S T U
F C H P T

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
Fakulta chemickej a potravinárskej technológie
Ústav informatizácie, automatizácie a matematiky
Oddelenie informatizácie a riadenia procesov

Informačný systém pre sklad motorových palív v Oktan a. s. Kežmarok

Diplomová práca

Vypracoval: Bc. Peter Šlapanský
Vedúci diplomovej práce: Ing. Ľuboš Čirka, PhD.

Bratislava, 2008

Slovenská technická univerzita v Bratislave
Oddelenie informatizácie a riadenia procesov

Fakulta chemickej a potravinárskej technológie
Akademický rok: 2007/2008

S T U
F C H P T

ZADANIE DIPLOMOVEJ PRÁCE

Autor práce:	**Bc. Peter Šlapanský (16176)**
Študijný program:	chemické inžinierstvo a riadenie procesov
Zameranie:	riadenie procesov
Študijný odbor:	5.2.17 chemické inžinierstvo
Vedúci práce:	Ing. Ľuboš Čirka, PhD.
Konzultant:	Martin Baláž

Názov témy: **Informačný systém pre sklad motorových palív v Oktan a.s. Kežmarok**

Rozsah práce: 60

Riešenie zadania práce od: 18. 02. 2008
Dátum odovzdania: 23. 05. 2008

Slovenská technická univerzita v Bratislave Fakulta chemickej a potravinárskej technológie dekanát 5

L. S.

Bc. Peter Šlapanský
riešiteľ diplomovej práce

prof. Ing. Dr. Miroslav Fikar
vedúci pracoviska

prof. Ing. Vladimír Báleš, DrSc.
garant študijného programu

Prehlásenie

Čestne prehlasujem, že som diplomovú prácu vypracoval samostatne podľa pokynov vedúceho diplomovej práce a s použitím literatúry, ktorú uvádzam v osobitnom zozname. Ďalej prehlasujem, že súhlasím s akýmkoľvek prípadným využitím tejto diplomovej práce ústavom.

V Bratislave, 20. mája 2008 ….....

Chcel by som sa poďakovať Ing. Ľubošovi Čirkovi, PhD., za odborné vedenie, rady a pripomienky aj ostatným, najmä Ing. Martinovi Balážovi z firmy Intech Control, ktorí mi pomohli pri vypracovaní mojej diplomovej práce.

Abstrakt

Práca zahŕňa štúdium potrebnej teórie informačných systémov, výber vhodného databázového systému a programovacieho/skriptovacieho jazyka, pričom sa predpokladá architektúra klient-server. Obsahuje návrh tabuliek databázy tak, aby boli použiteľné pre čo najširšie spektrum prípadov, zdokumentované vytvorené prepojenia a možnosti práce s údajmi v databáze. Zahŕňa programovanie/skriptovanie užívateľského rozhrania a aplikačnej vrstvy pod ním na základe požiadaviek firmy Intech Control tak, aby bolo možné pripravovať a tlačiť zostavy, prezerať si databázové tabuľky alebo ich časti a vytvárať prehľady.

Abstract

This diploma thesis involves study of necessary theory of information systems, choice of the most suitable database system and programming/scripting language, presuming use of client-server architecture. It includes design of database tables in a way that enables their use in the widest possible spectrum of cases, documents the designed relations and possibilities of working with data stored in the database. Programming/scripting of application interface and application layer underneath are also involved according to the requirements of Intech Control in such a manner that preparation and printing of reports and summaries, viewing of database tables or parts thereof are possible.

Obsah

1 Úvod

Informačný systém (IS) je systém osôb, záznamov údajov a aktivít, ktoré spracúvajú údaje a informácie v danej organizácii, vrátane manuálnych a automatizovaných procesov. Komponentom IS, ktorý sa zaoberá informačnými technológiami pre IS sú počítačové informačné systémy (computer-based information systems) [1].

Počítačových informačných systémov je veľa a často sú vytvorené alebo upravené pre plnenie špecifických úloh. Príkladom môže byť informačný systém pre predaj tovaru, pre správu skladov a podobne. Táto diplomová práca sa zaoberá tvorbou informačného systému pre sklad motorových palív.

Pôvodný návrh predpokladal existenciu dvoch vrstiev, nízko úrovňovej a tejto, ktorá predstavuje nadstavbu, umožňujúcu prezeranie si, úpravu a tlač informácií o príjme a výdaji motorových palív, ktoré má poskytovať predovšetkým nízko úrovňová časť. Tá má obsluhovať výdajné zariadenia ako sú napríklad Petrocount alebo Microload, získavať z nich informácie o vykonaných operáciách a zapisovať vybrané údaje do databázy. Súčasťou tejto práce je aj výber databázového systému a dizajn databázy tak, aby bola použiteľná pre obe časti.

Existujúcimi riešeniami pre výdaj paliva sú napríklad terminálové automatizačné systémy (Terminal Automation Systems) Cotas spoločnosti Mess- und Fördertechnik Gwinner GmbH & Co a Daniel spoločnosti Emerson Electric Co.

Firma Intech Control požadovala vytvorenie systému, ktorý by mohol byť pre ich činnosť alternatívou k týmto dvom systémom.

2 Výber databázového systému

2.1 Niektoré používané pojmy

Databázové servery sú programy, ktoré poskytujú databázové služby iným programom podľa modelu klient-server [1].

Databázový systém zaisťuje samotnú správu databáz a je množinou komponentov určených na definovanie, vytváranie a manipuláciu s databázou [2]. Na správu relačnej databázy sa potom používa relačný databázový systém (Relational Database Management System, RDBMS).

SQL je jazyk obvykle používaný na komunikáciu s databázovým systémom, správu databáz, tabuliek a údajov v nich [2]. Je jazykom na manipuláciu s údajmi (Data Manipulation Language, DML). Časť SQL tvorí jazyk na definovanie údajov (Data Definition Language, DDL). Príkladom čistého DDL je XML Schema (XML schema je popisom typu XML dokumentu, definujúcim obmedzenia štruktúry a obsahu XML) v kontexte XML [1].

Počítačové databázy sú teda podľa databázových modelov štruktúrované úložiská údajov. V súčasnosti je pravdepodobne najpoužívanejším modelom relačný databázový model, aj keď sa používajú aj niektoré iné modely [1]. Pre túto prácu a zadanú tému budú použité práve relačné databázy.

Ich základnou vlastnosťou je, že v nich možno definovať vzťahy medzi jednotlivými tabuľkami. Tie sú potom kľúčové pri získavaní údajov z viacerých tabuliek databázy, pretože tabuľky sa prepájajú práve vzťahmi. Dobré navrhnutie vzťahov je jedným z faktorov, ktorý rozhoduje o použiteľnosti databázy a výkone databázového servera. Výkon databázového servera nemusí mať výrazný vplyv pri výbere databázového software. Obvykle je ale požadovaný čo najvyšší výkon a tak je to aj v tomto prípade. Samozrejme, splnené musia byť aj iné požiadavky.

2.2 Požiadavky pri výbere databázového servera

Informačné systémy sa typicky využívajú v organizáciách, kde je potrebné spravovať a manipulovať s údajmi, ktoré organizácia prijíma alebo vydáva.

Ak by malo dôjsť k výpadku informačného systému, môže to organizáciu alebo jej časť dočasne ochromiť, hrozí strata pracovného času a teda aj peňazí.

Stabilita a **robustnosť** zvyknú preto byť druhým najdôležitejším kritériom pri výbere informačného systému a jeho komponentov z vyššie uvedených dôvodov.

Zabezpečenie údajov a celého systému je jedným z rozhodujúcich spôsobov ochrany proti prípadnej priemyselnej špionáži. Zabezpečenie je komplexná úloha a závisí od všetkých komponentov systému vnútri aj mimo počítačov.

Výkon a **škálovateľnosť** systému často rozhodujú po splnení ostatných kritérií. Ide ale o pojmy, ktoré sa niekedy ťažko zlučujú. Pod škálovateľnosťou rozumejme schopnosť používaného software podávať vyšší výkon pri zvyšovaní výkonu/množstva hardware, na ktorom software beží. Je snaha, aby škálovateľnosť systému ako celku bola blízka stúpajúcej priamke alebo lepšia.

Cena a **celkové náklady** sú dnes rozhodujúcimi faktormi obmedzujúcimi výber prakticky čohokoľvek.

Podpora býva elementom často podceňovaným a jej nedostatok môže mať za následok straty a zmenšenie produktivity. Do podpory patria produktová dokumentácia, podpora zo strany výrobcu a tretích strán, ktorá zahŕňa expertov, ktorí by mali byť schopní vyriešiť zákazníkov problém, ak zákazník, operátor alebo miestny správca systému nie je problém schopný vyriešiť, resp. ak je pre nich problém neriešiteľný. Môže byť dokupovaná aj samostatne. Pri software je nezanedbateľné obdobie, na ktoré je poskytovaná takáto podpora ako aj vydávanie bezpečnostných aktualizácií a opráv prípadných chýb či nedostatkov.

Platforma je v tomto kontexte hardware a software, ktorý určí ďalšie možnosti použitia. Zo software je to predovšetkým operačný systém a prípadné dodatočné programy a knižnice potrebné pre beh informačného systému a využitie údajov ním spravovaných.

Jednoduchosť obsluhy a **správy** sa premieta do nákladov a potreby podpory.

2.3 Porovnávané databázové systémy

Okrem Microsoft SQL Server, ktorý je spustiteľný len v operačnom systéme Windows, podporujú porovnávané databázové systémy aj iné operačné systémy. Podpora operačných systémov databázovými systémami sa zvykne meniť, všetky ale podporujú v nejakej svojej verzii systémy Microsoft Windows (založené na Windows NT) a Linux. V prípade Linuxu zvyknú databázové systémy s uzavretým zdrojovým kódom podporovať len určité Linuxové distribúcie.

Zadávateľom bol zvolený operačný systém Microsoft Windows. Dôvodom pre voľbu Windows je skutočnosť, že existujú varianty Windows, ktoré majú na trhu operačných systémov pre osobné počítače a pracovné stanice dominantný podiel. Štatistiky sa rôznia, ale odhaduje sa, že na viac ako 90% osobných počítačov a lacnejších pracovných staníc je nainštalovaný a používaný niektorý variant Windows. Z toho sa dá predpokladať, že firmy s dostatkom odvahy a vedomostí by si mohli spravovať server s Windows Server do istej miery aj bez externej pomoci alebo špeciálne školených zamestnancov.

Databázové systémy vybrané do užšieho okruhu na základe požiadaviek v časti 2.2 sú:

- MySQL
- PostgreSQL
- Firebird
- Microsoft SQL Server 2005 Express Edition
- Oracle Database 10g Express Edition
- IBM DB2 Express-C 9.5

Krátko som testoval aj Ingres 2006 Release 3, ale napriek tomu, že je k dipozícii tzv. Community Edition, ktorá je zdarma a naviac má otvorený zdrojový kód pod licenciou GNU GPL v. 2 som sa rozhodol ju vylúčiť, pretože ostatné databázové systémy boli už len pri čítaní, za použitia konektora pre Ingres (php_ingres), rýchlejšie a vlastnosťami postačujúce.

V ďalšom texte budem porovnávať výkon a vlastnosti jednotlivých databázových systémov na základe požiadaviek, ktoré sú predpokladané u informačného systému tohto rozsahu.

2.4 Výkonnostné porovnanie vybraných databázových systémov

Porovnával som výkon vybraných databázových systémov pri trocha hlavných operáciách, ktoré budú použité pri manipulácii s údajmi v databáze. Sú to:

1. výber údajov z databázy
2. zápis údajov do databázy a následný výber údajov
3. jednoduché využitie relačných vlastností vybraných databáz pri získaní obsahu po spájaní tabuliek

Predpokladá sa, že cieľová aplikácia bude mať počas reálneho nasadenia pomerne malý počet pripojení k databáze, teda aj pomerne malý počet súčasne pripojených užívateľov. Mala by byť ale schopná zvládnuť náhle väčšie množstvo zápisov a aj vyberanie väčšieho množstva údajov. Nepredpokladá sa iné ako občasné využitie SQL otázok na odstraňovanie a aktualizáciu údajov, preto neboli testované SQL otázky typu `UPDATE` a `DELETE`.

Každý z databázových systémov bol vyladený tak, aby používal zhruba rovnaké množstvo RAM, ak to bolo možné. Ak nie, pokúsil som sa nastaviť parametre, ktoré mali mať podobný účinok na rovnaké hodnoty. Konfiguračné súbory sa nachádzajú na DVD, ktoré je prílohou k diplomovej práci.

IBM DB2 Express-C 9.5 má v konfiguračných nástrojoch program na nastavovanie konfigurácie, ktorá by mala priamo ovplyvňovať výkon databázového systému [3].

Pre Oracle Database 10g Express Edition je možné meniť obmedzenia používaného množstva RAM, pri inštalácii sa toto množstvo nastavuje automaticky a jeho úpravy za behu systému neodporúčam, keďže aj drobá chyba v zmene konfigurácie môže znemožniť používanie databázy na istý čas.

Ukázalo sa, že prakticky všetky databázové systémy, okrem MySQL s MyISAM, využívajú pevný disk veľmi intenzívne. Preto je pravdepodobné, že pri použití výkonnejšieho diskového subsystému by dopadli testy priaznivejšie pre systémy, ktoré nepodávali dobrý výkon. Práve tieto pracovali s pevným diskom v zvýšenej miere.

Testovacia tabuľka pre prvý a tretí test obsahovala 100 000 riadkov. Zvolil by som väčšie množstvo, ale kvôli obmedzeniam testovacieho počítača a množstvu databázových systémov na ňom nainštalovanom som zvolil menší počet riadkov. Ako zdroj údajov na naplnenie databázy mi v tomto prípade poslúžili údaje, ktoré boli vytvorené pre projekt na testovanie databáz OSDB (The Open Source Database Benchmark) [4].

