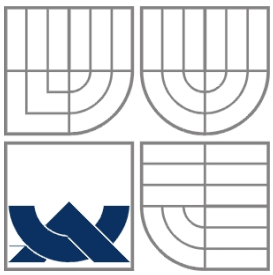
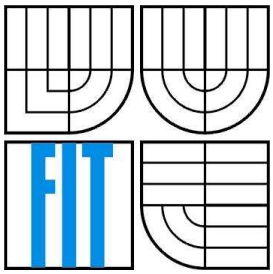


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DATABÁZOVÝ SYSTÉM EVIDENCE IMPLANTABILNÍCH ZDRAVOTNICKÝCH PROSTŘEDKŮ

EVIDENCE DATABASE SYSTEM OF HEALTH IMPLANTABLE DEVICES

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR
ONDŘEJ DLUHOŠ

VEDOUCÍ PRÁCE
SUPERVISOR
ING. PETR JAŠA

BRNO 2008

Abstrakt

Tato bakalářská práce se zabývá vytvořením nového databázového systému pro firmu Timplant, která vyrábí a expeduje zubní implantáty a jejich příslušenství. Úkolem této práce je analyzovat původní systém, navrhnout a realizovat nový databázový systém s využitím volně šiřitelného software. Nový systém bude navrhnutý dle požadavků firmy a také musí uchovávat data dle aktuálního Nařízení vlády.

Klíčová slova

Databáze, Java, MySQL, JDBC, Netbeans, SQL

Abstract

This bachelor thesis creates a new database system for the company Timplant, which produces and delivers dental surgical implants and their accessories. The task of the project is to analyze the original system design and to create a new database system with use of a freeware. The database will be designed in accordance with the demands of the company and will archive all the data as required by the current government regulations.

Keywords

Databáze, MySQL, Java, JDBC, Netbeans, SQL

Citace

Dluhoš Ondřej: Databázový systém evidence implantabilních zdravotnických prostředků. Brno, 2008, bakalářská práce, FIT VUT v Brně

Databázový systém evidence implantabilních zdravotnických prostředků

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Petra Jaši. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Ondřej Dluhoš
12.5.2008

Poděkování

Chtěl bych na tomto místě poděkovat panu Ing. Petru Jašovi za odborné vedení a konzultace, které mi poskytl v teoretické a implementační části bakalářské práce.

Obsah

1 Úvod

Počítače zpracovávají data od počátku své historie a schopnost aplikace vkládat, ukládat a uchovávat data, se kterými pracuje, je dnes považována za naprostou samozřejmost. Každý počítač potřebuje pro svůj chod operační systém a v závislosti na této platformě jsou vyvíjeny různé programy pro firmy, instituce, školy, státní správu, atd. Jelikož vývoj počítačů od 90. let šel značně kupředu, tak i vývoj všech aplikací s nimi musel držet krok. Vznikaly nové operační systémy a programy „šité na míru“ musely být inovovány pro svůj bezchybný chod. Tento postup techniky zachvátil snad skoro každého a tak i firma mého otce musela postupem času přecházet na nové („lepší“) operační systémy a potýkat se se změnou databázového systému, který je pro chod firmy velmi důležitý.

Tato bakalářská práce se zabývá vytvořením nového databázového systému pro firmu Timplant. Vlastníkem této firmy je můj otec a o to snazší bude komunikace při realizaci celého programu. Firma se zabývá výrobou a expedicí zubních implantátů a jejich příslušenství. Její zákazníci, vyrobené a expedované zboží je ukládáno do databázového systému. Tyto údaje musí být uchovávány dle Nařízení vlády a jsou nezbytné pro činnost firmy.

Text této práce je rozdělen do několika navazujících kapitol. První kapitola má název Úvod. V druhé kapitole se seznámíme s původním databázovým systémem implementovaném ve FoxBASE a následně v druhé polovině devadesátých let v programovacím jazyku FoxPro. Díky častým změnám v zákonech byl program často modifikován a téměř neustále vyvíjen a laděn. Dozvíme se zde také, že systém byl navržen pro MS-DOS, který se v té době jevil jako nejspolehlivější operační systém.

Třetí kapitola analyzuje funkcionalitu celého programu a jsou zde stanoveny požadavky na nový systém. Seznámíme se s prostředím původního systému, jsou zde rozebrány všechny nabídky a uspořádání dat do databázových tabulek. Seznámíme se s jakými daty a položkami bude databáze plněna a budou popsány operace se záznamy v databázi.

V další části se zaměříme na návrh nového systému pomocí jazyka UML. Pro návrh nového systému bude pomocí jazyka UML vytvořen E-R diagram a diagram užití. V této kapitole budou probrány důvody pro volbu daného implementačního jazyka a technologií. Zaměřím se na moderní technologie volně šiřitelného software.

Pátá kapitola zahrnuje realizaci nového systému a jeho nasazení do praxe. Jsou zde rozebrány hlavní části celého programu a podrobněji se zaměřím na komunikací mezi serverem MySQL a programem. Poměrně detailně je zde popsána migrace dat z původního systému a problémy, které nastaly při implementaci. Podkapitola 5.3 zahrnuje zálohování dat, což patří k nejdůležitějším úkonům, které by měl databázový systém provádět. Na tuto podkapitolu navazuje bezpečnost dat v systému, kde se podíváme na bezpečnost dat v MySQL, zamezení přístupu do programu a šifrování dat zálohy. V závěru kapitoly se stručně seznámíme s prostředím systému a jeho nasazením do firmy. Poslední kapitolou je závěr, který celou práci shrnuje.

2 Seznámení s původním systémem

2.1 Požadavky Nařízení vlády na systém

Od roku 2000 je pro výrobce zdravotnických prostředků v České republice povinnost dodržovat požadavky, které jsou stanovovány nařízeními vlády české republiky. Tato nařízení vycházela z mezinárodních norem, ve kterých byly podrobně specifikovány požadavky na návrh, výrobu a výstupní kontrolu zdravotnických výrobků s přihlédnutím k momentálnímu stavu legislativy v ČR a také v souvislosti s budoucím vstupem ČR do Evropské unie. Požadavky jsou odstupňovány dle tříd, do kterých jsou zdravotnické výrobky zařazovány. Nejkomplikovanější skupina zdravotnických prostředků (ZP) resp. skupina, pro kterou je dle Nařízení vlády č. 336/2004 Sb. v platném znění (EU směrnice 93/42EEC v platném znění) předepsáno nejvíce požadavků je skupina III. Do této skupiny patří aktivní implantabilní prostředky (např. kardiostimulátory). Do skupiny II.b byly začleněny pasivní implantabilní prostředky zdravotnické techniky, jako jsou např. zubní implantáty. Ve skupině II.a jsou ZP, které jsou trvale ve styku s lidskou tkání (např. příslušenství k zubním implantátům – pilířové nadstavby). Do skupiny I. byly zařazeny mimo jiné chirurgické nástroje.

Pro výrobce dentálního implantačního systému je v současné době povinností dodržovat požadavky aktuálního Nařízení vlády a jeho častých změn. Pokud je výrobce nositelem certifikátu systému managementu jakosti dle norem např. ČSN EN ISO 9001 a ČSN EN ISO 13485, je nutné dodržovat podmínky dle uvedených norem. Kontrolu dodržování požadavků provádí Česká obchodní inspekce. Instituce, která vydala certifikát, provádí kontrolu dodržování požadavků na systém managementu jakosti každoročním auditem.

Předpis pro sledování, dodržování a dokumentování konkrétních požadavků a postupů, vyplývajících z Nařízení vlády a ISO norem, je stanoven pro výrobky třídy II.b, II.a a I v příručce jakosti firmy. Dle kapitoly 7.5.3. *Identifikace a sledovatelnost* příručky jakosti je nutné, aby výrobce zajistil identifikaci a sledovatelnost výrobků po dobu nejméně 10 let od data expedice k odběrateli.

