil, a představím některé důležité části kódu.

10.1 Postup

Jako první jsem začal programovat administrátorskou část. Nejprve bylo nutné si uvědomit jaké funkce má administrátor mít k dispozici. Z toho vyplynuly tyto části:

- Sklady – úprava a zakládání skladů
- Skladníci – úprava a vytváření skladníků
- Vozidla – úprava a přidávání vozidel
- Zboží – úprava a vytváření zboží
- Požadavky – kontrola požadavků a algoritmus přesunu zboží
- Trasy – vytváření a úprava tras
- Klienti – úprava a přidávání klientů
- Odhlášení – odhlášení ze systému

V další fázi vývoje jsem pracoval na skladnické části. Pro všechny skladníky jsem vytvořil jednotné prostředí, lišící se pouze v obsahu, který závisí na tom, do jakého skladu je skladník přiřazen. Segmenty skladnické části jsou následující:

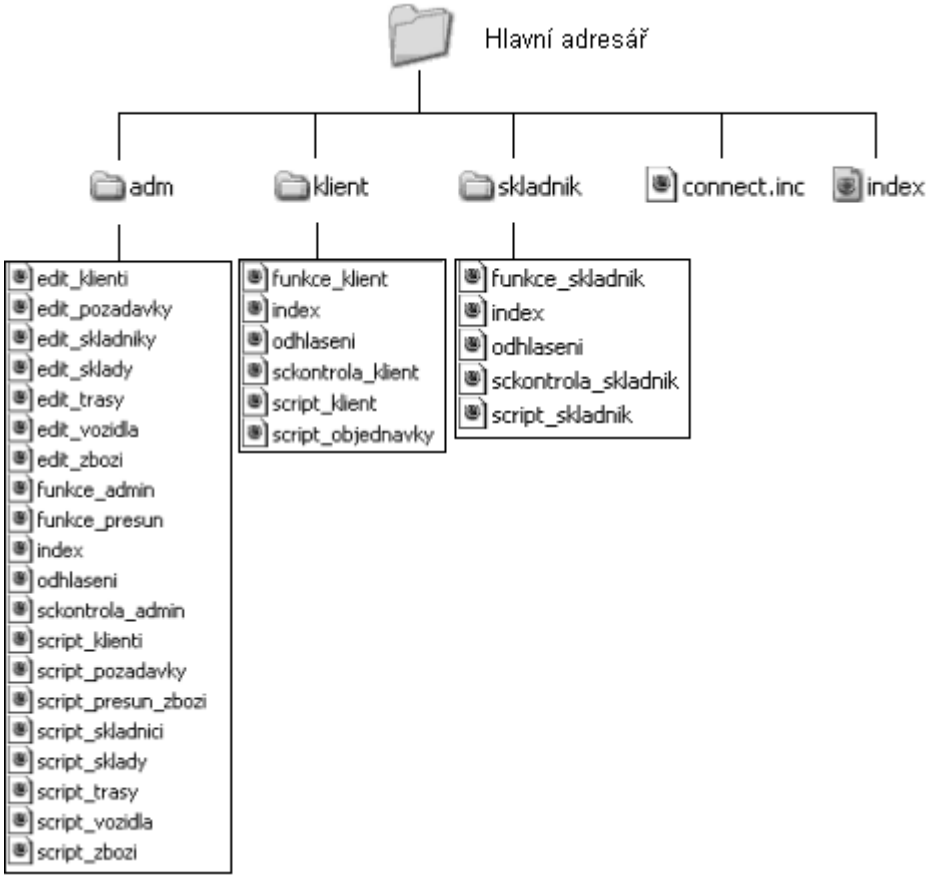
- Editace skladu – úprava zboží, vozidel a priorit skladu
- Vykládka – kontrola zboží, které se má v určitý den přijmout
- Rozpis pro auta – výpis vozidel a jejich rozpisu jízd
- Zboží k vydání – kontrola výdeje zboží klientům
- Odhlášení – odhlášení ze systému

Poslední součástí, kterou jsem programoval bylo klientské prostředí, které slouží k zadávání objednávek a tím nahrazuje napojení aplikace na e-shop. Klientské možnosti jsou:

- Nová objednávka – výběr výdejního skladu a objednávka zboží
- Objednávky – kontrola stavu objednávek

10.2 Struktura adresářů a souborů

Vytvořená webová aplikace se skládá z několika adresářů a souborů (viz. obr. 12). Hlavní adresář obsahuje tři podadresáře, které jsou pojmenovány podle toho jaké soubory obsahují. V adresáři *adm* jsou soubory obsluhující administrátorskou část. Ve složce *klient* jsou soubory obsluhující klientskou část a adresář *skladnik* obsahuje soubory pro funkce skladnické části. V hlavním adresáři se ještě nachází soubor `connect.inc.php` a `index.html`.