Pri testovaní som použil skripty v jazyku PHP, ktoré som si zostavil pre tento účel. Na tvorbu grafov zo získaných časových údajov som použil voľne dostupnú PHP knižnicu JpGraph.

Rýchlosť prístupu k databáze síce závisí od kvality modulu pre prístup k danému databázovému serveru, ale vzhľadom na to, že žiadny z použitých modulov nebol v prvej verzii a existuje istý čas som predpokladal, že prípadné výkonnostné rozdiely medzi jednotlivými modulmi budú zanedbateľné v porovnaní s rozdielom vo výkonnosti databázových systémov. Pri neprijatí tohto predpokladu by sa testovanie výrazne predĺžilo, čomu som sa chcel vyhnúť.

Testovanie bolo uskutočnené programom `ab`, Apache HTTP server benchmarking tool, teda nástroj na testovanie výkonu HTTP servera Apache. Program `ab` bol volaný z jednoduchého dosovského dávkového skriptu (`.bat`) v tvare

```
@ab -kc 20 -n 1000 {url},
```

kde prepínač `-k` zapína použitie požiadaviek keep-alive, prepínač `-c` a číslo za ním (20) udáva počet súčasných požiadaviek na adresu `{url}` a prepínač `-n` udáva celkové množstvo požiadaviek, ktoré majú byť vykonané programom `ab`.

Počet súčasných pripojení predstavuje predpokladaný maximálny počet u reálnej aplikácie, teda tohto informačného systému.

Celkový počet požiadaviek 1000 som zvolil preto, lebo bol časovo výhodný a nepresnosť oproti použitiu 10 000 požiadaviek bola najviac v desatinách sekundy.

Pred testovaním výkonu pri čítaní som defragmentoval diskový oddiel, na ktorom sa nachádzali databázy a systém. Použitý systém súborov bol NTFS, predvolená veľkosť bloku (4096 bytov). Podobne som postupoval pri testovaní paralelného čítania a zapisovania.

Pri použití iných rozhraní pre prístup k jednotlivým databázovým serverom bude, pravdepodobne, dosahovaný iný výkon. Keďže som od začiatku predpokladal, že samotný informačný systém budem písať v PHP, zvolil som tento jazyk aj pre overenie výkonu spojenia jednotlivých databázových systémov a serverov s PHP.

Oracle má v licenčnom ujednaní klauzulu, ktorá zakazuje publikovanie výkonnostných testov, ktoré by Oracle Database porovnávali s inými databázovým systémami. Preto pri Oracle Database neuvádzam konkrétne časy, aj keď je súčasťou porovnania. Pri IBM DB2 som podobné licenčné obmedzenie nenašiel, ale rozhodol som

sa neuvádzať presné čísla, ak by som toto obmedzenie prehliadol. Cieľom týchto testov bolo síce relatívne výkonnostné porovnanie rutinných operácii, pri ktorých sa predpokladá, že budú vo výslednej aplikácii prevažovať, ale je závislé na kvalite konektorov (rozhrania, ktorým sa k databázovému serveru aplikácia pripája).

Aj keď je rýchlosť podstatný faktor, zvolená databáza musí mať v prvom rade všetky vlastnosti, ktorých využitie v aplikácii je pravdepodobné a žiadúce, resp. musí mať minimum nedostatkov.

V tomto prípade sú veľmi žiadúce vlastnosti:

- podpora transakcií (transactions)
- podpora uložených procedúr (stored procedures).

O týchto a ostatných oceňovaných vlastnostiach budem písať pri porovnávaní kľúčových vlastností databázových systémov nižšie.

2.4.1 Výber údajov z databázy

Pri tomto teste bol som vyberal z testovaných databázových systémov sadu prvých 500 záznamov a prvých 50 záznamov. Toto množstvo sa nemusí v porovnaní s celkovým množstvom záznamov zdať veľké, ale bežný užívateľ si, pravdepodobne, bude na jednej strane s výsledkami prezerať len okolo 30 záznamov. Do medzipamäte môže byť skopírovaných ďalších približne dvadsať, aby sa urýchlilo posúvanie pri prezeraní si záznamov a užívateľ nemusel čakať, ak sa posunie len o pár riadkov. To však závisí od použitého postupu, ktorý bude zvolený pri návrhu konkrétnej aplikácie. Prezeranie si veľkého množstva údajov naraz sa predpokladá len pri prezeraní si záznamov, kde by mohlo byť potrebné pracovať s väčším množstvom údajov.

Vybrať posledných n záznamov je možné viacerými spôsobmi podľa toho, či majú byť údaje zoradené zostupne alebo vzostupne. Podľa toho sa aj zmení rýchlosť vrátenia výsledkov databázovým serverom.

2.4.1.1 Porovnanie výsledkov testov

Tabuľka 1: Porovnanie výkonu nameraného testovacím skriptom, počet databázových záznamov: 50.

Metóda pripájania sa k databázovému serveru	Priemerný čas vykonania skriptu [s]
mysql_pconnect	0,00193
mssql_pconnect	0,00395
mysqli_connect_innodb	0,00496
mysql_connect	0,00539
mysqli_connect	0,00707
firebird_pconnect	0,01155
db2_pconnect	viac ako 0,01155
postgre_pconnect	0,03770
mssql_connect	0,05500
firebird_connect	0,19860
mssql_odbc_pconnect	0,25090
oracle_pconnect	viac ako 0,25090
db2_connect	viac ako 0,25100
postgre_connect	1,65355
oracle_connect	viac ako 1,65355

Tabuľka 2: Porovnanie výkonu nameraného testovacím skriptom, počet databázových záznamov: 500.

Metóda pripájania sa k databázovému serveru	Priemerný čas vykonania skriptu [s]
mysql_pconnect	0,01133
mysql_connect	0,01375
mysqli_connect_innodb	0,01752
mysqli_connect	0,02020
mssql_pconnect	0,02695
mssql_connect	0,03659
firebird_pconnect	0,04131
db2_pconnect	viac ako 0,04131
postgre_pconnect	0,06654
firebird_connect	0,38185
oracle_pconnect	viac ako 0,38185
db2_connect	viac ako 0,52310
postgre_connect	1,90306
oracle_connect	viac ako 1,90306
mssql_odbc_pconnect	6,24735

2.4.2 Zápis údajov do databázy a ich následný výber

Tento test zahŕňa zápis 30 záznamov do databázy a následný výber posledných 30 záznamov z databázy. Vzhľadom na to, že `ab` používa viacero súčasných požiadaviek, je táto úloha do istej miery paralelizovaná a lepšie odráža skutočný scenár, ku ktorému môže dôjsť.

Väčšie množstvo údajov zapisované naraz dávkovým spôsobom sa predpokladá najmä pri príjme a výdaji palív a niektorých ďalších operáciách.

Pri tejto metóde som tiež vyskúšal efekt prepared statements na rýchlosť vkladania údajov do databázy a vo väčšine prípadov je badateľné isté zlepšenie, aj keď pomerne malé a prakticky zanedbateľné. Rozdiel bol vo väčšine prípadov menší ako šesť desatín sekundy, predovšetkým v prípade rýchlejšie vykonávaných skriptov. Spomínaný malý rozdiel je dostatočné malý na to, aby sa o ňom dalo uvažovať ako o chybe merania. Vyskytli sa tu dve výnimky, Firebird a IBM DB2. V prípade Firebird došlo k výraznému zvýšeniu rýchlosti vykonania skriptov. IBM DB2 mala pri všetkých testoch nie veľmi stále výkonnostné charakteristiky. Nepodarilo sa mi ale zistiť, čo ich spôsobuje, keďže postup pri testovaní som nemenil a bol rovnaký u všetkých databázových systémov. Je možné, že kvalita PHP konektora pre DB2 nie je taká dobrá ako pre ostatné systémy. Stávalo sa, že počas testovania DB2 bol server zahltený požiadavkami a po dosiahnutí maximálneho počtu pripojení ďalšie neprijímal. Vtedy `ab` ukončil testovaciu sériu. Nastal aj stav, v ktorom nastal pri testovaní DB2 pád servera Apache, čo sa mi pri testovaní ostatných systémov nestalo.

Zápis aj čítanie údajov operoval na tabuľke `writetests`, ktorá je súčasťou databázy `testovacia_velka`, ktorá je na DVD prílohe k diplomovej práci.

2.4.2.1 Porovnanie výsledkov testov

Tabuľka 3: Porovnanie výkonu nameraného testovacím skriptom pri použití prepared statements a bez nich

Metóda pripájania sa k databázovému serveru	Bez použitia prepared statements [s]	S použitím prepared statements [s]	Rozdiel [s]	Rozdiel [%]
mysql_pconnect	0,03770	0,03445	0,00325	0,33
mysql_connect	0,05643	0,05234	0,00409	0,41
mysqli_connect	0,09737	0,03466	0,06272	6,27
mssql_pconnect	0,33144	0,29509	0,03635	3,63
mssql_connect	0,51392	0,45587	0,05805	5,81
oracle_pconnect	viac ako 0,51392	viac ako 0,45587	-	-
postgre_pconnect	1,12474	1,06282	0,06192	6,19
mysqli_connect_innodb	2,44675	2,41839	0,02837	2,84
firebird_pconnect	2,74415	1,36393	1,38022	138,02
firebird_connect	3,77924	1,88253	1,89672	189,67
oracle_connect	viac ako 3,77924	viac ako 1,88253	-	-
postgre_connect	3,33647	3,18553	0,15093	15,09
db2_pconnect	viac ako 3,33647	viac ako 3,18553	-	-
db2_connect	viac ako 9,81861	viac ako 10,88532	-	-

2.4.3 Výber údajov po spájaní tabuliek

Táto operácia predpokladá využitie otázky s použitím `JOIN` a spájania pomocou stĺpca, ktorý je unikátnym indexom. Spájanie tabuliek za účelom získania údajov z oboch prepojených tabuliek je jednou z najbežnejších operácií pri práci s relačnými databázami. Pre tento prípad som vytvoril kópiu tabuľky `testdata` a nazval ju `testdata_join`. Test spočíval v jednoduchom spojení tabuliek `testdata` a `testdata_join` a následnom výbere 500 záznamov. V tomto prípade som počet záznamov zvolil veľký len preto, lebo predstavoval pre databázový systém aj server väčšiu záťaž ako menší, aj keď reálnejší počet.

2.4.3.1 Porovnanie výsledkov

Tabuľka 4: Porovnanie výkonu nameraného testovacím skriptom, počet databázových záznamov: 500.

Metóda pripájania sa k databázovému serveru	Priemerný čas vykonania skriptu [s]
mysql_pconnect	0,01141
mysql_connect	0,01785
mysqli_connect	0,02516
mysqli_connect_innodb	0,09729
mssql_pconnect	0,10980
firebird_pconnect	0,23241
mssql_connect	0,34157
postgre_pconnect	0,46328
oracle_connect	viac ako 0,46328
oracle_pconnect	viac ako 0,47903
db2_pconnect	viac ako 0,51090
firebird_connect	1,84039
postgre_connect	4,06163
db2_connect	viac ako 4,06163

2.5 Porovnanie kľúčových vlastností vybraných databázových systémov

Hneď na úvod tejto časti by som chcel poznamenať, že DB2 Express-C 9.5 má značne problematický inštalátor a hoci existujú oficiálne diskusné fóra IBM na Internete práve pre tento produkt, problém s inštalátorom som si musel vyriešiť sám a trvalo to viac ako týždeň. Poukazuje to na dôležitosť (ne)prítomnosti podpory zo strany výrobcu produktu. Riešenie sa nakoniec ukázalo byť pomerne jednoduché a popísal som ho na fórach IBM [5].

Uložené procedúry môžu byť pomerne jednoduchým a účinným prostriedkom na zvýšenie celkového výkonu databázového systému a/alebo zoskupovanie a zjednodušovanie niektorých procedúr, ktoré by bolo inak potrebné vykonávať v aplikačnej vrstve.

Transakcie umožňujú zabezpečiť vykonanie buď všetkých príkazov v tele transakcie, alebo žiadnych v nej obsiahnutých. Predstavujú tak prostriedok na ochranu logickej konzistencie údajov pre prípad výpadku napájania alebo poruchy.

Cenená je tiež možnosť získať nástroje na správu databázy zdarma, keďže tento informačný systém budú pravdepodobne používať zákazníci Intech Control citliví na cenu riešenia.

Z dôvodu uvedeného vyššie boli vybrané len databázy, ktoré majú verziu poskytovanú zdarma. Zároveň majú ale všetky možnosť platenej podpory alebo upgrade na verziu s podporou.

ACID je skratka pre atomicity (atomicita), consistency (konzistencia), isolation (izolácia), durability (trvanlivosť). V tomto kontexte je atomicita schopnosť databázového systému vykonať buď úlohy zadané v transakcii, alebo žiadnu z nich. Konzistencia je vlastnosť, ktorá zaručuje, že databáza je v konzistentom stave pred transakciou a po jej ukončení, t. j. správa sa rovnako pred transakciou a po jej ukončení. Izolácia je schopnosť systému vykonať operácie v transakcii tak, že sa zdajú izolované od ostatných operácií. Trvanlivosť zaručuje nevratnosť transakcie potom, ako databázový systém oznámil užívateľovi, že sa transakcia skončila [1].

V tabuľke 5 je pripravený krátky prehľad požadovaných vlastností a ich splnenie porovnávanými databázovými systémami.