Ve smyslu této příručky je konečným odběratelem zákazník, tedy zubní lékař, který implantát odebral. Povinnost výrobce je držet informace o výrobku od vstupního materiálu až po základní údaje o zákazníkovi 10 let a v případě potřeby (kontroly nebo nežádoucí příhody) tyto informace použít.

Další zásadní požadavek systému managementu jakosti je dle Příručky jakosti uveden v kapitole 8.3. *Řízení neshodného produktu*. V případě většího výskytu stejné neshody je dle předpisu pro stahování výrobků povinností příslušnou šarži výrobků stáhnout zpět od zákazníka a nahlásit nežádoucí příhodu v souladu s par. 32 zákona 346/2003 Sb. a v souladu se Sb. zák. č. 501/2000.

Informace o produktech, zákaznících a jejich vyhodnocení, které jsou požadovány systémem jakosti pro vývoj, výrobu, servis zubních implantátů, příslušenství a chirurgických nástrojů, je uvedena v Příručce jakosti celá řada. Dodržování předepsaných požadavků a jejich realizace je

existenčním procesem v činnosti firmy. Zdroje, ze kterých jsem převážně čerpal při psaní této kapitoly, jsou [1], [2], [3] a [4].

2.2 Databázový systém Tilat

Osvědčený databázový systém, který byl schopen respektovat již zmíněné požadavky, byl vyvíjen na zakázku několik let. Tento program byl nejprve vyvíjen v prostředí databázového programovacího jazyku FoxBASE a následně v druhé polovině devadesátých let v programovacím jazyku FoxPro. Díky častým změnám v zákonech byl program s pracovním názvem *Tilat* často modifikován a téměř neustále vyvíjen a laděn. Pracovní název *Tilat* vznikl složením začátku slov Timplant a Latin1, což představuje název firmy a způsob kódování českých znaků v celém programu.

Vzhledem k tomu, že operační systém MS Windows 95 neposkytoval záruku bezchybné funkce a přenositelnosti na jiné verze MS Windows, byl program *Tilat* navržen pro prostředí MS DOS. Toto prostředí bylo dostatečně stabilní na a umožnilo nezávislý chod na MS Windows.

Současné MS Windows nepodporuje jádro DOSu v režimu, který by zajišťoval spolehlivé provozování programu. Vznikají zde problémy se spuštěním programu a jeho bezchybným chodem, který je pro firmu důležitý. Stávající program už není možno dále upravovat, protože není k dispozici zdrojový kód programu, celý systém je již zastaralý a potřeboval by značné inovace.

V dnešní době nalezneme dokonalejší prostředky pro tvorbu databázového systému, než před několika lety. Proto vznikl požadavek na vytvoření nového programu, který by byl v souladu s aktuální legislativou a softwarovou úrovní. Nový systém musí být bez problému funkční v aktuálním prostředí Windows XP. Firma využívá tento operační systém (OS) již několik let a v dohledném období nepředpokládá přechod na jiný OS, tudíž nově vyvíjený program bude prioritně testován na Windows XP. Výsledky testování na jiných OS se budou brát v úvahu, protože musíme myslet na budoucnost, ale dle požadavků bude vyvíjený systém prioritně funkční pro OS Windows XP.

2.3 FoxPro

FoxPro je systém řízení databáze (DBMS[1]) pro prostředí operačního systému MS-DOS. Pro ukládání dat používá formát souborů DBF. Programovací jazyk využívá procedury a patří do skupiny jazyků xBase. V naší zemi byl značně populární a rozšířen pro svoji rychlost, stabilitu a v neposlední řadě i licenční politiku bezplatného šíření runtime pro běh vytvořených aplikací. Pro vývojové prostředí a vlastní aplikace je charakteristické použití myši a uživatelských oken. Navazuje na rozšířenou a ve

[1] Zpracováním a přístupem údajů v databázi bývá pověřen program DBMS (DataBase Management System), nebo česky SŘBD (Systém Řízení Báze Dat).

své době populární FoxBASE. Poslední verze v této podobě má označení 2.60 z roku 1994 a existovala i pro operační systémy Windows, UNIX a Macintosh. Později ji společnost Microsoft rozšířila do současné moderní podoby pod názvem Microsoft Visual FoxPro. V současné době se jedná o komerční produkt s objektově orientovaným programováním. Aktuální verzi můžeme najít pod označením Microsoft Visual FoxPro 9.0 Professional.

Databázový engine FoxPro byl ve své době do značné míry výjimečný metodou vyhledávání nazývanou Rushmore, která specificky používala indexové soubory k vyhledávání a aktualizaci ve velkých objemech dat. FoxPro 2.0 byla původně vytvořena v překladači jazyka WatCOM C++. Charakteristické je dobré využití operační paměti a přenositelnost datových souborů a programových kódů zaměřených na zpracování těchto dat mezi řadou operačních systémů: Windows, UNIX a Macintosh.[5]

3 Analýza funkcionality systému

Aktuální systém se spouští s příkazového řádku MS Windows. Dávkový soubor tilat.bat při spuštění nastavuje klávesnici na keybgr, testuje existenci zálohového souboru, nastavuje adresáře a spouští modul, který vznikl překladem zdrojového kódu. Bohužel, v současné době není k dispozici ani zdrojový kód ani programátor, který program vyvíjel. Takže informace o struktuře programu jsou získány analýzou funkcionality programu.

Vstup do programu je zabezpečen heslem. Po otevření okna obrazovky se objeví MENU, které obsahuje 8 položek: 0-konec programu, 1-nová věta, 2-přehled dat, 3-opravy a prohlížení, 4-služby, 5-jmenovky, 6-asanace souborů, 7-tiskové sestavy, 8-databáze. Všechny tyto položky hlavního menu mohou být zvoleny šipkami nahoru nebo dolu, klávesovou zkratkou (číslice) nebo klepnutím kurzorem na příslušnou nabídku.

3.1 Nabídky systému

Základní informace o spuštěné databázi jsou uvedeny v horním řádku. První položka zleva označuje vybranou databázi, druhá položka časové období pro výběr od – do, následuje položka datum poslední aktualizace a aktuální datum.

0. Konec programu

Korektně ukončí program. Před ukončením se ukládají databázové tabulky do příslušných souborů.

1. Nová věta

Umožňuje přidat nové data do databáze. Jedná se o přidání zákazníka, zboží či položky do katalogu. Pokud chceme přidat nové data do databáze, musíme nejprve vybrat konkrétní databázi položkou z menu *8. Databáze*.

2. Přehled dat

Zobrazí souhrn všech dat dle zvolené databáze. Můžeme zde např.: vyhledávat, tisknout data, filtrovat, zobrazit sumu dat, označit či zobrazit zvýrazněnou položku v tabulce. Při vybrání zvýrazněné položky zde můžeme tuto položku upravovat nebo smazat. Další volby nám umožňují vyhledávat, skočit na jinou položku, zobrazit kalkulačku, zobrazit následující nebo předchozí položku, tisk dat (jmenovka, obálka, věta, zboží), zobrazit zboží v určitém časovém úseku, výběr dat dle kritérií, zobrazit přehled dat atd. Všechny volby menu, které jsou nám nabídnuty, můžeme vyvolat pouze klávesovou zkratkou.

3. Opravy/prohlížení

Umožňuje provádět opravy, prohlížení, mazání a další úkony se zvolenou databází. Všechny volby jsou totožné, jako kdybychom v hlavním menu zvolili *2. Přehled dat* a vybrali zvýrazněnou položku klávesou ENTER. Popis všech možností je uveden o odstavec výše.

4. Služby

Tato sekce obsahuje různé služby, které lze s programem provádět.
0. Konec -> návrat - návrat do hlavního menu
1. Zálohování dat - záloha databázových souborů na zvolené umístění
2. Obnova databází - obnova databázových souborů ze zálohy
3. Oprava indexů - oprava indexů po nekorektním ukončení programu
4. Změna hesla - změna přístupového hesla do programu
5. Tiskárna - změna připojené tiskárny (umožňuje nastavit tiskárnu pouze LASER JET 4P nebo EPSON)
6. Uživatel – definuje formát zobrazení údajů firmy pro tisk obálek
7. Období exp. zboží - omezuje zboží na období dle data expedice

5. Jmenovky

Můžeme vytisknout jmenovky na dopisy po zvolení tiskové sestavy. Tiskovou sestavou je myšlen soubor, který se nachází ve stejném adresáři jako program.