Obr. č. 12: Struktura adresářů a souborů

10.2.1 Popis souborů

V souboru `connect.inc.php` jsou informace o přístupu do databáze a nastavení znakové sady, ve které aplikace komunikuje. `Index.html` obsahuje pouze rozcestník mezi jednotlivými částmi.

V adresáři *adm* slouží jako hlavní stránka soubor `index.php`. Probíhá zde přihlášení a poté načtení hlavního obsahu. Soubory, které

jsou pojmenovány `edit_něco.php` zajišťují zobrazení stránek, mezi kterými se pohybuje administrátor. Formuláře, které obsahují, se zpracovávají vždy v příslušném souboru `script_něco.php`. `Funkce_admin.php` je includovaný do všech souborů v adresáři *adm*. Obsahuje totiž funkce, které zobrazují hlavní menu a formulář na přihlášení. Kontrola přihlašovacích údajů probíhá v `sckontrola-_admin.php`, odhlášení pak v `odhlaseni.php`. Obdobné soubory jsou i v ostatních adresářích. Soubor `script_presun_zbozi.php` obsahuje algoritmus na výpočet rozpisu jízd pro jednotlivá vozidla. Přičemž používá funkce, které jsou ve `funkce_presun.php`.

Adresář *klient* má jako hlavní stránku `index.php`, ve které se provádí přihlašování, objednávání zboží i kontrola objednávek. Soubor `script_klient.php` obsluhuje formulář na výběr výdejního skladu, soubor `script_objednávky.php` pak vlastní objednávku. Posledním souborem je `funkce_klient.php`, ten obsahuje funkci na zobrazení přihlašovacího formuláře a funkci na zadávání požadavků do tabulky *Pozadavky*.

Ve složce *skladnik* jsou podobné soubory jako v těch předcházejících. Hlavní stránka je `index.php. Script_sklad-nik.php` zpracovává všechny formuláře a `funkce_skladnik.php` má funkci na přihlášení a zadávání požadavků do tabulky *Pozadavky.*

10.3 Ukázky vybraných částí kódu

V této kapitole předvedu některé stěžejní součásti kódu.

10.3.1 Objednávka zboží z ostatních skladů

Následující kód řeší případ, když na výdejním skladě není dostatek zboží, které bylo objednáno, a musí se zadat požadavky na přesun zboží z ostatních skladů. Ty se vybírají podle priorit určených v tabulce *Priority.* Tento kód je součástí skriptu, který se nachází v souboru `script_objednávky.php` v adresáři *klient.*

```
// --- kdyz na skladu neni dostatek zbozi ----
  if($rozdil<0){
    // --- vyberu vse co je ve vydejnim skladu ----
    if($pocet_skladem>0){

zadej_pozadavek($id_objednavky,$id_zbozi,$pocet_skladem,$id_skladu,$id_skladu);
       $pocet_skladem=0;
       // ---- Zapis zmen do databaze ----
       mysql("bsklady","UPDATE zbozi_na_skladech SET pocet='$pocet_skladem'
                    WHERE (id_zbozi='$id_zbozi' AND id_skladu='$id_skladu')");
    }
     $doobjednat=$rozdil*(-1);
     // ---- Vyber z ostatnich skladu podle priority ----
     $dotaz_priority=mysql("bsklady","SELECT id_odkud, priorita FROM priority
                          WHERE (id_kam='$id_skladu') ORDER BY priorita DESC");
     $pocet_skladu=mysql_NumRows($dotaz_priority);

     // ---- Projdu ostani sklady a vyberu z nich zbozi dokud nebude splnena objednavka
    $i=0;
    while ($i<$pocet_skladu AND $doobjednat>0):
      $id_odkud=mysql_result($dotaz_priority,$i,"id_odkud");
      $dotaz_nasklade=mysql("bsklady","SELECT pocet FROM zbozi_na_skladech
                           WHERE (id_zbozi='$id_zbozi' AND id_skladu='$id_odkud')");
      $pocet_nasklade=mysql_result($dotaz_nasklade,0,"pocet");
      $pom_rozdil=$pocet_nasklade-$doobjednat;  // zjistim rozdil
     // ---- kdyz je na skladu dostatek zbozi -----

if($pom_rozdil>=0){zadej_pozadavek($id_objednavky,$id_zbozi,$doobjednat,$id_skla
du,$id_odkud);
       $pocet_nasklade=$pocet_nasklade-$doobjednat;
       $doobjednat=0;
        // ---- Zapis zmen do databaze ----
        mysql("bsklady","UPDATE zbozi_na_skladech SET pocet='$pocet_nasklade'
                                WHERE (id_zbozi='$id_zbozi' AND
id_skladu='$id_odkud')");
       }
     // --- kdyz na skladu neni dostatek zbozi ----
      if($pom_rozdil<0){
       // --- vyberu vse co je ve skladu ----
       if($pocet_nasklade>0){

zadej_pozadavek($id_objednavky,$id_zbozi,$pocet_nasklade,$id_skladu,$id_odkud);
         $pocet_nasklade=0;
         // ---- Zapis zmen do databaze ----
         mysql("bsklady","UPDATE zbozi_na_skladech SET pocet='$pocet_nasklade'
                          WHERE (id_zbozi='$id_zbozi' AND id_skladu='$id_odkud')");
       }
      $doobjednat=$pom_rozdil*(-1);

     }
     $i++;
     if($i==$pocet_skladu AND $doobjednat>0){
       $nedostupne=$nedostupne+$doobjednat;
       $doobjednat=0;
       $chyba=1;
      }
    endwhile;
  }
```