Tabuľka 5: Stručné porovnanie požadovaných vlastností vybraných databázových systémov						
	MySQL	**Postgre**	**Firebird**	**MS SQL**	**Oracle 10g**	**IBM DB2**
Verzia zdarma	✓	✓	✓	✓	✓	✓
Testovaná verzia	5.1.23	8.3.1	2.1RC2, Superserver	Express 2005	10g Express	Express-C 9.5
Platená podpora*	✓	✓	✓	✓	✓	✓
Transakcie	✓	✓	✓	✓	✓	✓
Uložené procedúry	✓	✓	✓	✓	✓	✓
Podpora ACID	✓	✓	✓	✓	✓	✓
Licencia	Duálna: GNU GPL / Komerčná	BSD	InterBase Public License	Uzavretá	Uzavretá	Uzavretá
Full textové vyhľadávanie**	✓	✓	×****	✓	✓	×****
Nástroj na správu databáz zdarma	✓	✓	✓	✓	✓	✓
Použiteľnosť nástrojov dodávaných s databázovým systémom***	N/A (3)	4	N/A (3)	5	4	3
Obmedzenia verzie zdarma	žiadne	žiadne	žiadne	Využíva max. 1 procesor, 1 GB RAM. Max. veľkosť databázy je 4 GB	Jedna databáza (množstvo schém nie je obmedzené), max. 4 GB údajov, využíva do 1 GB RAM a jeden procesor	Využíva max. dve jadrá CPU a 2 GB RAM
Replikácia	✓	✓	×****	✓*****	✓******	×
Max. veľkosť databázy*******	neobmedzená	neobmedzená	> 30 TB	524258 TB	neobmedzená (4 GB)	512 TB
Max. veľkosť tabuľky*******	2 GB - 16 TB	32 TB	37 GB	524258 TB	4 GB násobená veľkosť bloku	512 TB

* zahŕňa aj možnosť prechodu na vyššiu verziu, ak pre verziu zdarma nie je poskytovaná podpora výrobcom

** full textové vyhľadávanie je možné riešiť aj inštaláciou ďalších programov, ktoré by databázu indexovali pre full textové vyhľadávanie, ale v rámci snahy minimalizovať počet programov na inštaláciu a údržbu získa kladné hodnotenie databázový systém, ktorý má podporu full textového vyhľadávania

*** max. hodnotenie je 5, minimálne 1. Ak sa s databázovým systémom nedistribuuje nástroj na správu, je v zátvorke hodnotenie voľne dostupného nástroja

**** na zabezpečenie tejto funkcie sú dostupné externé nástroje

***** MS SQL Express 2005 vie len prijímať údaje (je Subscriber) zo servera, ktorý slúži ako Publisher, aj to nie v plnom rozsahu

****** len základná, obmedzené možnosti

******* uvádzaná max. veľkosť neobmedzenej verzie, obmedzenie verzie zdarma je v zátvorke

2.6 Konečné rozhodnutie

Na základe vykonaných testov a zadaných požiadaviek som sa rozhodol odporúčať Postgre SQL, pretože má v aktuálnej verzii podľa [1] najviac vlastností využiteľných pri dotazovaní sa v jazyku SQL a podľa vykonaných testov má vo väčšine situácií prijateľný výkon. Ten pod operačným systémom Windows nie je taký vysoký ako napríklad pod OS Linux alebo FreeBSD. Môže to byť spôsobené aj skutočnosťou, že pre operačný systém Windows tento databázový systém nie je vyvíjaný tak dlho ako pre Linux alebo FreeBSD. Pre OS Windows je dostupný od verzie 8.0.

MySQL pri použití InnoDB ako storage engine bol odporúčaný, ak by mala rýchlosť o niečo vyššiu prioritu ako vlastnosti, ktoré musí mať vybraný databázový systém.

Nakoniec bol zadávateľom vybraný databázový systém Microsoft SQL Server 2005 Express Edition, ktorý som odporúčal ako tretí. Z vykonaných testov síce nevyšiel najlepšie, ale po konzultácii s Ing. Balážom bol vybraný práve SQL Server od Microsoftu, keďže je vo firme Intech Control používaný a zatiaľ sa im osvedčil ako dostatočne robustné riešenie pre menšie aplikácie.

3 Výber jazyka

Výber programovacieho jazyka, ktorý sa použije pri tvorbe prakticky ľubovoľného programu alebo systému máva rozhodujúci vplyv na niektoré vlastnosti výsledného produktu a tiež na rýchlosť vývoja daného riešenia.

3.1 Zvažované jazyky a konečný výber

Podobne ako pri databázových systémoch, aj tu bol urobený menší rozbor vhodnosti jednotlivých jazykov. Najprv bolo zvažované použitie webového rozhrania pre používanie informačného systému. Kandidujúcimi jazykmi boli PHP, Perl, Python a Java.

Java má dobrú povesť pri použití vo firemnom prostredí ako stabilný a pomerne výkonný jazyk.

Perl a Python sú síce obvykle pomalšie pri porovnávaní rýchlosti vykonania podobných úloh, ale pri webovom programovaní sú používané a možnosti pripojenia na rôzne databázové systémy sú dobré.

PHP je jazyk, ktorý bol vytvorený práve na použitie pri tvorbe webových sídiel. Patrí medzi najjednoduchšie skriptovacie jazyky. Možnosti a jednoduchosť pripojenia sa k databázovým serverom sú veľmi dobré a pri použití operačného systému Windows lepšie ako pri Pythone a Perle (sú skôr aktualizované konektory a sú oficiálne podporované projektom PHP pre všetky testované databázy). Naviac mám s jazykom PHP isté skúsenosti, takže by mi vývoj v ňom išiel trocha rýchlejšie ako v ostatných, vyššie uvedených jazykoch.

Po ukončení časti 2 tejto práce bol znova zvážený predpokladaný počet používateľov tohto informačného systému, ktorý by mal byť do 5 používateľov. Potom bol firmou Intech Control zvolený jazyk Visual Basic .NET. Argumenty pre jeho zvolenie namiesto C# na platforme .NET, Javy, C++, Perlu a Pythonu boli:

- Visual Basic sa už vo firme Intech Control používa
- výhodou oproti jazykom Perl a Python je rýchlosť
- oproti Jave, Perlu a Pythone je výhodou aj podpora v rámci MS Windows
- oproti ostatným jazykom je to rýchlosť vývoja aplikácií s grafickým používateľským rozhraním (GUI, Graphical User Interface) pri použití Microsoft Visual Studio 2008.

4 SQL a Visual Basic .NET

Jazyk SQL je jazykom štandardizovaným podľa noriem ANSI a ISO. Posledná verzia štandardu tohto jazyka je SQL:2006 z roku 2006. Od verzie SQL:2003 z roku 2003 obsahuje jazyk SQL podporu funkcií databáz pre prácu s XML, pričom SQL:2006 povoľuje použitie jazyka XQuery konzorcia W3C (World Wide Web Consortium) [1].

Reálne implementácie štandardov SQL do databázových systémov sú však takmer v každom z nich iné. Naviac sú takmer vždy obohatené vlastnosťami (funkciami a pod.) naviac. Niektoré z týchto funkcií môžu byť neskôr štandardizované, ale nie je to pravidlom.

Pre Visual Basic je natívnou platformou operačný systém Microsoft Windows, keďže ide o jazyk, ktorý bol touto spoločnosťou vyvinutý. Posledná verzia tohto jazyka je Visual Basic 6 z roku 1998 [1].

Visual Basic .NET je nástupcom jazyka Visual Basic a je jeho implementáciou na platforme .NET. Súčasná verzia, Visual Basic 2008, je v poradí deviatou oficiálne vydanou verziou jazyka Visual Basic a štvrtou verziou jazyka Visual Basic .NET [1].

Microsoft .NET Framework je v podstate zbierkou tried a funkcií v jazykoch ním podporovaných. Zahŕňa funkcie a triedy pre tvorbu užívateľského rozhrania, webových aplikácií, prácu s databázami, kryptografiou, sieťovou komunikáciou, numerickými algoritmami a pre prístup k údajom.

4.1 Ukážky použitia jazyka SQL

Výber údajov

Výber údajov je v jazyku SQL realizovaný príkazom `SELECT`, s ktorým sa dá používať viacero parametrov, ako sú napríklad `JOIN`, klauzula `WHERE` a podobne. Príkazy sú končené bodkočiarkou, je teda možné ich rozdeliť na viacero riadkov.

Príklady použitia príkazu `SELECT`:

Výber údajov zo všetkých stĺpcov (`*`) v prvých 100 riadkoch z databázovej tabuľky `testdata`, zoradený podľa stĺpca nazvaného `col1`:

```
SELECT TOP 100 * FROM testdata ORDER BY col1;
```

Výber údajov zo všetkých stĺpcov (`*`) v prvých 100 riadkoch z databázovej tabuľky `writetests`, zoradený podľa stĺpca nazvaného `col1` zostupne. Klauzula `WHERE` ohraničuje počet vracaných riadkov na tie, ktoré majú hodnotu v stĺpci `col1` väčšiu, ako počet riadkov v tabuľke `writetests` (získaný pomocou tzv. subquery, podpríkazu) zmenšený o 100:

```
SELECT TOP 100 * FROM writetests
WHERE col1 >= ((SELECT count(*) FROM writetests ) - 100 )
ORDER BY col1 ASC;
```

Ak potrebujeme spojiť údaje z viacerých tabuliek, dá sa to spraviť príkazom `JOIN`, ktorý spája tabuľky podľa zadaného kritéria. Kritérium, podľa ktorého má dôjsť ku spojeniu je možné udať v parametri `ON` nasledovaným kritériom v okrúhlych zátvorkách (pr.: `ON (t1.col1=t2.col1)`) alebo ako súčasť klauzuly `WHERE` (pr.: `WHERE t1.col1=t2.col1`).

```
SELECT TOP 500 t1.col1, t1.col2, t1.col3, t1.col4, t1.col5,
t1.col6, t2.col7, t2.col8, t2.col9, t2.col10, t2.col11 FROM
testdata t1 JOIN testdata_join t2 ON (t1.col1=t2.col1) WHERE
t1.col1>=((select count(*) from testdata) - 499) ORDER BY col1
```

Ak potrebujeme spojiť údaje z príkazov `SELECT`, je možné použiť príkaz `UNION` (nevracia prázdne riadky) a `UNION ALL` (vracia aj prázdne riadky s hodnotou `NULL`).

```
SELECT AVG(mnozstvo) FROM prijmy_do_skladu
UNION ALL
SELECT AVG(o.mnozstvo_gross) FROM odbery o
UNION ALL
SELECT MIN(o.mnozstvo_gross) FROM odbery o
UNION ALL
SELECT MAX(o.mnozstvo_gross) FROM odbery o
UNION ALL
SELECT AVG(o.mnozstvo_net) FROM odbery o
UNION ALL
SELECT MIN(o.mnozstvo_net) FROM odbery o
```

```
UNION ALL
SELECT MAX(o.mnozstvo_net) FROM odbery o
UNION ALL
SELECT count(id) FROM zakaznici
UNION ALL
SELECT count(id) FROM dodavatelia;
```

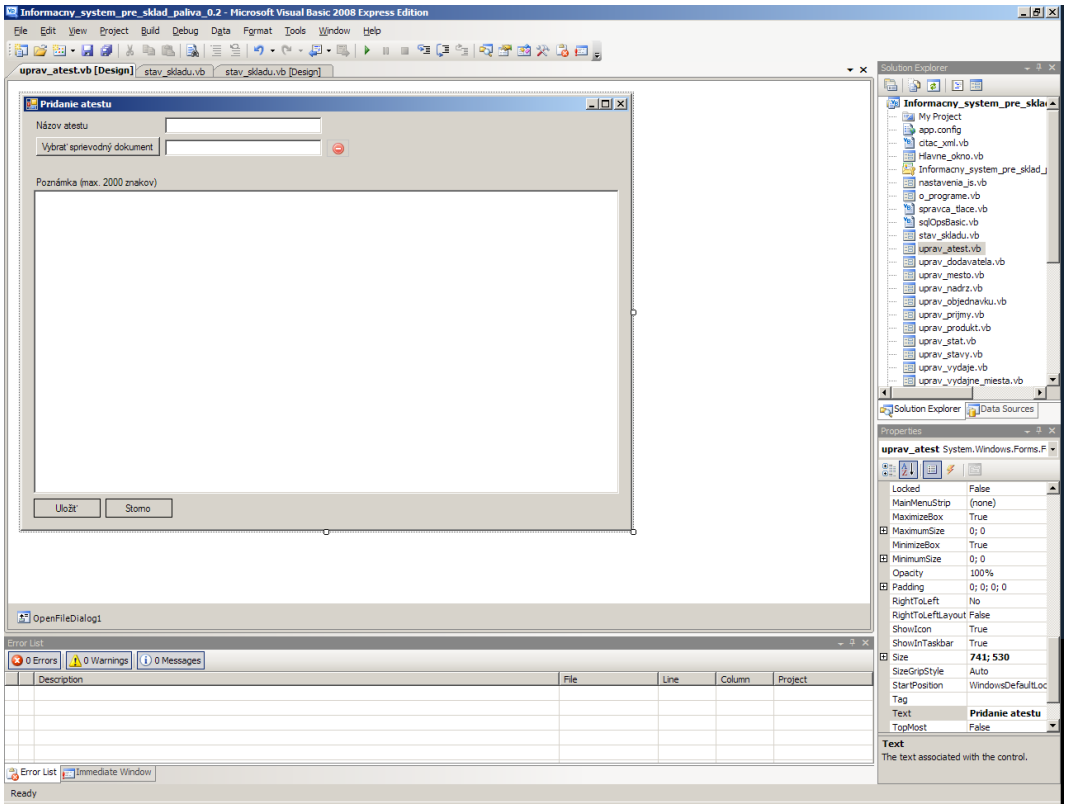
Funkcia `AVG(mnozstvo)` vypočíta priemernú hodnotu z hodnôt v stĺpci množstvo, `MIN(mnozstvo)` nájde minimálnu hodnotu v danom stĺpci, `MAX(mnozstvo)` nájde maximálnu hodnotu v danom stĺpci a `SELECT COUNT(id) FROM zakaznici` zistí počet riadkov v danej tabuľke. Namiesto `COUNT(id)` sa dá použiť (a obvykle sa tak aj robí) `COUNT(*)`.