6. Asanace souborů

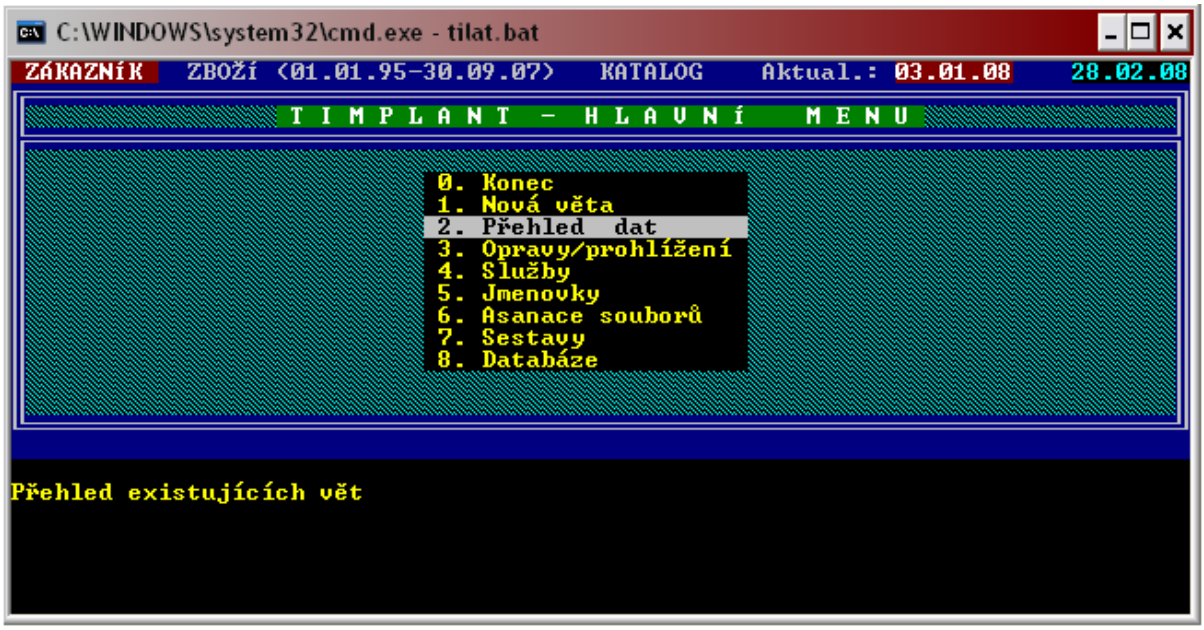
Data v databázi jsou chráněna proti nechtěnému smazání právě touhle volbou. Pokud chceme některá data z databáze smazat, můžeme tak učinit z hlavního menu *2. Přehled dat* nebo *3. Opravy/prohlížení*. Data ovšem nebudou smazána z databáze, ale pouze označena ke smazání a budou nadále zobrazena v přehledu dat. Takto označená data nejsou nijak změněna a v případě potřeby mohou být odznačena a znovu používána. Asanace souborů umožňuje nenávratně smazat všechny označené data ke smazání z databáze.

7. Tiskové sestavy

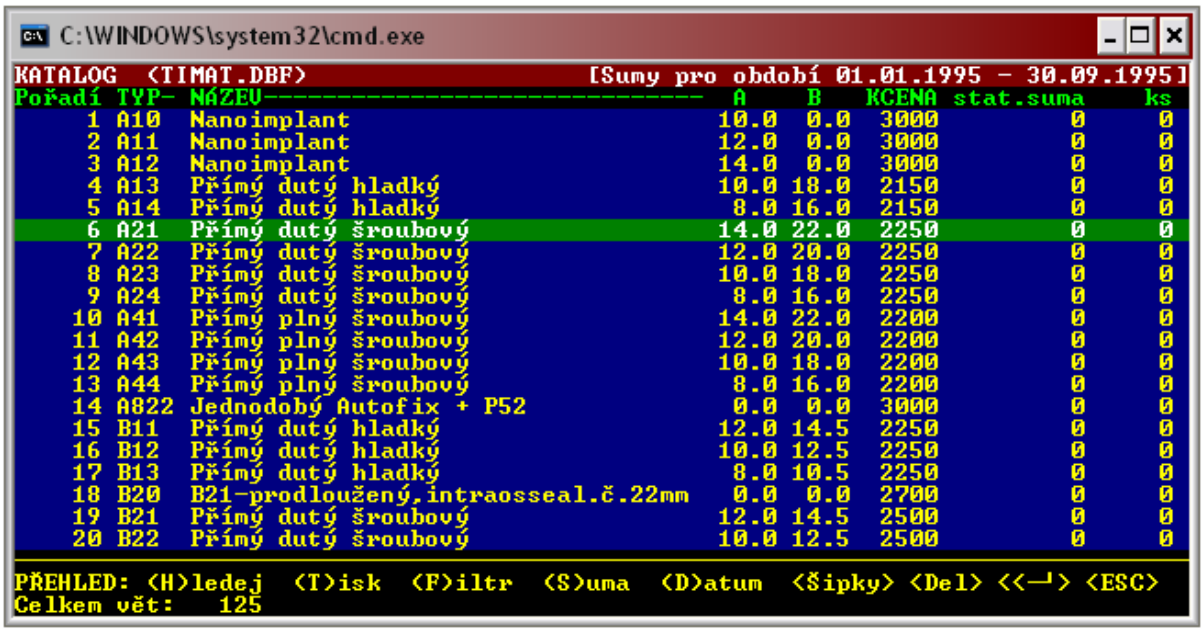
Tisk různých sestav po zvolení tiskové sestavy. Tiskovou sestavou je myšlen soubor, který se nachází ve stejném adresáři jako program.

8. Databáze

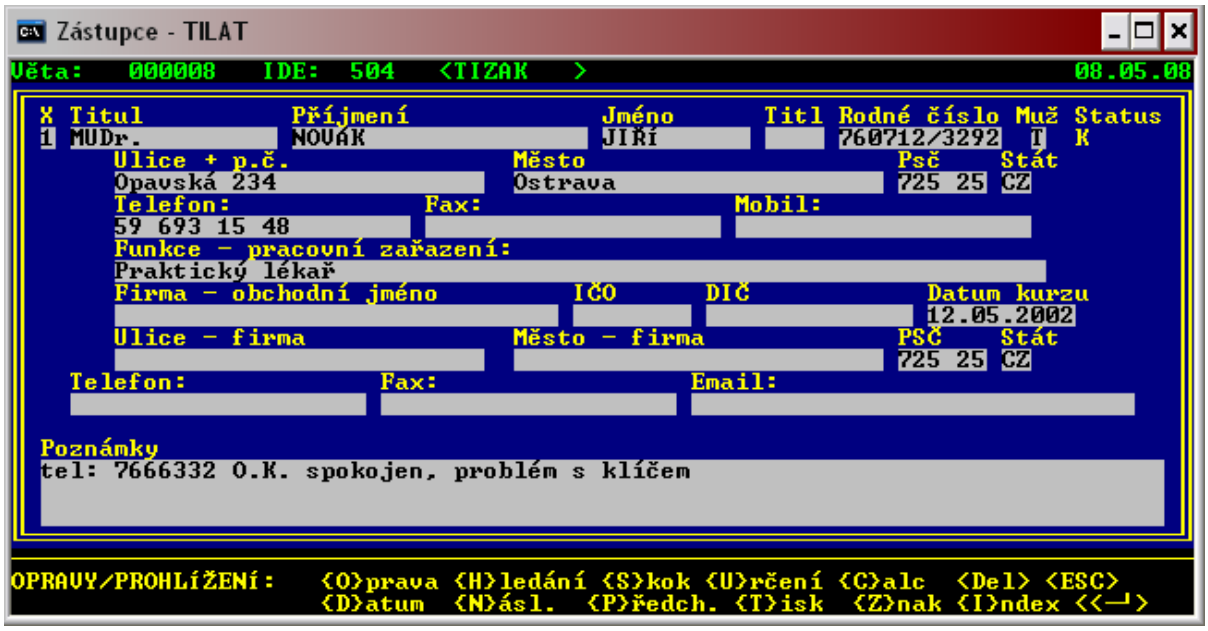
Nejdůležitější položka hlavního menu umožňuje výběr ze tří databází: ZÁKAZNÍK, ZBOŽÍ a KATALOG. Program *Tilat* je založen na spolupráci těchto tří na sobě závislých relací. Zvolením databázového souboru vybíráme databázi, se kterou chceme v programu pracovat. Pokud chceme pracovat např. s databází zákazníka (*1. Nová věta, 2. Přehled dat, 3. Opravy/prohlížení*, atd.) musíme nejprve zvolit databázový soubor ZÁKAZNÍK.