10.3.2 Algoritmus počítání přesunu zboží

Tento algoritmus se nachází v souboru `script_presun-_zbozi.php` v adresáři *adm* a vypočítává přesun zboží, který se bude realizovat v následujícím dni. Nejprve spočítá pohyb zboží po všech trasách směrem do centrálního skladu a potom zpět.

```
// dotaz na zitrejsi datum
  $dotaz_vkladanedatum=mysql("bsklady","SELECT current_date +
INTERVAL 1 DAY");
   $datum1=mysql_result($dotaz_vkladanedatum,0,"current_date + INTERVAL
1 DAY");
  // dotaz na trasy
  $dotaz_trasy=mysql("bsklady","SELECT id FROM trasy");
  $pocet_tras=mysql_num_rows($dotaz_trasy);
  // nejprve spocitame prevoz zbozi do centralniho skladu
  $j=0;
  while ($j<$pocet_tras):
    $id_trasy=mysql_result($dotaz_trasy,$j,"id");
    spocitejsmercentral($id_trasy,$datum1);
    $j++;
  endwhile;
   // potom spocitame prevoz zbozi z centralniho skladu
  $i=0;
  while ($i<$pocet_tras):
    $id_trasy=mysql_result($dotaz_trasy,$i,"id");
    spocitejsmerodcentral($id_trasy,$datum1);
    $i++;
  endwhile;
```

10.3.3 Přihlašování a bezpečnost

Z důvodu bezpečnosti je ve všech souborech aplikace umístěn skript, který zkoumá jestli je uživatel přihlášen. Když tomu tak není, vypisuje přihlašovací formulář. Tento pochází ze souboru `index.php` v adresáři *adm.*

```
include 'funkce_admin.php';
session_start();

if(!$_SESSION['login_admin']){
      prihlas_admin_form();
  }
  else{
```

10.3.4 Přidání skladu do trasy

Následující skript řeší přidání skladu do trasy. Skriptu předávám pořadí skladu, za který chci umístit nový. Algoritmus musí udělat místo pro přidávaného a všem následujícím skladům zvýšit pořadí o jedna. Skript se nalézá v souboru `script_trasy.php` v adresáři *adm*.

```
// --- Vyber skladu, ktere jsou za skladem kam se bude vkladat novy
    $dotaz=mysql("bsklady","SELECT poradi, id_skladu FROM sklady_na_trase
                    WHERE (id_trasy='$id_trasy'
                     AND poradi > '$za_sklad') ORDER BY poradi");
    $pocet=mysql_NumRows($dotaz);
    $j=0;
    while ($j<$pocet):
     $poradi= mysql_result($dotaz, $j, "poradi");
     $id_skladu= mysql_result($dotaz, $j, "id_skladu");
     $poradi++; // zvyseni poradi o 1, aby se udelalo misto pro novy sklad
     mysql("bsklady","UPDATE sklady_na_trase SET poradi= '$poradi'
            WHERE (id_trasy='$id_trasy' AND id_skladu = '$id_skladu') ");
    $j++;
    endwhile;

    $poradi_novy=$za_sklad+1;

    // ---- Zápis do databáze -----
    mysql("bsklady","INSERT INTO sklady_na_trase (id_trasy, id_skladu, poradi)
                    VALUES ('$id_trasy','$id_novy','$poradi_novy')");
```