4.2 Ukážky použitia jazyka Visual Basic .NET

Vývoj programov vo Visual Basic .NET zvykne prebiehať podobným spôsobom:

1. programátor otvorí vývojové prostredie - pravdepodobne nástroj pre vývoj vo Visual Basic .NET od spoločnosti Microsoft ako je napríklad na obr. 4.1
2. pridá do projektu niekoľko nových položiek ako sú napríklad triedy a formuláre
3. pridá ovládacie prvky pre vytvorené formuláre
4. ošetrí kódom všetky časti, u ktorých je to potrebné (triedy, akcie vykonávané pri stlačení tlačidla a podobne)

Postupnosť krokov, okrem prvého, sa, samozrejme, môže líšiť, ale takmer isto bude postup pri vytváraní programu obsahovať vyššie uvedené kroky.



Obr. 4.1: Ukážka Microsoft Visual Basic 2008 Express

Keďže niekedy môže byť problematické nájsť v tomto jazyku jednoduché a funkčné ukážky, niekoľko ich uvediem na príklade, ktorým je naplnenie mriežky, ako je vidno napr. na obr. 5.3, údajmi z databázovej tabuľky.

Pre tento úkon, keďže je pomerne častý, som si vytvoril triedu pre vykonávanie databázových operácií.

```
Imports System.IO
Imports System.Data.SqlClient
Imports System.Data
Imports System.Xml
Imports System.Configuration
Imports System.Collections.Specialized
Imports System.Reflection

Public Class sqlOpsBasic
      Public konfiguracne_hodnoty As NameValueCollection
      Private pripojenie As SqlConnection
```

```
    Private td_DataAdapter As New SqlDataAdapter()
    Private DataSet As New DataSet()
    Private connection_string As String

    Sub New()
        Try
                Dim citaj_xml As New citac_xml()
                konfiguracne_hodnoty = citaj_xml.nacitaj_konfiguraciu()
                connection_string = "Server=" +
konfiguracne_hodnoty("server") + "; Database=" +
konfiguracne_hodnoty("databaza") + "; user id=" +
konfiguracne_hodnoty("uzivatel") + "; password=" + konfiguracne_hodnoty("heslo")
        Catch ex As Exception
                If konfiguracne_hodnoty Is Nothing Then
                        MessageBox.Show("Nepodarilo sa načítať konfiguračný
súbor. Nie je možné pracovať s databázou. Odporúčame ukončiť program, vytvoriť
konfiguračný súbor podľa manuálu a spustiť program znova. Chyba: " + ex.Message,
"Chyba")
                End If
        End Try
    End Sub

    Public Function getData(ByVal Query As String, ByVal caller As Object)
        pripojenie = New SqlConnection(connection_string)
        If (Query Is Nothing) Then
                Query = "SELECT TOP 100 * FROM zakaznici ORDER BY id"
        End If
        td_DataAdapter.SelectCommand = New SqlCommand()
        td_DataAdapter.SelectCommand.Connection = pripojenie
        td_DataAdapter.SelectCommand.CommandText = Query
        Try
                pripojenie.Open()
                Try
                        td_DataAdapter.Fill(caller)  ' caller je DataSet
                Catch ex As Exception
                        MessageBox.Show("Zlyhalo napĺňanie datasetu údajmi.
" & ex.Message, My.Resources.messagebox_error)
                        Return False
                End Try
```

```
                pripojenie.Close()
        Catch ex As Exception
                ' V prípade chyby zatvor spojenie.
                If pripojenie Is Nothing Then
                        pripojenie.Dispose()
                        pripojenie = Nothing
                End If
                        MessageBox.Show("Chyba pri pripájaní sa k databáze.
" & ex.Message, My.Resources.messagebox_error)
                        Return True
        End Try
        Return True
    End Function
End Class
```

Stručné vysvetlenie niektorých prvkov kódu uvedeného vyššie.

Príkaz `Imports` vraví kompilátoru, ktoré súčasti .NET Framework bude daný formulár/trieda/iný prvok potrebovať na to, aby fungovala a programátor nemusel znova vytvárať to, čo už bolo pre jeho/jej pohodlie naprogramované.

`Public Class sqlOpsBasic` začína definíciu triedy, podobne ako `Public Function getData()` začína definíciu funkcie. `End Class`, resp. `End Function` tieto definície ukončuje.

`Sub New()` je tzv. konštruktor, teda metóda, ktorá sa spúšťa pri vytvorení inštancie objektu. Je v nej umiestnený kód, ktorý používa objekt `citaj_xml`, ktorý je inštanciou triedy `citac_xml`. Pred použitím je potrebné ho deklarovať podobne: `Dim citaj_xml As New citac_xml()`, kde `Dim` vraví kompilátoru, že sa začala deklarácia novej premennej alebo objektu v danej funkcii/metóde.

Ak chceme deklarovať premennú, ktorá má byť dostupná pre celú triedu a jej metódy a funkcie a je možné ju volať len v rámci tejto triedy, potom sa dá použiť deklarácia `Private td_DataAdapter As New SqlDataAdapter()`, čím sme si deklarovali inštanciu triedy `SqlDataAdapter`, prostredníctvom ktorej sa budú posielať databáze požiadavky.

Ak má byť premenná/funkcia/metóda/objekt dostupná pre použitie mimo materskej triedy, potom musí byť namiesto `Private` použitá deklarácia typu `Public`.

Reťazce, datatyp `String`, je možné spájať pomocou znamienka `+`, ako vidno pri napĺňaní premennej `connection_string`.

Podmienka `If <výraz> Then … End` má funkčnosť ekvivalentu `if(<výraz>) {…;}` v jazyku C/C++.

"Užívateľské" ošetrenie výnimiek sa vykonávka v bloku

```
Try
   ...
Catch ex As Exception
   ...
End Try
```

kde ex je objekt, ktorý sa naplní chybovými údajmi. ex.Message napríklad vypíše popis chyby, ktorá nastala. Objekt MessageBox predstavuje oznamovací dialóg

```
MessageBox.Show("Zlyhalo napĺňanie datasetu údajmi. " & ex.Message, My.Resources.messagebox_error)
```

Ak by sme potrebovali naplniť získanými údajmi v DataSete, čo je objekt, ktorý obsahuje tabuľky vrátené SQL príkazom, napr. objekt DataGridView, ktorý predstavuje mriežku, ako je napr. na obr. 5.3, potom by sme to mohli spraviť takto

```
vykonajSQL.getData(Query, Me.DataSet)
      Dim tabulky As DataTableCollection = DataSet.Tables
         Try
                  Dim zobrazenie As New DataView(tabulky(0))
                  uzivateliaBindingSource.DataSource = zobrazenie
                  UdajeDataGridView.DataSource = uzivateliaBindingSource
         ...
```

Objekt `BindingSource` umožňuje niektoré ďalšie operácie s údajmi `DataSetu`, ako napr. filtrovanie a iné.

Keďže kompletný popis jazyka Visual Basic .NET je obsiahly a popis možností, funkcií a tried .NET Framework je ešte obsiahlejší, pre ich podrobnú dokumentáciu sa, prosím, obráťte na MSDN (Microsoft Developer Network), čo je on-line dokumentácia dostupná na adrese http://msdn.microsoft.com/.

5 Informačný systém Palivo

Informačný systém bol pomenovaný Palivo, aj keď jeho komerčná verzia môže mať iný názov.

Požiadavky na funkčnosť sú v zásade:

1. Čítanie údajov z databázy
2. Tieto údaje je potrebné aj vkladať a upravovať
3. Tlač niektorých údajov, najmä informácií o odberoch a príjmoch v sklade

Celý informačný systém je jeden program, ale ak má mať užívateľ možnosť prezerať si sprievodné dokumenty k atestom, potom bude, pravdepodobne, potrebovať aj ďalšie programy ako napríklad čítač PDF dokumentov, čítač dokumentov vo formátoch používaných kancelárskymi balíkmi a podobne.

Rozhranie programu bolo vytvorené tak, aby bol možný rýchly prístup k položkám, o ktorých sa predpokladá, že budú používané najčastejšie.

Niektoré ukážky informačného systému v ďalších častiach neobsahujú vo formulároch text. Je to preto, že väčšina textu bola vymyslená a niekedy bola len náhodným reťazcom.

Prakticky všetky časti programu možno používať súčasne, teda je možné napríklad pridávať atest a zároveň mať otvorený jeden z adresárov.

5.1 Adresáre

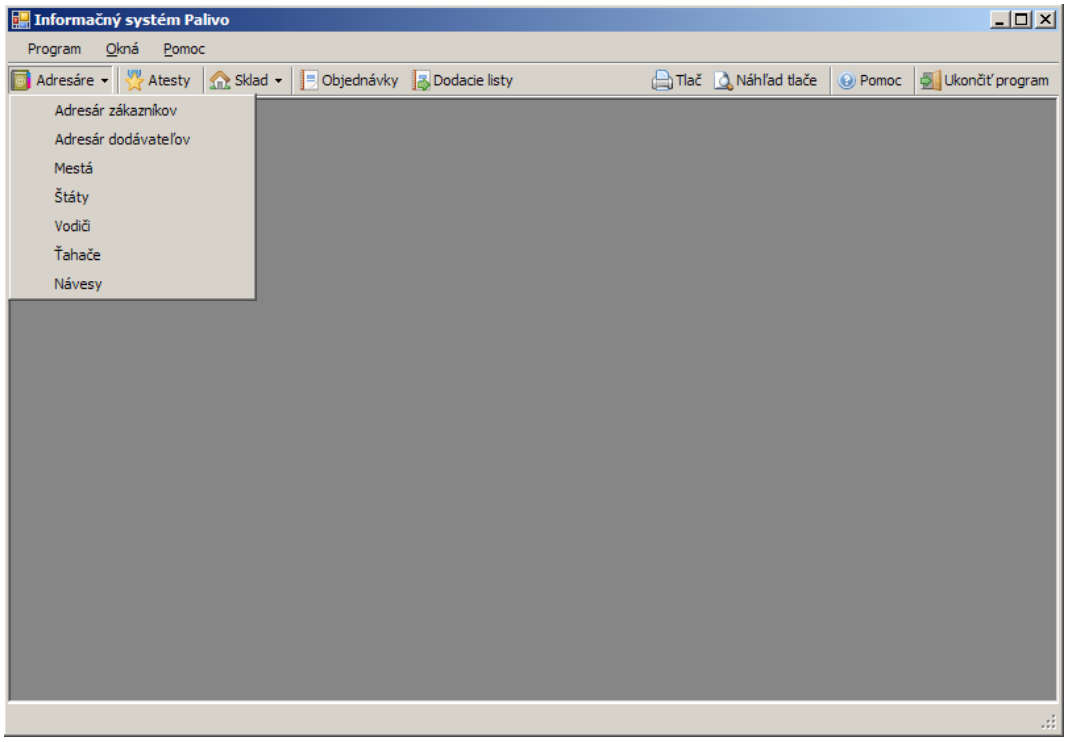
Adresáre sú jednou z prvých častí programu, s ktorými by sa mal užívateľ zoznámiť a ktoré by mal používať. Mať dopredu vyplnené údaje v adresári je veľmi výhodné, pretože údaje z adresárov sa používajú v kľúčových častiach systému ako sú objednávky a dodacie listy.

Vytvorené boli adresáre:

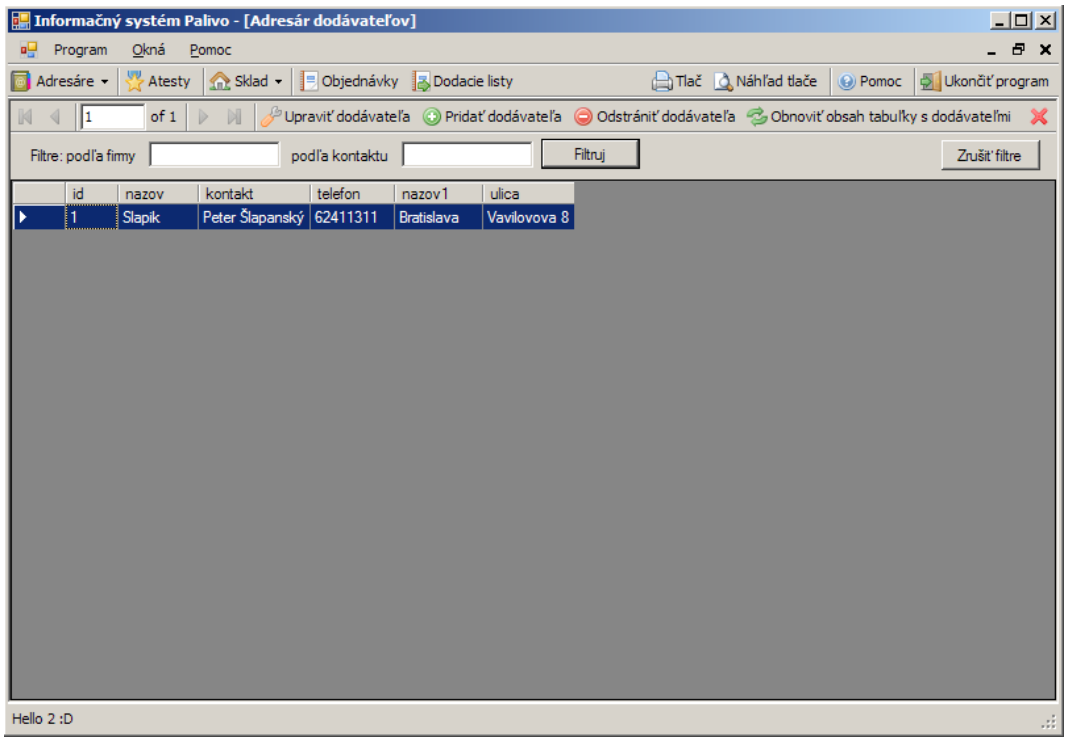
1. zákazníkov
2. dodávateľov
3. miest
4. štátov
5. vodičov

6. ťahačov
7. návesov

Žiadaný adresár si užívateľ systému môže vybrať z jednoduchej ponuky vo forme rozbaľovacieho sa zoznamu ako vidno na obr. 5.1.