Obrázek 3.1 Hlavní menu databázového programu Tilat



Obrázek 3.2 Přehled dat katalogu



Obrázek 3.3 Oprava či prohlížení zákazníka

Obrázek 3.3 nám zobrazuje kartu zákazníka, kterou lze opravovat či prohlížet. V prvním řádku jsou poznámky vybraného zákazníka, jsou pouze informativní a nelze je upravovat. Zobrazení údajů je závislé na výběru tabulky a místa, kde se nacházíme.

- *Věta* - jedna položka v databázi a vyjadřuje pořadové číslo v příslušném výběru
- *IDE* - unikátní číslo přiděleno zákazníkovi a nelze jej modifikovat
- *TIZAK* - označení aktuální databáze (TIZAK, TIZBO, TIMAT)
- *PRÁZDNÁ* - věta nemá základní údaj (jméno a příjmení), v tomto místě se může objevovat označení věty pro výmaz - *ZRUŠENA*
- *V pravém horním rohu můžeme nalézt aktuální datum*

Hlavní část obrazovky nám tvoří údaje o zákazníkovi a v posledním řádku nalezneme volby pro konkrétní databázi. Přehled těchto voleb v celém systému je uveden v kapitole *4.1.2 Diagram užití*.

3.2 Databáze systému

Jak již bylo řečeno, celý systém byl vyvíjen několik let a během této doby se celá databáze minimalizovala na použití pouze tří tabulek. Databáze obsahuje tabulky: ZÁKAZNÍK, ZBOŽÍ a KATALOG.

Tabulka ZÁKAZNÍK obsahuje veškeré údaje o zákazníkovi: jméno, příjmení, tituly, adresy firemní i soukromé, telefony, e-mail, datum kurzu, poznámku atd. Indikátor X označuje skupinu zákazníků č. 1 lékaři – stomatologové, č. 2 dodavatelé a č. 3 ostatní. Položka STATUS indikuje výběr

věty například pro hromadný tisk obálek. Aktivní věta má status „+“. V tomto režimu lze vkládat, měnit a mazat údaje o zákazníkovi a jeho odebraném zboží. Tabulka může obsahovat až 35 údajů o zákazníkovi, z nichž jsou povinné pouze dva údaje *příjmení* a *IDE* (generováno automaticky). Pro rychlý přehled dat zákazníků v programu jsou v tabulce uchovávány položky s názvem ZNK, ZNN, ZNI, ZNP, ZNR, které jsou modifikovány v závislosti s odebraným zbožím zákazníka a dle toho zda zákazník absolvoval implantační kurz. Jejich použití a popis bude uveden v následující kapitole.

Tabulka ZBOŽÍ obsahuje veškeré zboží bez ohledu na zákazníka. Po vyvolání přehledu lze vybrat jakékoliv zboží, které bylo vyexpedováno. Postupným prohlížením typu výrobků se objevuje i jméno zákazníka, který výrobek odebral, včetně všech potřebných údajů pro identifikaci výrobku. V tomto režimu lze vkládat, měnit a mazat údaje o zboží.

Tabulka KATALOG obsahuje veškeré typy výrobků včetně jejich specifikace rozměrů, názvu, jednotkové ceny, celkového počtu vyexpedovaných výrobků konkrétního typu a celkové ceny. V tomto režimu lze vkládat, měnit a mazat údaje o zboží v katalogu.

Na přehledové kartě každé z databází je výběr z možností: TISK, VYHLEDÁVÁNÍ, FILTR, SUMA, DATUM a DELETE. Po rozbalení nabídky TISK je možné vytisknout na monitor nebo tiskárnu kompletní přehled podle typu databáze (zákazník, zboží nebo katalog). Vyhledávání umožňuje najít jakoukoliv položku v databázi. Filtr tvoří podmínky pro výběr v jednotlivých databázích. Tyto podmínky lze modifikovat s minimální znalostí základních příkazů FOX PRO. Suma udává součet katalogové ceny vybraných položek v časovém období, které je dáno položkou datum. Ta umožňuje získat údaje v určitém časovém úseku (den, měsíc, rok, několik let). Příkaz *delete* označí položku k vymazání z databáze. K vlastnímu výmazu dojde až po zadání příkazu ASANACE SOUBORŮ z hlavního menu.

Při spouštění databázového systému jsou načítány jednotlivé databázové soubory, které musí být uloženy v adresáři spouštěného programu. Každá tabulka z databáze má přiřazen konkrétní soubor pro čtení a ukládání ve formátu „*.dbf“. Pro jednotlivé tabulky jsou přiřazeny soubory *timat.dbf* (katalog), *tizak.dbf* (zákazníci) a *tizbo.dbf* (zboží). Nově vytvořený systém bude muset tyto soubory načíst a převést do vytvořené databáze v MySQL. Mimo jiné soubory obsahují doplňkové informace o databázi, se kterými se nedá pracovat, ale jsou podstatné pro zobrazení a funkce, kterými program disponuje. Tyto informace uložené v souboru budou podrobně zkoumány při realizaci nového systému a zohledňovány při návrhu databáze v MySQL.

3.3 Požadavky na úpravu

Vzhledem k tomu, že není dostupný zdrojový kód pro úpravu stávajícího programu ani návod nebo příručka k obsluze, bude nutné vytvořit nový program se stejnou funkcionalitou vylepšenou o nové požadavky, které vyplývají z potřeb např. systému managementu jakosti.

Požadavky firmy na úpravy jsou následující:

1. Zachování stejného nebo podobného způsobu ovládání programu, rozsahu funkcí a služeb (ovládací prvky, vstup a výstup dat, tiskové rutiny, atd.)
2. Volně šiřitelný systém pro aktuální OS MS Windows XP
3. Bezchybný opakovatelný převod dat ze stávajícího programu do nové verze
4. Možnost volby ovladače pro libovolnou tiskárnu
5. Automatické průběžné zálohování
6. Zamezení nežádoucího přístupu k datům i k zálohovaným souborům
7. Umožnit ladění nového programu paralelně s původní verzí (uložené data v Tilat převádět do nové verze)
8. Vytvořit systém hlídání data narození pro jednotlivé zákazníky

4 Koncept nového systému

Konceptuální modelování, které se v databázové terminologii označuje někdy také jako konceptuální návrh, patří do etapy analýzy požadavků. Jeho cílem je analyzovat požadavky na data, která budou uložena v databázi. Z hlediska klasifikace datových modelů, patří mezi modely založené na objektech (konceptech) aplikační domény, pro kterou softwarový systém vyvíjíme. Jestliže budeme vyvíjet databázový systém zdravotnických prostředků, budeme modelovat data reprezentující zákazníky, zboží, katalog a vztahy mezi nimi. Někdy se tato úroveň modelování také označuje jako sémantické modelování.[6]