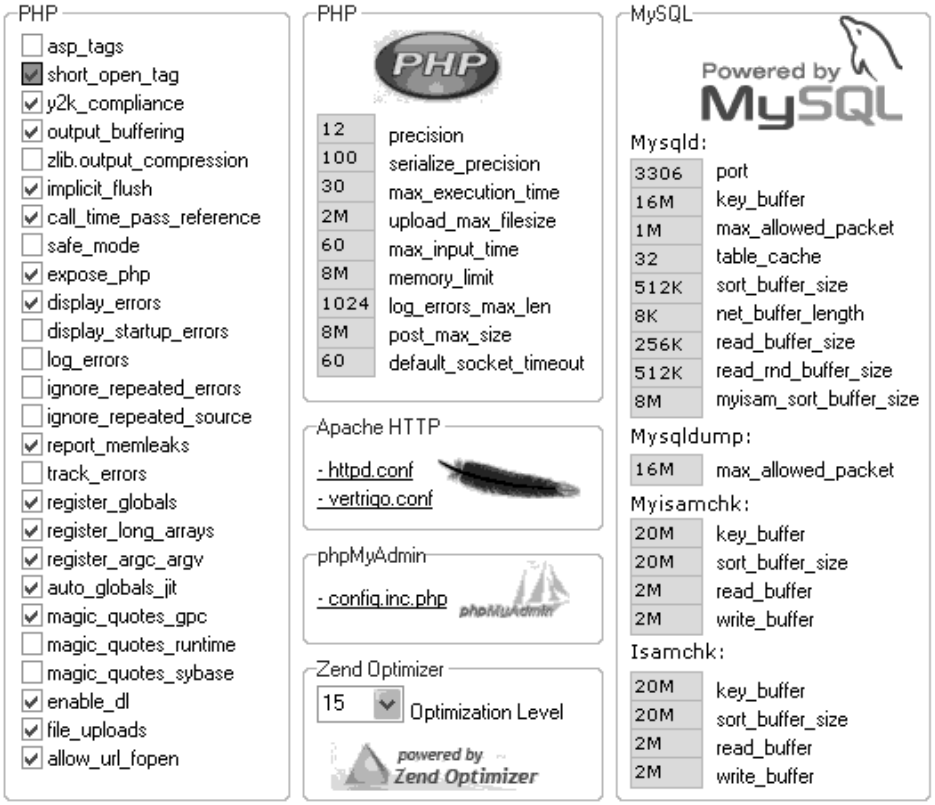
10.3.5 Nakládání vozidla

Při nakládce auta sleduje skript hmotnost a objem nakládaného zboží a hlídá, aby nebyla překročena nosnost a objem přepravního prostoru vozidla. Kód je součástí funkcí v souboru `funkce-_presun.php` v adresáři *adm.*

```
$pomhmot=$pocet*$hmotnost1;
$pomobj=$pocet*$objem1;

$hmotnost_nakladu=$hmotnost_nakladu+$pomhmot;
$objem_nakladu=$objem_nakladu+$pomobj;

if($nosnost_auta<$hmotnost_nakladu OR $objem_auta<$objem_nakladu){
   $pom=0; // pocet ks
   // zjistime pocet zbozi z pozadavku, ktere se do auta nevlezou
   while ($nosnost_auta<$hmotnost_nakladu OR
$objem_auta<$objem_nakladu):
    $hmotnost_nakladu=$hmotnost_nakladu-$hmotnost1;
    $objem_nakladu=$objem_nakladu-$objem1;
    $pom++;
   endwhile;
   // Upravim pocet zbozi v pozadavku a na prebyvajici zbozi vystavim novy
pozadavek, ktery se vyridi v dalsich dnech
   $pocet=$pocet-$pom;

mysql("bsklady","UPDATE pozadavky SET pocet = '$pocet' WHERE id =
'$id_pozadavku'");


zadej_pozadavek($id_objednavky,$id_zbozi,$pom,$id_kam,$id_centralni,$datu
m,0);
   $j=$pocet_pozadavku; // ukoncime nakladku
 }
   mysql("bsklady","INSERT INTO presun_zbozi (datum, id_auta,
id_pozadavku_zpet) VALUES ('$datum1', '$id_auta', '$id_pozadavku')");
```

11 Testování systému

Testování funkčnosti systému jsem prováděl za parametrů, které uvádí následující obrázek. Používal jsem při tom aplikaci VertrigoServ 2.10, která je blíže popsána v kapitole 8.2.