Obr. 5.1: Ukážka rozbalenej ponuky adresárov



Obr. 5.2: Ukážka otvoreného adresára dodávateľov

Pri otvorení ktoréhokoľvek z adresárov sa zobrazí okno podobné oknu na obr. 5.2. Takéto usporiadanie bolo zvolené aj pre ďalšie časti systému, kde je potrebné zobrazovať väčšie množstvo položiek.

Samozrejme, sú zmenené texty tlačidiel a upravené možnosti filtrovania, ale vzhľadom sú inak takmer identické.

Filtrácia sa vykonáva vo všetkých zoznamoch rovnako. Po vyplnení políčka, podľa ktorého chceme údaje filtrovať (stačia počiatočné písmená) sa stlačením tlačidla filtruj spustí filtrácia, prípadne zmenia filtrovacie kritéria na novo zadané.

Manipulácia s položkami zoznamu sa realizuje pomocou tlačidiel Upraviť, Pridať a Odstrániť dodávateľa (resp. zákazníka, mesto atď.)

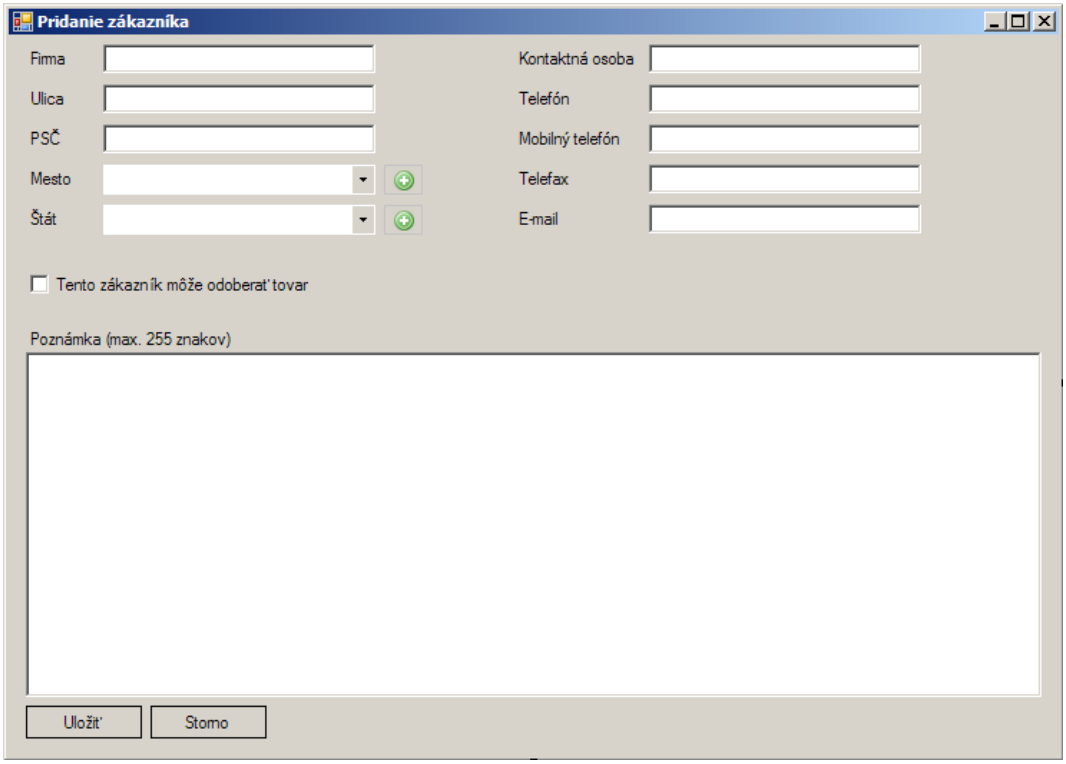
Ak by po úprave / pridaní záznamu nedošlo k obnove zoznamu (napr. dodávateľov), má užívateľ možnosť tabuľku záznamov obnoviť ručne tlačidlom Obnoviť obsah tabuľky s dodávateľmi (zákazníkmi, mestami, …).

Odstraňovanie záznamov je možné, len keď daný záznam nie je využívaný iným záznamom. Ak nie je možné jednoduché overenie použitia tohto záznamu, potom je odstraňovanie záznamov zakázané.

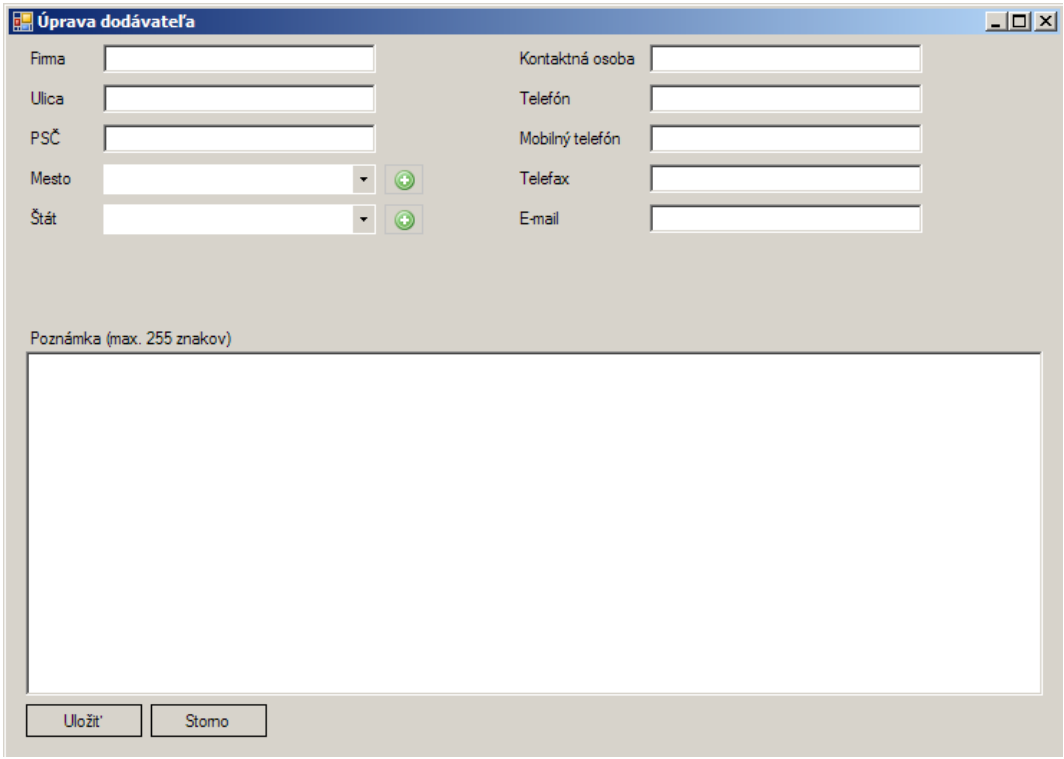
Pri pridávaní a úprave záznamu sa používa rovnaký formulár, v prípade úpravy

záznamu je vyplnený existujúcimi údajmi, ktoré môže užívateľ zmeniť a záznam opäť uložiť.

Nasledujú ukážky formulárov pre pridávanie a úpravu záznamov pre jednotlivé adresáre. Pre lepšiu prehľadnosť sú v ukážkach len formuláre bez rodičovského okna.

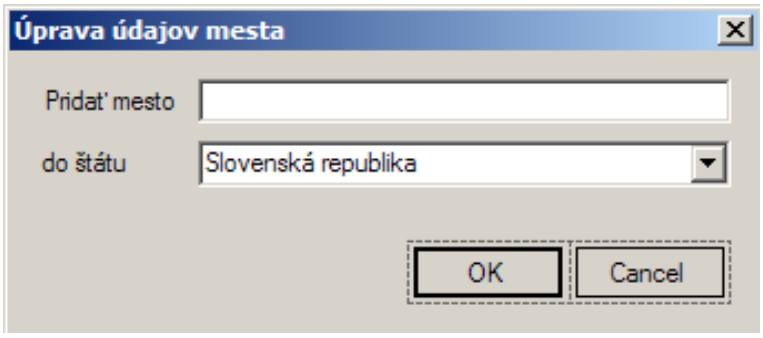


Obr. 5.3: Formulár na pridanie zákazníka

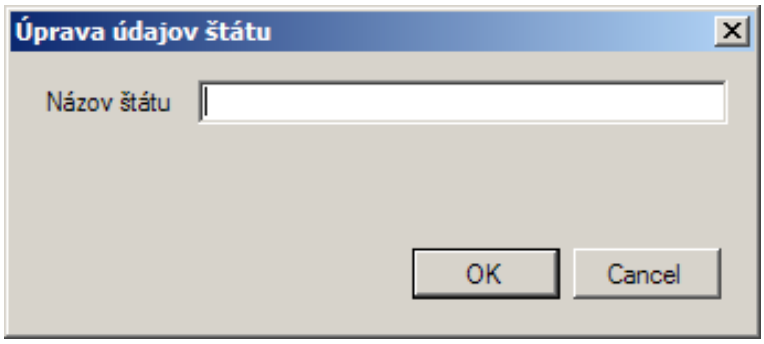


Obr. 5.4: Formulár na pridanie dodávateľa

Formulár na pridanie zákazníka a formulár na pridanie dodávateľa sú v momentálnej verzii IS Palivo dva samostatné formuláre, ktoré sa obsahom a vzhľadom líšia len málo. Rozdelenie bolo realizované preto, aby bolo možné pre zákazníkov so zvláštnymi potrebami oba formuláre nezávisle meniť a pridávať do nich potrebný obsah bez ohľadu na obsah druhého formulára.



Ukážka 5.5: Formulár na pridanie mesta

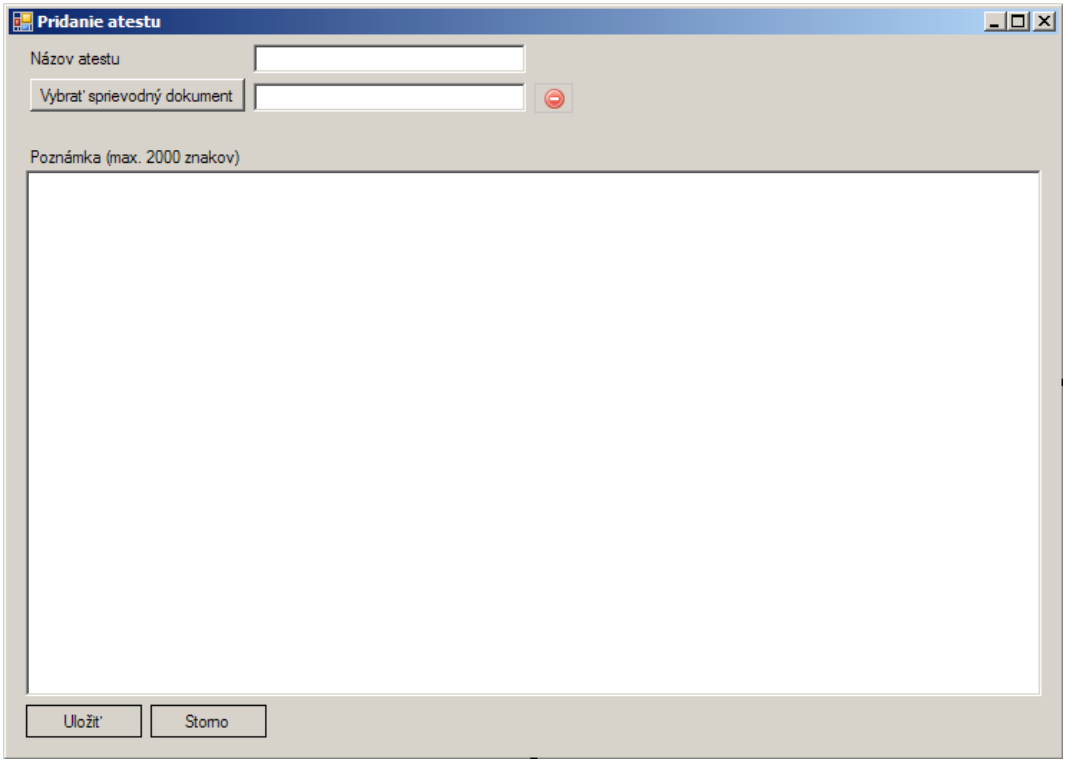


Obr. 5.6: Formulár na pridanie štátu

S formulármi na pridanie mesta a štátu do databázy je to podobné ako s predošlým párom formulárov. Tiež ide o takmer identické formuláre, z rovnakých dôvodov ako predtým však boli rozdelené.

5.2 Atesty

Atesty sú v tomto prípade predovšetkým dokladom o vlastnostiach daného produktu. Bolo pre ne vyčlenené zvláštne miesto pre prípad, že by do databázovej tabuľky s atestami boli pridávané aj atesty, ktoré nie sú len pre produkty, ktoré je možné prijímať alebo vydávať.



Obr. 5.7: Formulár pre pridanie atestu

Kliknutím na tlačidlo mínus v červenom kruhu sa označí aktuálne vybraná položka na odstránenie a po stlačení tlačidla Uložiť bude odstránená a nový dokument, ktorý bol predtým alebo potom vybraný, sa skopíruje do úložiska dokumentov.

Pri pridávaní atestov bolo potrebné vyriešiť najmä problém sprievodných dokumentov. Tieto dokumenty sú informáciou od vydavateľa atestu.