Návrh aplikace může teoreticky vzniknout pouze za použití tužky, mozku a poznámkového bloku, s jejichž pomocí načrtnete jednoduchý model aplikace. Model ale bude používat jen vám známou syntaxi a sémantiku, takže budete jen těžko své záhadné omalovánky s někým sdílet. Dalším problémem je, že za těmito náčrty většinou nestojí žádný logický metamodel ani formálně definovaná sémantika jednotlivých elementů modelu. Pro tento účel vnikl jazyk UML (Unified Modeling Language), který umožňuje modelovat jakékoli aplikace pomocí stejné formální syntaxe, čímž usnadňuje sdílení informací, vývoj a implementaci libovolných softwarových mechanismů.[7]

4.1 Jazyk UML

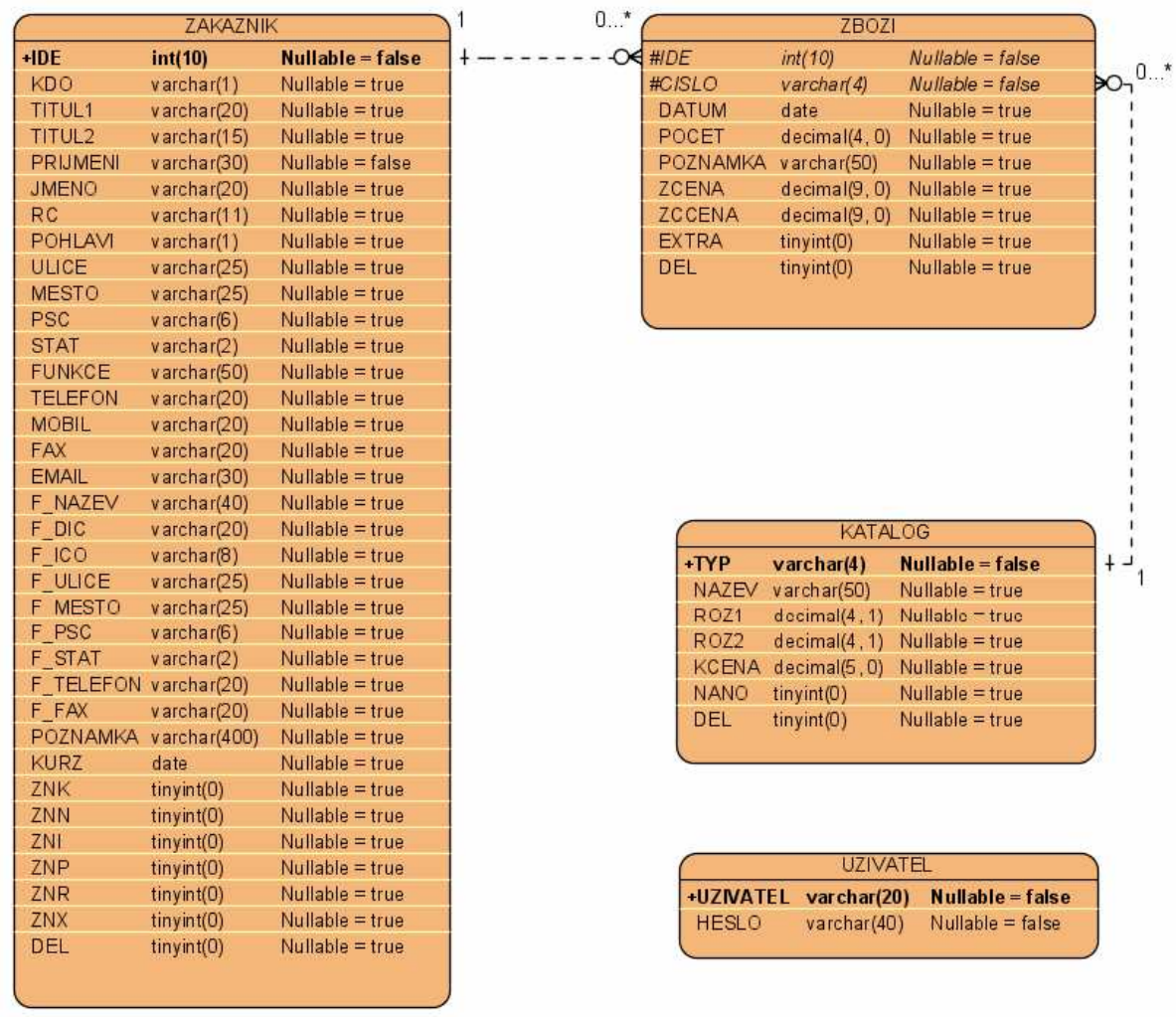
Historie jazyka není nijak dlouhá, jedná se o metodologii vytvořenou předními kapacitami oboru informatiky (Booch, Rumbaugh, Jacobson). Předcházela mu snaha mnoha znalců o vytvoření kvalitní metody, která by umožňovala kvalitní modelování. Jazyk UML je v softwarovém inženýrství grafický jazyk pro vizualizaci, specifikaci, navrhování a dokumentaci programových systémů. UML nabízí standardní způsob zápisu jak návrhů systému včetně konceptuálních prvků jako jsou business procesy a systémové funkce, tak konkrétních prvků jako jsou příkazy programovacího jazyka, databázová schémata a znovupoužitelné programové komponenty. UML neobsahuje způsob, jak se má používat, ani neobsahuje metodiku, jak analyzovat, specifikovat či navrhovat programové systémy. [8]

Jednou z důležitých charakteristik UML je jeho nezávislost na metodologiích. Cílem jazyka bylo vytvořit specifikaci nezávislou na konkrétních programovacích jazycích a procesech vývoje a analýzy. Možná i z toho pramení jeho široké rozšíření jakožto implementačního jazyka. UML se snaží používat grafickou syntaxi z různých zdrojů (metod), protože tato notace je již zažitá a nemá význam ji proto měnit. Důležité je tedy skloubení syntaxe, což se UML daří. [8]

Pro návrh nového systému použiji z jazyka UML E-R diagram (Entity Relationship Diagram) a diagram užití (Use Case Diagram).

4.1.1 E-R diagram

Je mnoho způsobů, jak vytvořit funkční model databáze. Jednou z možností je vytvořit E-R diagram. E-R diagram modeluje data, která potřebujeme v systému uchovávat a vztahy mezi těmito daty. E-R diagram je síťový model popisující návrh uložených dat v systému na vyšší úrovni abstrakce. V souvislosti s E-R diagramem se vyskytují pojmy **entita**, **atribut** a **vztah**. Entitami jsou v našem případě ZÁKAZNÍCI, ZBOŽÍ, KATALOG a UŽIVATELÉ. V E-R diagramu se vyznačují jako obdélníky s popiskem. Mezi jednotlivými entitami jsou definovány vztahy. Zde nám vztahy říkají, že zákazník může mít více odebraného zboží a v tabulce zboží se vyskytují výrobky z katalogu, atd. Atributy jsou vlastnosti entity a jsou patrny z následujícího schématu databáze. Jsou jimi položky např. IDE, DATUM, PRIJMENI, CISLO atd.