Obr. č. 13: Nastavení PHP a databáze MySQL při testování

12 Závěr

Cílem této bakalářské práce bylo představit a zhodnotit současné technologické trendy při vývoji webových aplikací, zhodnotit dostupné systémy řízení skladových zásob a spediční systémy, a vytvořit vlastní aplikaci pro sledování a převod zboží mezi jednotlivými sklady s využitím vozového parku.

Povedlo se vytvořit aplikaci, která je schopna plnit požadavky na správu skladů a vozidel. Aplikace umožňuje vytvořit strukturu skladů podle uživatelových potřeb. Pro výpočet optimalizace přesunu zboží mezi jednotlivými sklady byl implementován vlastní algoritmus. Další algoritmy pro zpracování různých struktur rozmístění poboček nebyly z důvodu jejich náročnosti a rozsahu bakalářské práce implementovány.

Předpokladem systému je, že bude napojen na nějaký další systém např. e-shop. Vzhledem k náročnosti nebylo propojení aplikováno. Pro představení funkčnosti systému byly výstupy z e-shopu nahrazeny výstupy z klientské části, která byla naprogramována.

V aplikaci byly zohledněny postupy nakládání, vykládání a přesunu zboží, které se používají ve spedici. Tím je aplikace připravena k použití v běžné praxi.

ÚDAJE PRO KNIHOVNICKOU DATABÁZI

Název práce	Systém pro řízení skladu ve více lokalitách
Autor práce	Jan Podlešák
Obor	IT
Rok obhajoby	2007
Vedoucí práce	Ing. Lukáš Čegan
Anotace	Teoretická část se zaměřuje na vývoj webových aplikací. V implementační části je naprogramován systém pro řízení skladu ve více lokalitách.
Klíčová slova	IT, webová aplikace, sklad, řízení zásob

SEZNAM POUŽITÉ LITERATURY

[1] BERNARD, B., *Využití objektových technologií při vývoji webových aplikací*, Praha: Vysoká škola ekonomická v Praze, 2004, s. 1-2, 6-7

[2] *Programování webových aplikací* [online]. Adaptic, , [cit. 2007-03-25], Dostupný z WWW: http://www.adaptic.cz/novy-web/programovani-webovych-aplikaci.htm

[3] LACKO, L., *Web a databáze*, Praha: Computer Press, 2001. ISBN 80-7226-555-5

[4] *PHP* [online]. Wikipedie, [cit. 2007-03-26], Dostupný z WWW: http://cs.wikipedia.org/wiki/PHP

[5] *Active Server Pages* [online]. Wikipedie, [cit. 2007-03-26], Dostupný z WWW: http://cs.wikipedia.org/wiki/Active_Server_Pages

[6] *Java Server Pages* [online]. Pavel Nejedlý, [cit. 2007-03-27], Dostupný z WWW: http://atrey.karlin.mff.cuni.cz/~bim/pub/jsp/referat/referat.html

[7] *ASP.NET* [online]. Wikipedie, [cit. 2007-03-26], Dostupný z WWW: http://cs.wikipedia.org/wiki/ASP.NET

[8] *Co je to databáze MySQL?* [online]. Artic studio, slovníček pojmů, , [cit. 2007-03-27], Dostupný z WWW: http://www.artic-studio.net/slovnicek-pojmu/databaze-mysql/

[9] SVOBODA, V., LATÝN, P., *Logistika*, Praha: Vydavatelství ČVUT, 2003. s. 47, 49, 52, 57-59

[10] DANĚK, J., KŘIVDA, V., *Základy dopravy*, Ostrava: Vysoká škola báňská – technická univerzita Ostrava, 2003. s. 114-116, ISBN 80-248-0410-7

[11] *TRAVELING SALESMAN PROBLEM: Tour Construction Algorithms* [online]. Online logistics tutorial, School of Industrial and System Engineering Georgia Institute of Technology, Atlanta, GA, USA, [cit. 2007-05-10], Dostupný z WWW: http://www2.isye.gatech.edu/logisticstutorial/vehicle/tsp/tsp007__.htm

[12] *Klient-server* [online]. Wikipedie, [cit. 2007-05-13], Dostupný z WWW: http://cs.wikipedia.org/wiki/Klient-server

[13] *Tutoriál Web Services* [online]. KUBA, M., Superpočítačové Centrum Brno, ÚVT MU, [cit. 2007-05-11], Dostupný z WWW: http://dior.ics.muni.cz/~makub//soap/tutorial.html#cojsou