Otázny bol spôsob, akým majú byť tieto dokumenty uchovávané. V zásade boli dve možnosti ich ukladania:

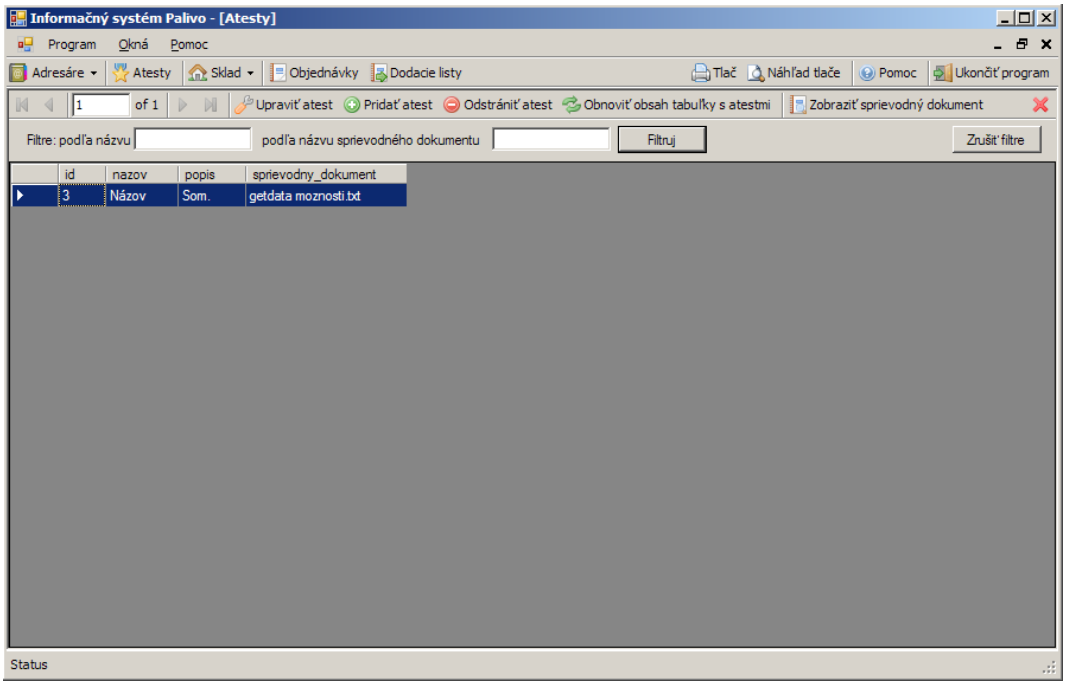
1. ukladať ich do databázy ako binárne údaje (napríklad ako typ BLOB – Binary Large Objects, veľké binárne údaje)
2. ukladať ich do prednastaveného priestoru na pevnom disku alebo na iné zariadenie, napr. SAN (Storage Area Network, vzdialené zariadenia, ktoré sú pripojené k serveru a zobrazujú sa užívateľovi ako lokálne) alebo NAS (Network-Attached Storage, v podstate ide o samostatný počítač, ktorého úlohou je ukladanie a poskytovanie súborov).

Zvolený bol spôsob druhý, teda ukladanie sprievodných dokumentov ako súborov do zvoleného umiestenia.

Hlavné dôvody pre toto rozhodnutie boli:

1. v čase tvorby systému sa predpokladal minimálny počet užívateľov informačného systému
2. pravdepodobne budú užívateľ, systém aj databázový server len na jednom počítači
3. rýchlosť prístupu údajov by teda mala byť plne dostačujúca aj pri použití súborového riešenia
4. riešenie s ukladaním súborov mimo databázy znižuje množstvo údajov v databáze, v ktorej sú namiesto binárnych objektov len názvy súborov sprievodných dokumentov
5. v týchto dokumentoch nie je potrebné vyhľadávať údaje, stačí ich zobrazovať.

Nevýhodou tohto riešenia je najmä potreba kontrolovať, či pri asociácii nového sprievodného dokumentu s novým/existujúcim atestom už v úložisku týchto dokumentov nie je súbor rovnakého názvu.

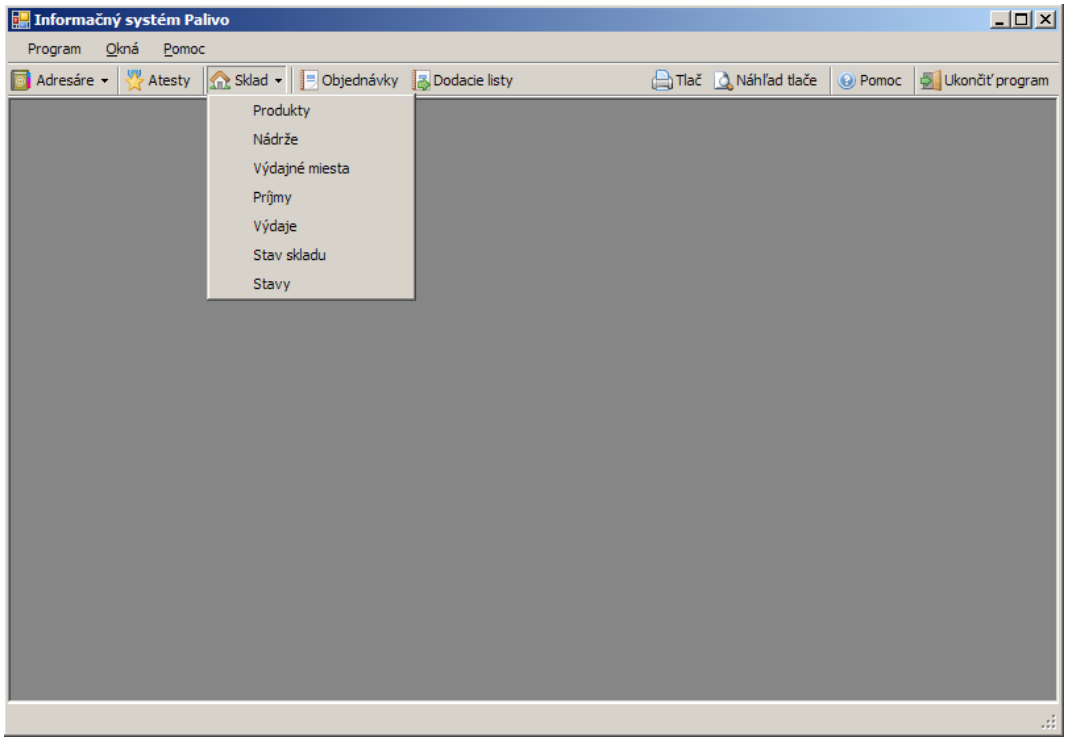


Obr. 5.8: Zobrazenie existujúcich atestov

Zobrazenie sprievodného dokumentu je možné aj zo zoznamu existujúcich atestov stlačením tlačidla Zobraziť sprievodný dokument.

5.3 Sklad

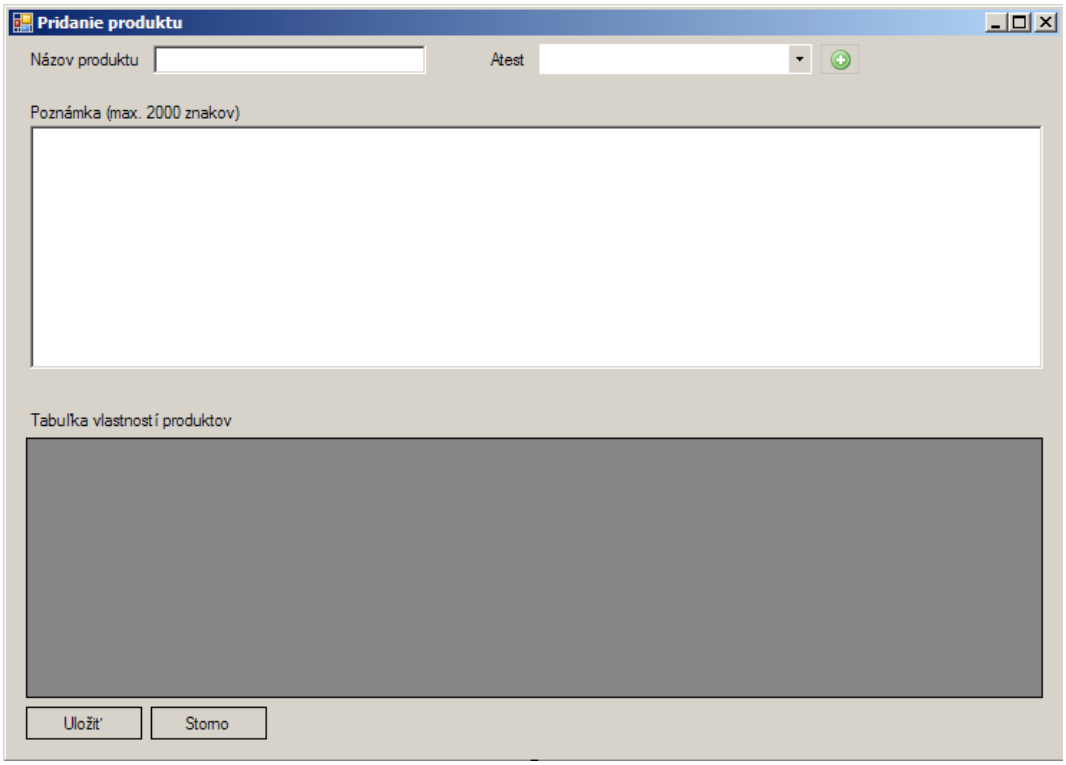
V položke Sklad sa vykonávajú všetky úkony, ktoré súvisia s príjmom alebo výdajom produktov.



Obr. 5.9: Ukážka ponuky Sklad

5.3.1 Produkty

Na tomto mieste sa zobrazujú produkty takým spôsobom, akým sa zobrazujú Atesty a zoznamy v Adresári. Iný je, samozrejme, formulár na pridávanie, resp. úpravu produktov.



Obr. 5.10: Formulár na pridanie nového produktu

Pre produkty bola zadaná požiadavka na možnosť pridávať vlastnosti produktu. Ako vyhovujúce riešenie sa ukázalo použitie dátového typu XML v Microsoft SQL Server 2005, aj keď XML dokumenty by mohli byť uložené aj pri použití textového dátového typu (napr. varchar). Vo verzii 0.2, ktorú popisuje táto diplomová práca, je pri upravovaní načítaný XML dokument z databázy a pri vytváraní produktu je vytvorený práve pred ukladaním údajov do databázy.

Pridávanie vlastností je jednoduché, stačí vpísať údaje do riadka, ktorý sa vytvorí zakaždým, keď je písané do predošlého alebo ak ešte žiadny nie je. Odstránenie vlastnosti spočíva len vo vybraní riadka s vlastnosťou a stlačení klávesy delete.

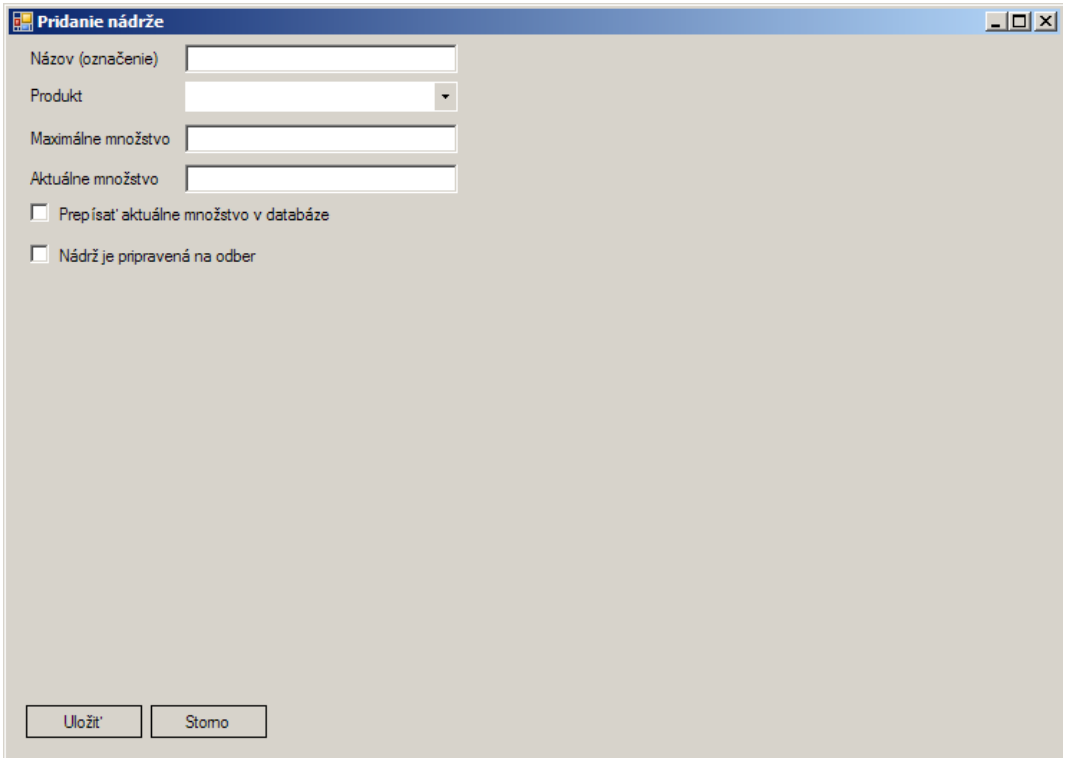
V prípade potreby je možné pridať nový atest aj z tohto okna použitím príslušného tlačidla (+ v zelenom kruhu).

5.3.2 Nádrže a výdajné miesta

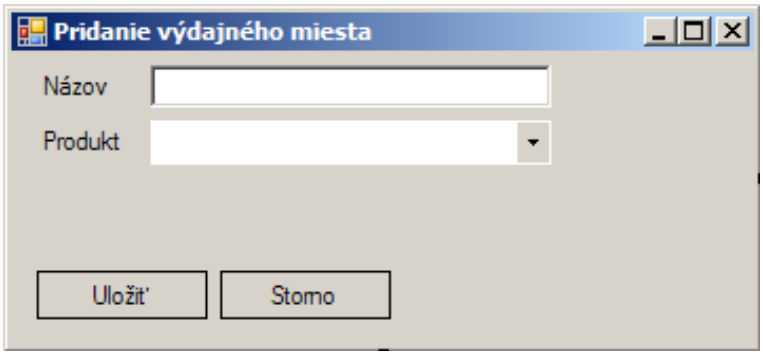
Výdajné miesta (alebo výdajné lávky) sú miesta, v ktorých sa realizuje odber. Palivo je držané v nádržiach. Tých môže byť na výdajnom mieste viacero.