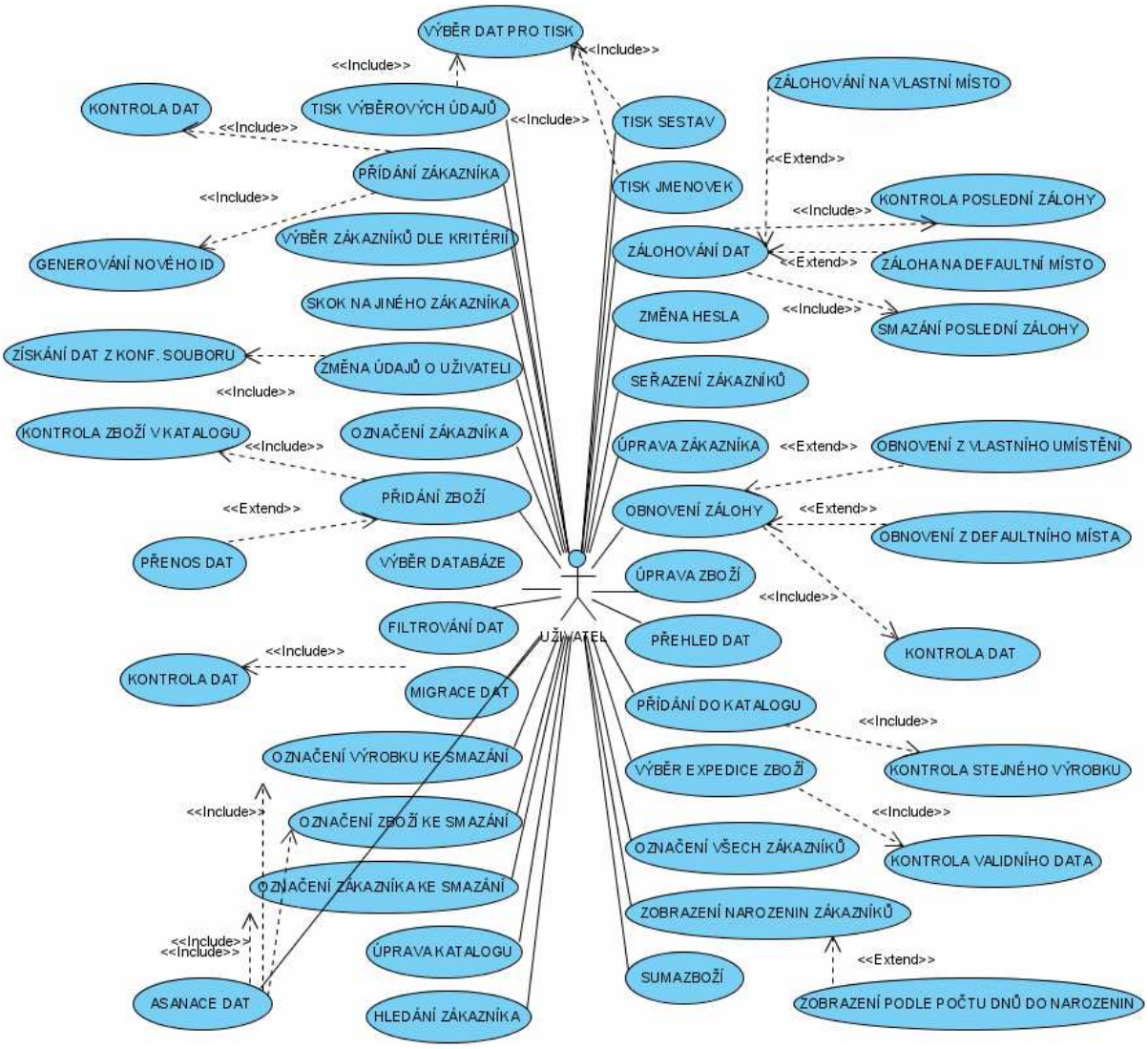


Obrázek 4.1 E-R diagram systému

4.1.2 Diagram užití

Pro zachycení požadavků na systém se používají diagramy případů užití, které jsou součástí modelovacího jazyka UML. Klíčovými aktivitami metodiky ve fázi specifikace požadavků jsou nalezení případů užití a jejich účastníků a nalezení detailů vybraných případů užití. Jeden případ užití je chápan jako funkce, kterou systém vykonává jménem jednotlivých účastníků nebo v jejich prospěch. [9]

Přestože UML pro tuto specifikaci nezavádí žádný pevně daný formát, bývá pravidlem použití nějaké tabulkové struktury, která obsahuje jméno případu použití, identifikátor případu použití, popis, seznam aktérů, hlavní tok, alternativní toky a další elementy. Vzhledem k tomu, že specifikace případů použití tímto způsobem by díky jejich množství zabrala několik stránek, omezíme se pouze na nestrukturovaný a stručný textový popis. Diagram užití pro nový systém zobrazuje hlavního aktéra *uživatele* systému.



Obrázek 4.2 Stručný diagram užití uživatele

4.2 Volba volně šiřitelného software

Firma Timplant má ve specifikaci na nový systém, využití volně šiřitelného software. Není divu, že tento požadavek vznikl, protože náklady na zakoupení a upgrade nového software by rozhodně nebyly malé. Aktuální databázový systém firmy byl psán ve FoxPro, který v té době byl zdarma k využití. Později společnost Microsoft rozšířila FoxPro do současné moderní podoby pod názvem Microsoft Visual FoxPro a software se tak stal komerčním. Z tohoto důvodu se musím poohlédnout po jiném, vhodném, volně šiřitelném software či programovacím jazyku. Během mého studia na FIT jsem se setkal s nemálo nástroji a software pro tvorbu databázového systému, tudíž jsem si mohl vybírat. Pečlivě jsem zvažoval a po konzultaci s vedoucím této práce jsem se rozhodl pro programovací jazyk JAVA s využitím databázového systému MySQL. Jazyk JAVA je jedním z nejpoužívanějších programovacích jazyků na světě a představuje mnoho funkcí pro práci s databází MySQL. Pro vývoj celého systému budu využívat vývojové prostředí NetBeans (NetBeans IDE) verzi 6.0.1. Tento Open Source projekt patří pod firmu Sun Microsystems a v dnešní době je jeden z nejpoužívanějších vývojových prostředí pro programovací jazyk Java. Pro použití tohoto prostředí jsem se zejména rozhodl kvůli automatickému generování grafických komponent, kterých bude zajisté nemálo.

4.2.1 Jazyk Java

Java je objektově orientovaný programovací jazyk, který vyvinula firma Sun Microsystems a představila 23. května 1995. Java je vyspělý programovací jazyk, obsahující všechny vlastnosti, které jsou vyžadovány v moderním programování, od modularity programu, řídících konstrukcí, přes silnou typovou kontrolu, multithreading, ošetření výjimek, správu paměti, silnou podporu pro databáze, XML a síťové operace.

K jejím výhodám patří zejména multiplatformita, robustnost, škálovatelnost a vysoká bezpečnost, která jí profituje pro používání na kritické aplikace na mainfraimových počítačích. I když základní vývojové prostředí obsahuje pouze řádkový překladač, existuje mnoho vývojových nástrojů a rozšíření dalších firem autorů. [10]

Obsah Javy však nelze omezit jen na výčet jejích příkazů. Java je především silně objektová, což umožňuje v ní modelovat, vytvářet, používat a rozšiřovat rozsáhlé knihovny a systémy. Právě objektově je třeba myslet ne jen při psaní programu, ale již při návrhu a analýze. Pro tyto účely byl vytvořen jazyk UML. [10]

Nejběžnějším způsobem ukládání dat je zápis do databází. Mezi nejpoužívanější databázové servery patří MySQL, který je volně ke stažení. Připojení k databázi nám zajistí aplikační rozhraní JDBC (Java Database Connectivity), které umožňuje aplikacím jazyka Java přistupovat k datům, která jsou strukturována do tabulek. Primárním zdrojem dat, ke kterému pomocí JDBC přistupujeme, jsou data v relační databázi.

4.2.2 MySQL

MySQL je databázový Open Source systém, vytvořený švédskou firmou MySQL AB. Jeho hlavními autory jsou Michael „Monty“ Widenius a David Axmark. V roce 2008 společnost Sun Microsystems koupila společnost MySQL AB a tak se MySQL stává dalším Open Source softwarem od firmy Sun Microsystems.

MySQL je multiplatformní databáze. Komunikace s ní probíhá pomocí jazyka SQL. Podobně jako u ostatních SQL databází se jedná o dialekt tohoto jazyka s některými rozšířeními. Pro svou snadnou implementovatelnost (lze jej instalovat na Linux, MS Windows, ale i další operační systémy), výkon a především díky tomu, že se jedná o volně šiřitelný software, má vysoký podíl na v současné době používaných databázích. Velmi oblíbená a často nasazovaná je kombinace MySQL, PHP a Apache jako základní software webového serveru.