Ich zobrazenie je tabuľkové a vzhľadom je rovnaké ako zobrazenie dodávateľov v adresári, resp. produktov v sklade.

Formuláre na pridávanie a úpravu záznamov sú zobrazené na obr. 5.11 a 5.12. Pripomínam, že na pridanie a na úpravu záznamu sa používa vždy ten istý formulár.



Obr. 5.11: Pridanie nádrže

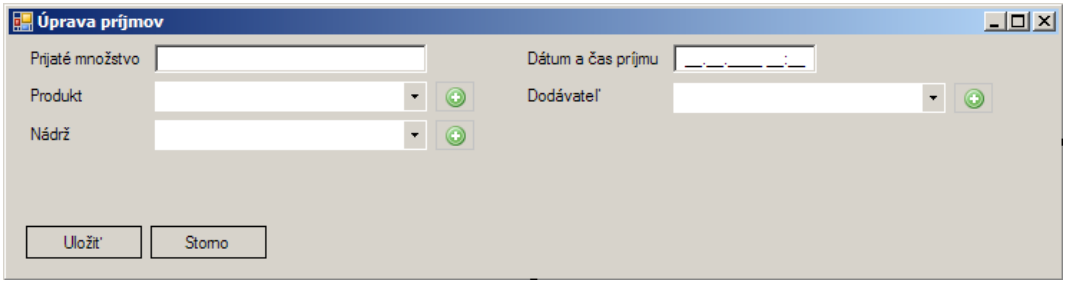


Obr. 5.12: Pridanie výdajného miesta

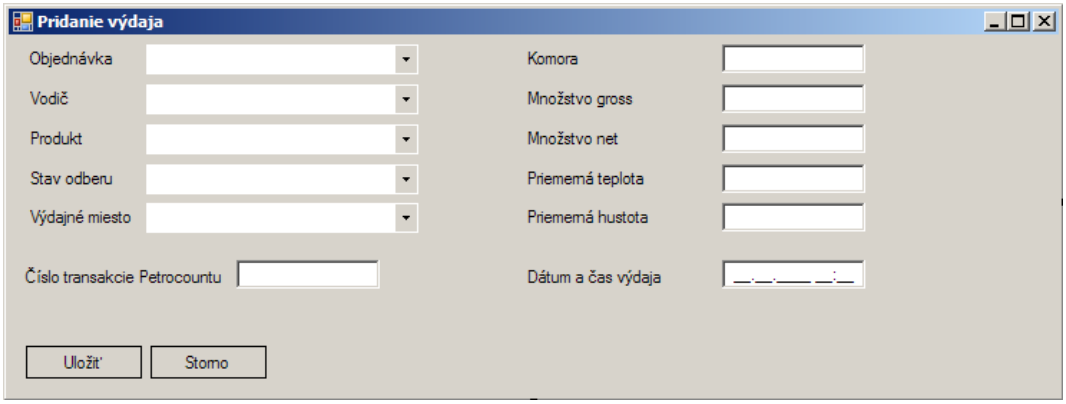
5.3.3 Príjmy a výdaje

Tieto údaje by nemali byť za bežných okolností menené a v informačnom systéme sú zavedené preto, aby bolo možné v prípade zlyhania nízko úrovňovej časti zadávať informácie o príjmoch a výdajoch ručne.

Nízko úrovňová časť je aplikácia, ktorá obhospodaruje fungovanie výdajných zariadení Petrocount, resp. Microload. Tieto zariadenia posielajú do riadiacej aplikácie informácie o odobranom množstve, fyzikálnych parametroch odoberaného paliva a informácie o odberateľovi. Riadiaca aplikácia ich zapisuje po prípadnom spracovaní do databázových tabuliek na príjmy a výdaje. Odtiaľ ich IS Palivo číta, zobrazuje a vo formulári Stav skladu aj vytvára jednoduché štatistiky.



Obr. 5.13: Formulár na úpravu príjmov



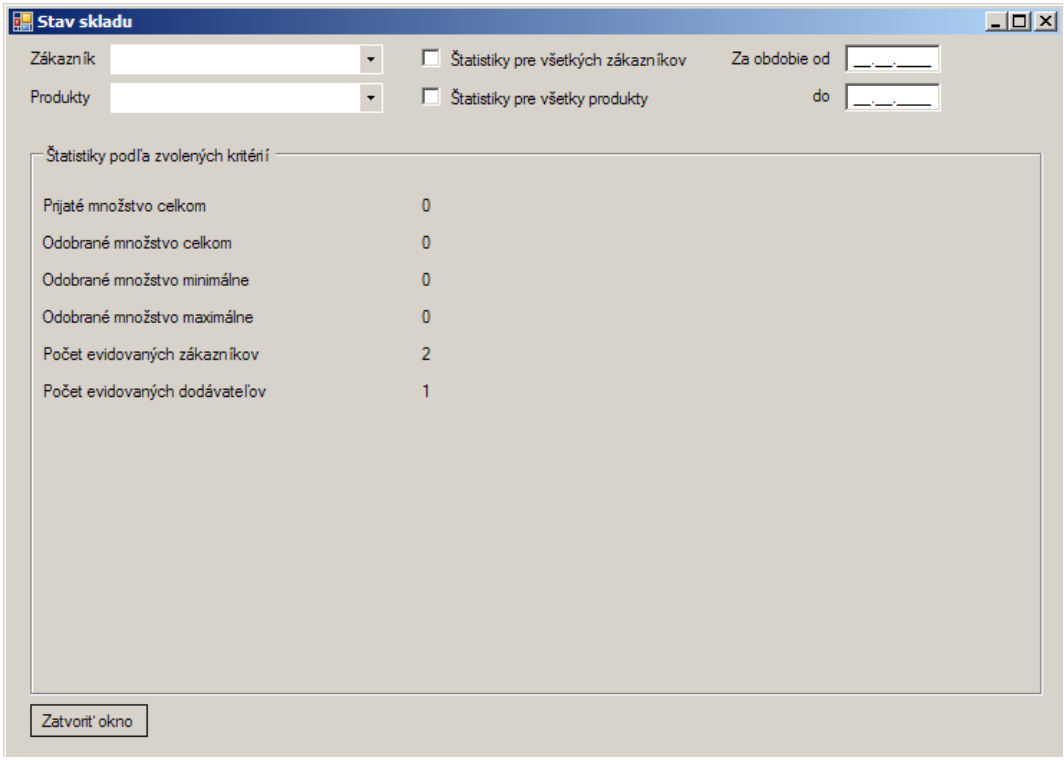
Obr. 5.14: Formulár na úpravu výdajov

Vo formulári na úpravu výdaja je pole Komora. Do tohto poľa je možné zadať viac ako jednu komoru, ak je to potrebné.

5.3.4 Stav skladu

Je to „miesto“ v programe, ktoré by mal užívateľ navštíviť, ak chce získať alebo vytlačiť si štatistiky, ktoré tento formulár zobrazuje.

Vo verzii 0.2 sú štatistiky len obmedzené a základné, ale v tomto formulári je dostatok miesta na pridávanie ďalších. V prípade potreby je tiež možné použiť objekt DataGridView, ktorý predstavuje tabuľkové zobrazenie, ako to vidno napríklad na obr. 5.8.

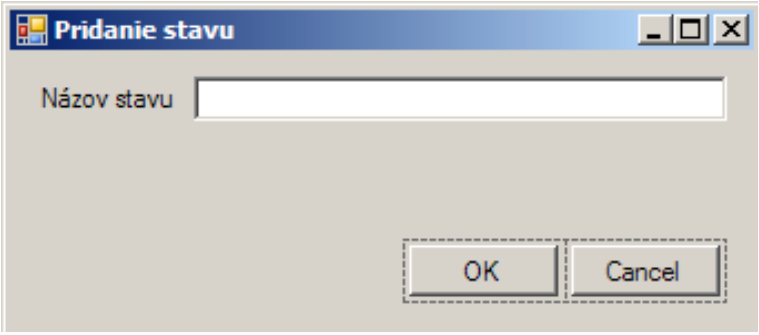


Obr. 5.15: Formulár so štatistikami

Štatistiky alebo skôr súhrny je možné generovať podľa zákazníkov a produktov za vybrané obdobie, predvolené obdobie nie je a generované sú štatistiky zo všetkých záznamov v databáze.

5.3.5 Stavy

Je tiež možné definovať stavy pre odbery, takže je možné ich opätovné použitie a nie je potrebné udržiavať texty so stavmi v tabuľke odberov. Tento postup je výhodný aj pre prípad, kedy by zákazník požadoval aplikáciu v nejakom inom jazyku ako v slovenčine.



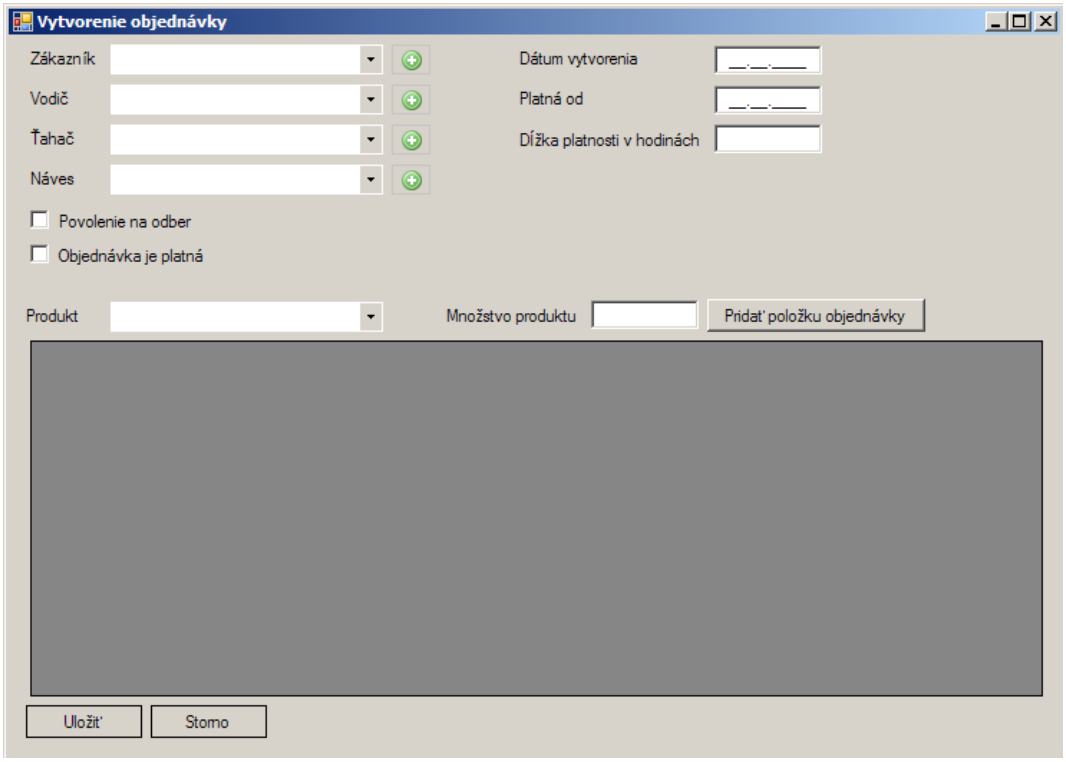
Obr. 5.16: Formulár na pridanie stavu

Formulár je veľmi jednoduchý a vo verzii 0.2 sa v ňom dá definovať len názov nového stavu.

5.3.6 Objednávky a dodacie listy

Objednávky sú rozhodujúcim prvkom IS Palivo.

Existencia objednávky je nevyhnutnou, ale nie postačujúcou podmienkou pre odber paliva.



Obr. 5.17: Formulár na vytvorenie objednávky

Ide v podstate o bežný formulár, ale je tu jeden rozdiel.

Podobne ako pri formulári pre produkty, aj tu je použitím objektu DataGridView zobrazený zoznam, ale položky sa dajú do zoznamu pridávať len vybraním Produktu z rozbaľovacej ponuky, zadaním Množstva produktu a stlačením tlačidla Pridať položku objednávky.

Dodacie listy nemajú samostatné formuláre. Sú generované z objednávok a odberov vybraním objednávky a stlačením tlačidla Dodacie listy. Obsahujú pôvodnú objednávku aj reálne odobrané množstvá, ktoré sú získavané práve z tabuľky odberov.

Objednávky aj dodacie listy majú číslo generované automaticky pri zavedení objednávky do databázy.

5.4 Tlačové zostavy a výstupy

Tlač v IS Palivo je veľmi jednoduchá a uskutočňuje sa z prehľadov, aký je napríklad na obr. 5.2. Výnimkou je tlač informácií o stave skladu, kde existuje len jeden formulár.

Požiadavky na tlač pre túto prácu neboli väčšie ako prinútiť program tlačiť na vybranú tlačiareň. Preto sú tlačové formuláre značne nevzhľadné a predpokladá sa, že budú upravované podľa požiadaviek jednotlivých zákazníkov do nimi požadovanej formy.

Možnosti tlače sú štandardné, v tabuľkových zobrazeniach stačí vybrať riadok s údajmi, ktoré chce užívateľ vytlačiť a stlačiť tlačidlo pre tlač alebo si pred tlačou pozrieť náhľad (Print Preview).

6 Záver

Táto práca bola moja prvá reálna skúsenosť s tvorbou desktopových aplikácií s grafickým užívateľským rozhraním (GUI, graphical user interface) a ako taká je cennou programátorskou skúsenosťou.

Bola tiež prvou príležitosťou, kedy som si mohol vyskúšať prostredie pre RAD a jednoduchú tvorbu GUI. Skratka RAD znamená Rapid Application Development, vo voľnom preklade rýchly vývoj aplikácií (programov).

Niektoré časti programu sú jednoduchšie, pretože sa počíta s ich budúcim rozširovaním na základe požiadaviek konkrétnych zákazníkov firmy Intech Control, ktorí by mali o aplikáciu tohto typu záujem.