MySQL bylo od počátku optimalizováno především na rychlost, a to i za cenu některých zjednodušení: má jen jednoduché způsoby zálohování, a až donedávna nepodporovalo pohledy, triggery, a uložené procedury. Tyto vlastnosti jsou doplňovány teprve v posledních letech, kdy začaly nejčastějším uživatelům produktu poněkud scházet. [11]

4.2.3 Vývojové prostředí NetBeans IDE

NetBeans IDE je úspěšný Open Source projekt s velmi rozsáhlou uživatelskou základnou a rostoucí komunitou vývojářů po celém světě. Firma Sun Microsystems založila Open Source projekt NetBeans v červnu 2000 a je zároveň i hlavním sponzorem celého projektu. NetBeans jsou dostupné pro Windows, Linux, Mac OS X a Solaris. NetBeans IDE se velmi podobá Visual Studiu.

Vývojové prostředí NetBeans IDE je nástroj, pomocí kterého programátoři mohou psát, překládat a ladit aplikace. Vývojové prostředí je vytvářeno v jazyce Java - ale může podporovat jakýkoliv programovací jazyk. Kromě toho také existuje velké množství modulů, které toto vývojové prostředí rozšiřují. Vývojové prostředí NetBeans je bezplatně šířený produkt, který je možné používat bez jakýchkoliv omezení.

Mezi hlavní výhody, které mě přiměly použít NetBeans, patří automatické generování grafických komponent, nabídka procedur či funkcí při práci s objekty, jednoduché spouštění programu a nástroje pro spolupráci s databází MySQL. Přestože jsem začátečník v užívání jazyku Java, pomůže mi v začátcích jednoduše generovat grafické komponenty a vytvářet celý systém.

5 Realizace nového systému

Implementační prostřední Netbeans IDE je k uživateli velmi přívětivé a každý začátečník může již po pár minutách vytvořit svůj první program „HELLO WORLD“. První krok, který jsem musel udělat, byl zprovoznění MySQL Servru jako localhost. Open-source software lze stáhnout přímo ze stránek výrobce (http://www.sun.com) a při instalování jsme vyzváni k zadání uživatelského jména, hesla a dalších nastavení pro databázi. Součástí distribuce MySQL pod Windows je řádkový klient *mysql.exe*. Spouštěcí soubor se nachází v podadresáři …/bin instalace SQL serveru. Řádkový klient nám poskytuje nepřeberné množství funkcí pro práci s databází, a proto první skutečností, kterou jsem udělal, bylo vytvoření nové databáze a tabulek pomocí SQL skriptu. Vytvořil jsem 4 tabulky (ZAKAZNIK, ZBOZI, KATALOG, UZIVATEL), které obsahují atributy dle navrženého diagramu. Celá databáze již byla připravena k použití a tak mohlo začít programování nového systému.

Připojení k databázi byl prvním úkolem, který jsem musel vyřešit. Programovací jazyk Java poskytuje pár rozhraní pro přístup k databázím, z nichž nejpoužívanější je rozhraní JDBC.

5.1 Připojení k databázi a práce s MySQL

Java Database Connectivity (JDBC) je aplikační rozhraní, které umožňuje aplikacím jazyka Java přistupovat k datům, která jsou strukturována do tabulek. Primárním zdrojem dat, ke kterému pomocí JDBC přistupujeme, jsou data v relační databázi. Vztah JDBC a databáze je možné přirovnat ke vztahu mezi rozhraním a třídou. Díky tomu nabízí jednoduchý a konzistentní způsob práce s relační databází nezávisle na jejím typu. Konkrétní JDBC driver lze získat v podstatě pro jakýkoliv databázový systém. Pro připojení k MySQL jsem použil driver Connector/J, který lze stáhnout ze stránek výrobce. [10]

Základní použití JDBC je velmi jednoduché. Pomocí několika tříd a rozhraní lze dosáhnout veškeré základní funkcionality. Základem pro práci je rozhraní `java.sql.Connection`, `java.sql.Statement` a `java.sql.ResultSet`.

Použití se skládá z několika kroků:

1. Import potřebných tříd.
2. Načtení JDBC driveru.
3. Identifikace zdroje dat.
4. Alokace objektu Connection.
5. Alokace objektu Statement.
6. Provedení dotazu pomocí Statement objektu.
7. Získání data z vráceného objektu ResultSet.

8. Uzavření ResultSetu.
9. Uzavření objektů Statement a Connection.

Nejprve musíme načíst driver a to provedeme pomocí statické metody `forName` třídy `Class`, která reprezentuje ovladač připojení. Pro správnou funkčnost se musí zachytávat výjimky `ClassNotFoundException` a `InstantionException`.

```
Class.forName("com.mysql.jdbc.Driver");
```

Spojení s databází získáme pomocí metody `getConnection` třídy `DriverManager`. Jako parametry jí předáme URL databázového serveru (v našem případě `localhost`) spolu s číslem portu (obvykle `3306`), jméno databáze, přihlašovací jméno a heslo. Konkrétní příklad:

```
Connection con = DriverManager.getConnection(
            "jdbc:mysql://localhost:3306/testDB", "root", "pas");
```

Nakonec vytvoříme objekt pro manipulaci s databází třídy `Statement` a jsme připraveni k vlastním operacím s daty.

```
Statement stmt = con.createStatement();
```

Pro zaslání dotazu v jazyce SQL zavoláme funkci `stmt.executeQuery(String sql)`, která je metodou třídy `Statement`, a vrátí se objekt typu `ResultSet`, nyní využijeme jeho funkce. Nejdůležitější funkcí je `next()`, která přesouvá ukazatel na další řádek výsledku. Pro vyvolání dat využíváme metod, které začínají na "`get`", kde sloupce začínají číslem „1“. Příklad:

```
ResultSet rs = stmt.executeQuery("SELECT * FROM ZAKAZNIK");
rs.next() // přesunu se na první řádek výsledku
rs.getString("3") // získám řetězec z třetího sloupce tabulky
```

5.2 Migrace dat z původního systému

Databáze již byla vytvořena a tak jsem začal se vkládáním dat. Pro první kroky s databází by stačilo vkládat pokusné data přes řádkového klienta `mysql.exe` nebo data vkládat programově využitím třídy `Statement.` Tyto metody dokážou bez jakýchkoliv potíží snadno vložit nové data, ale po zvážení jsem musel začít s migrací dat z původního systému. Tabulka ZAKAZNIK obsahuje 35

atributů a jejich ruční vložení pro pokusné účely by zabralo delší dobu a navíc s migrací dat bych se musel dřív nebo později vypořádat.

Původní systém využívá pro ukládání dat formát souborů DBF. Tento formát je v dnešní době stále aktuální, avšak jsem se rozhodl pro databázi MySQL. Pokud bych zůstal u databáze využívající formát DBF, musel bych tuto databázi upravit, protože od doby, kdy byl systém vytvořen, uplynulo pár let a databázoví klienti prošli značným vývojem. Změny se dočkaly i atributy jednotlivých tabulek, které byly upraveny dle konkrétních požadavků. Byly zde vypuštěny zbytečné sloupce a nahrazeny jinými.