Pôvodne mala byť aplikácia mať webové rozhranie a používať jazyk PHP a jeden z vybraných databázových systémov, ktorých porovnanie som realizoval v časti 2 tejto práce. Po ukončení práce na tejto časti však bolo rozhodnuté, že tento informačný systém bude realizovaný použitím technológií, ktoré sa už vo firme Intech Control používajú aj pre ich vlastné potreby.

Zoznámil som sa s vývojovým prostredím Microsoft Visual Studio 2008 a programovacím jazykom Visual Basic .NET, ktorý je deviatou verziou tohto jazyka. Ako naznačuje samotný názov jazyka, je viazaný na platformu .NET a v použitej verzii je možné používať vlastnosti Microsoft .NET Framework 3.5, ktorá je v čase písania tejto práce najnovšou verziou Microsoft .NET Framework.

Zoznámenie sa s jazykom Visual Basic .NET nebolo bez problémov. Najväčším z nich bola dokumentácia, pretože najlepším zdrojom informácií bola on-line dokumentácia na webových stránkach MSDN (Microsoft Developer Network), ktoré sú dostupné na adrese http://msdn.microsoft.com/. Inak bolo potrebné vyhľadávať použiteľné praktické ukážky použitia niektorých objektov a funkcií prostredníctvom vyhľadávačov, akým je napríklad Google.

Získal som nové znalosti o práci s Microsoft SQL Server 2005 Express, aj napriek tomu, že som nevyužil všetky jeho vlastnosti. Microsoft SQL server sa bežne používa tam, kde sú prítomné technológie a programovacie jazyky podporované firmou Microsoft, preto predpokladám, že tieto znalosti v budúcnosti využijem aj na iné účely.

Keďže konečná aplikácia bude využívaná firmou Intech Control, aj keď sa

predpokladá modifikácia tohto produktu pre účely konkrétnych zákazníkov, budú zdrojové súboru tejto práce dostupné na požiadanie okrem sady ikon Silk, ktorá bola použitá bez akýchkoľvek úprav a je voľne dostupná nielen z webovej stránky autora tejto súpravy ikon. Je možné, že v budúcnosti prejde firma Intech Control na inú súpravu ikon z licenčných dôvodov. Tie ale nie sú prekážkou pre ich použitie v tejto práci.

Na kompletizáciu riešenia pre sklad paliva je potrebné ošetriť nízko úrovňovú časť spomínanú aj v úvode k tejto práci. Tá mala byť náplňou inej diplomovej práce, ale nakoniec sa tak nestalo.

Zoznam použitej literatúry

[1] Prispievatelia projektu Wikipedia, Wikipedia, 2008, http://en.wikipedia.org/

[2] Williams, Hugh E., Lane David, PHP a MySQL, 2002, str. 10-12, 102-104,

[3] Suma C. Shastry, Mohan Saraswatipura, DB2 performance tuning using the DB2 Configuration Advisor, 2004, http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0605shastry/

[4] OSDB, the Open Source Database Benchmark, 2006, http://osdb.sourceforge.net/

[5] Peter Šlapanský, Fórum IBM DB2 Express-C, 2008, http://www.ibm.com/developerworks/forums/thread.jspa?threadID=195958

Prílohy

SQL pre vytvorenie databázy a databázových tabuliek pre informačný systém Palivo

```
create database palivo;
create table zakaznici
(
    id                          int identity(1,1) NOT NULL,
    nazov                       varchar(255) NOT NULL,
    ulica                       varchar(255) NOT NULL,
    mesto_id                    int NOT NULL,
    psc                         varchar(10) NOT NULL,
    stat_id                     int NOT NULL,
    telefon                     varchar(20),
    fax                         varchar(20),
    kontakt                     varchar(255) NOT NULL,
    mobil                       varchar(20),
    email                       varchar(255),
    poznamka                    varchar(255),
    moze_odoberat               tinyint NOT NULL
);

create table mesta
(
    id                          int identity(1,1) NOT NULL,
    nazov                       varchar(255) NOT NULL,
    stat_id                     int NOT NULL
);

create table staty
(
    id                          int identity(1,1) NOT NULL,
    nazov                       varchar(255) NOT NULL,
);
```

```
create table vodici
(
    id                          int identity(1,1) NOT NULL,
    meno                        varchar(255) NOT NULL,
    priezvisko                  varchar(255) NOT NULL,
    zakaznik_id                 int NOT NULL,
    tahac_id                    int NOT NULL,
    naves_id                    int NOT NULL,
    skolenie                    tinyint NOT NULL,
    poznamka                    varchar(1000)
);

create table vozidla -- tahace
(
    id                          int identity(1,1) NOT NULL,
    znacka                      varchar(50) NOT NULL,
    vodic_id                    int NOT NULL,
    poznamka                    varchar(1000)
);

create table navesy
(
    id                          int identity(1,1) NOT NULL,
    znacka                      varchar(50) NOT NULL,
    cislo_licencie              int NOT NULL,
    vodici_id                   int NOT NULL,
    poznamka                    varchar(1000)
);

create table vydajne_lavky
(
    id                          int identity(1,1) NOT NULL,
    popis                       varchar(255),
    produkt_id                  int NOT NULL
);
```

```
create table odbery
(
    id                          int identity(1,1) NOT NULL,
    id_objednavky               int NOT NULL,
    vodic_id                    int NOT NULL,
    komora                      tinyint,
    produkt_id                  int NOT NULL,
    stav_odberu_id              int,
    datum_cas                   smalldatetime,
    transakcia_petrocountu_id   int,
    vydajne_miesto_id           int,
    mnozstvo_gross              real, -- realne odobrane
    mnozstvo_net                real, -- realne odobrane
    avg_temperature             real, -- realne odobrane
    avg_density                 real, -- realne odobrane
);

create table objednavka
(
    id                          int identity(1,1) NOT NULL,
    vodic_id                    int,
    naves_id                    int,
    tahac_id                    int,
    odber_povolenie             tinyint,
    platnost                    tinyint,
    datum_zadania               smalldatetime,
    datum_platnosti             smalldatetime,
    cas_platnosti               smallint
    produkty_id                 xml NOT NULL
);
```

```
create table nadrze
(
    id                          int identity(1,1) NOT NULL,
    nazov                       varchar(255),
    mnozstvo_max                float NOT NULL,
    mnozstvo_akt                float,
    produkt_id                  int,
    pripravenost                tinyint
);

create table prijmy_do_skladu
(
    id                          int identity(1,1) NOT NULL,
    datum_cas                   smalldatetime,
    mnozstvo                    float,
    produkt_id                  int,
    nadrz_id                    int,
    dodavatel_id                int
);

create table produkty
(
    id                          int identity(1,1) NOT NULL,
    nazov                       varchar(255),
    popis                       varchar(2000),
    atest_id                    int,
    vlastnosti                  xml
);

create table stavy
(
    id                          int identity(1,1) NOT NULL,
    nazov                       varchar(255),
    popis                       varchar(2000),
);
```

```
create table atesty
(
    id                          int identity(1,1) NOT NULL,
    nazov                       varchar(255),
    popis                       varchar(2000),
    sprievodny_dokument         varchar(500)
);

create table dodavatelia
(
    id                          int identity(1,1) NOT NULL,
    nazov                       varchar(255) NOT NULL,
    ulica                       varchar(255) NOT NULL,
    mesto_id                    int NOT NULL,
    psc varchar(10)             NOT NULL,
    stat_id                     int NOT NULL,
    telefon                     varchar(20),
    fax                         varchar(20),
    kontakt                     varchar(255) NOT NULL,
    mobil                       varchar(20),
    email                       varchar(255),
    poznamka                    varchar(255)
);
```

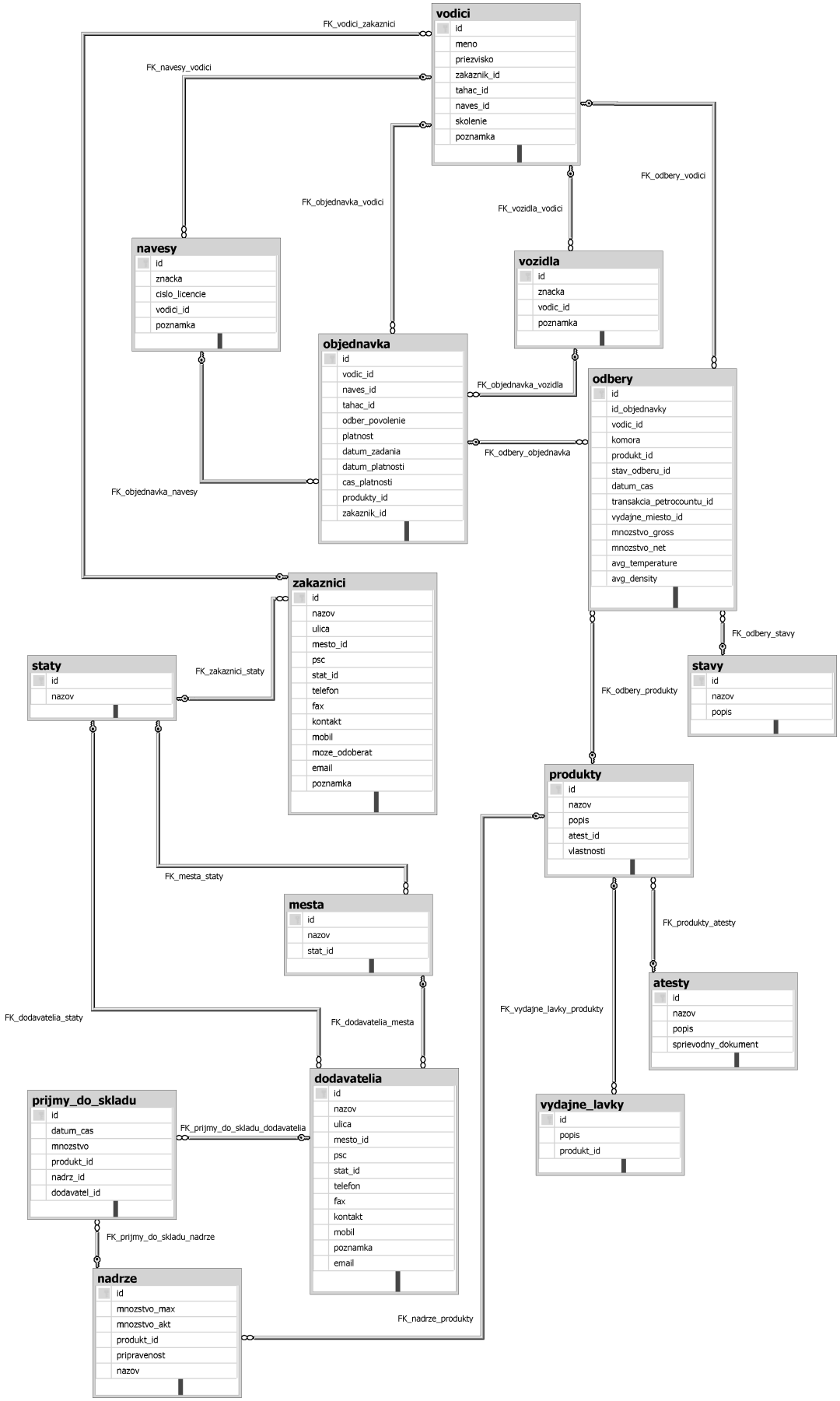
Všeobecná štruktúra XML pre vlastnosti produktov

```
<vlastnosti>
    <vlastnost>
              <nazov></nazov>
              <hodnota></hodnota>
              <jednotka></jednotka>
    </vlastnost>
</vlastnosti>
```

Všeobecná štruktúra XML pre produkty objednávky

```
<produkty>
    <produkt>
              <nazov></nazov>
              <mnozstvo></mnozstvo>
              <jednotka></jednotka>
    </produkt>
</produkty>
```

Zobrazenie návrhu štruktúry a prepojenia navrhnutých databázových tabuliek


vodici
id
meno
priezvisko
zakaznik_id
tahac_id
naves_id
skolenie
poznamka
FK_vodici_zakaznici
FK_navesy_vodici
FK_objednavka_vodici
FK_vozidla_vodici
FK_odbery_vodici
navesy
id
znacka
cislo_licencie
vodici_id
poznamka
vozidla
id
znacka
vodic_id
poznamka
objednavka
id
vodic_id
naves_id
tahac_id
odber_povolenie
platnost
datum_zadania
datum_platnosti
cas_platnosti
produkty_id
zakaznik_id
FK_objednavka_vozidla
FK_odbery_objednavka
FK_objednavka_navesy
odbery
id
id_objednavky
vodic_id
komora
produkt_id
stav_odberu_id
datum_cas
transakcia_petrocountu_id
vydajne_miesto_id
mnozstvo_gross
mnozstvo_net
avg_temperature
avg_density
zakaznici
id
nazov
ulica
mesto_id
psc
stat_id
telefon
fax
kontakt
mobil
moze_odoberat
email
poznamka
FK_zakaznici_staty
FK_odbery_stavy
FK_odbery_produkty
staty
id
nazov
stavy
id
nazov
popis
produkty
id
nazov
popis
atest_id
vlastnosti
FK_mesta_staty
mesta
id
nazov
stat_id
FK_produkty_atesty
atesty
id
nazov
popis
sprievodny_dokument
FK_dodavatelia_staty
FK_dodavatelia_mesta
FK_vydajne_lavky_produkty
dodavatelia
id
nazov
ulica
mesto_id
psc
stat_id
telefon
fax
kontakt
mobil
poznamka
email
vydajne_lavky
id
popis
produkt_id
prijmy_do_skladu
id
datum_cas
mnozstvo
produkt_id
nadrz_id
dodavatel_id
FK_prijmy_do_skladu_dodavatelia
FK_prijmy_do_skladu_nadrze
nadrze
id
mnozstvo_max
mnozstvo_akt
produkt_id
pripravenost
nazov
FK_nadrze_produkty