Migrace dat představuje načtení původní databáze a přenos do nové databáze v MySQL. Pro práci s *.dbf soubory existuje několik knihoven, avšak některé z nich jsou placené. Knihovna, která není komerční, se jmenuje JavaDBF a je zdarma ke stažení na stránce: http://sarovar.org/projects/javadbf. JavaDBF je knihovna pro čtení a zápis do databází typu Xbase (DBF). Po stažení musíme tuto knihovnu (javadbf.jar) přidat do systémové proměnné $CLASSPATH. JavaDBF se nachází v balíčku com.linuxense.javadbf a pro použití je nezbytné importovat tento balíček ve zdrojovém kódu. Základem pro práci je rozhraní DBFReader, jehož parametr při vytváření je vstupní stream (*.dbf soubor). Pro správnou funkčnost se musí zachytávat výjimka DBFException. Příklad použití:

```
InputStream inputStream  = new FileInputStream("file.dbf");
DBFReader reader = new DBFReader(inputStream);
Object []rowObjects;

while( (rowObjects = reader.nextRecord()) != null) {
   for( int i=0; i<rowObjects.length; i++) {
        // Vypíši na stdout všechny řádky tabulky
        System.out.println( rowObjects[i]);
    }
}
inputStream.close(); // uzavřu vstupní stream
```

Problém, se kterým jsem se při migraci setkal, byl kódování DB souborů. Systém FoxPro využíval pro kódování českých znaků v programu kódování ISO-8859-1. Pro migraci dat bylo nezbytné načíst data ve správném kódování a uložit je. Pokud jsem načetl data a zobrazil je na výstup, tak na první pohled jiné kódování nezpůsobovalo žádné změny. Potíže nastaly až při migraci dat s diakritikou. Česká diakritika nebyla zobrazena správně, jediné řešení bylo překódovat vstupní soubory nebo načítat data ze souboru se zvoleným kódováním a ukládat v jiném kódování. První

možnost, překódovat celé soubory, se zdála nejjednodušší, ale po několika nezdařilých pokusech jsem musel přistoupit na možnost druhou. Celé soubory se mi nedařilo překódovat anii s použitím nejrůznějších programů z internetu. Využil jsem tedy metody druhé a všechny řetězce z DB načítal v kódování ISO-8859-1 a ukládal v kódování CP852 (PC Latin 2). Toto řešení se ukázalo jako spolehlivé a tak všechny data z původní DB byly překódovány a uloženy. Jedinou nevýhodou se ukázala rychlost této změny, přece jenom se musí procházet každý řetězec. Při větším objemu dat se konverze provádí několik sekund, migrace dat bude prováděna jen z počátku nasazení nového systému, tudíž rychlost algoritmu není důležitá. Příklad konverze řetězce:

```
String transString = new String(dbString.getBytes("ISO-8859-1"),
          "cp852");
```

Vývojový systém FoxPro nedokázal v době návrhu systému uchovávat řetězce delší než 76 znaků, což na delší poznámku o zákazníkovi nestačilo. Proto byly vytvořeny další dva sloupce v DB, kde mohla být uchována další informace. Dohromady jsme tedy mohli pro poznámku použít 228 znaků, které byly rozděleny mezi 3 sloupce tabulky. V samotném použití programu jsme byli pouze omezeni délkou poznámky 228 znaků, rozdělení mezi 3 atributy v DB jsme nepoznali a tento problém byl řešen programově. Nová verze MySQL 5.0.3. přinesla hodně změn a s nimi i omezení délky řetězce. U datového typu `VARCHAR` je nyní až 65532 bytů a nejsou ořezány případné mezery na konci. Poznámka v nové databázi může být několikanásobně delší než v původním systému, avšak z požadavků uživatele jsem zvolil dostačující délku 400 znaků. Poznámka byla do nového systému jednoduše přenesena spojením tří sloupců poznámek z původní databáze.

5.3 Zálohování dat

Zálohování patří k nejdůležitějším úkonům, které by měl databázový systém provádět. I přesto, že se počítače neustále vyvíjejí a nabízejí důmyslnější možnosti ochrany, můžeme během vteřiny, přijít o všechno, co máme v počítači uloženo. Počítačové soubory se snadno poškodí. Jeden chybný krok, chyba v softwaru nebo nesprávné vypnutí stačí k jejich poškození a znečitelnění. Soubory také snadno podléhají virům, červům a trojským koňům. Tyto nebezpečné kódy mohou způsobit nenapravitelné škody. Ke ztrátě dat může dojít velmi jednoduše a kdykoliv. I malý incident může mít katastrofické následky.

Zálohování dat patří zpravidla mezi opomíjené činnosti. Ponechat zálohování zcela na uživateli je cesta k maléru. Průměrný uživatel zkrátka data zálohovat nebude, a pokud ano tak jen proto, že v minulosti již o nějaká ta data přišel. To samozřejmě neznamená, že bychom uživatelům neměli umožnit zálohování provést, spíše to znamená, že bychom se na to neměli v žádném případě spolehnout.

Požadavky na nový systém zahrnují automatické průběžné zálohování, aby se uživatel o tuto rutinu nemusel starat. MySQL poskytuje několik způsobů pro zálohování databáze, avšak nová verze MySQL umožňuje důmyslnou zálohu. Klasický příkaz `SELECT` obsahuje velmi milé rozšíření, které nám umožní přímo vyexportovat vybraná data. Výsledná množina dat se vloží do nového souboru, kde můžeme specifikovat jeho místo uložení. Je to velmi jednoduchá záloha a proces zálohování pomocí tohoto příkazu je dosti rychlý. Příklad zálohy pomocí SQL dotazu:

```
SELECT * FROM ZAKAZNIK INTO OUTFILE 'zakaznik.txt';
```

Vytvoří se nám soubor `zakaznik.txt`, který obsahuje veškerá data z tabulky ZAKAZNIK. Tento způsob zálohy ukládá pouze samotná data a nikoliv strukturu dat. Nevýhodou či výhodou je, že výsledný soubor je čitelný člověkem, není tudíž nijak šifrován. Z tohoto důvodu vznikl další požadavek na systém, aby v programu byla přidána možnost tyto data šifrovat či ponechat nešifrované. Uživatel si tedy bude moci zvolit „otevřenost“ těchto dat.

Program bude dle požadavků provádět automatickou zálohu dat do určeného adresáře a to při každém ukončení programu. Pokud bude existovat v adresáři starší záloha, bude automaticky smazána a nahrazena zálohou novou. Jelikož MySQL neumožňuje soubory šifrovat přímo při vytvoření, jsou soubory v nezašifrované podobě. Tyto soubory načtu, zašifruji, zašifrované data uložím do jiného souboru a nezašifrovaný soubor smažu. Smazaný soubor se nebude uchovávat v „koši“ systému Windows. Touto metodou docílím toho, že zálohované data budou šifrované. Vedle automatické zálohy bude moci provádět zálohu či obnovu i sám uživatel, který si bude moci specifikovat místo uložení zálohy.

5.4 Bezpečnost uchovávaných dat

Bezpečnost dat ve formě řízení přístupu je v dnešní době velice důležitá. Přesto je v praxi význam ochrany dat podceňován a lidé se uchylují ke slepé víře v schopnosti jim dostupných technologií. Reálná hodnota dat bývá jejich vlastníkem doceněna až při jejich neautorizovaném využití, zničení nebo momentální, či trvalé nedostupnosti. Požadavky uživatele na nový systém zahrnují i tuto oblast: „*zamezení nežádoucího přístupu k datům i k zálohovaným souborům*“.

5.4.1 Bezpečnost dat v MySQL

Veškeré data, se kterými pracujeme v systému, jsou uchována v databázi MySQL. Přístup k databázovému serveru je zabezpečen pomocí přístupového hesla, které se může nastavit při instalování serveru, později pomocí řádkového klienta *mysql.exe* nebo použitím podpůrných aplikací pro práci s MySQL. Nastavení hesla pro přístup k serveru není povinné, avšak pro zabezpečení přístupu k datům je to nutnou podmínkou. Uživatel si může zvolit libovolné heslo, ovšem měl by brát

v úvahu obecné zásady pro vytvoření přístupového hesla. Nový systém musí uchovávat heslo pro připojení k databázi, zadávání hesla při každém spuštění programu by bylo velice nepraktické. Heslo je šifrováno algoritmem AES, u kterého doposud není znám žádný případ prolomení. Šifra je uchována v konfiguračním souboru programu. Při spuštění programu je tato šifra dekódována a heslo je použito pro připojení k databázi.

Data se v databázi uchovávají ve formě souborů, které jsou v případě lokálního serveru uloženy v podadresářích instalovaného serveru. Přístup k datům (souborům) je omezen dle přístupových práv. Všechny data (soubory) jsou automaticky šifrována a jejich změny či čtení jsou pouze v režii MySQL. Pokud uživatel uchovává důvěrné data, tak se nemusí o bezpečnost těchto dat obávat, vše zajišťuje server.

5.4.2 Přístup do programu

Původní systém vyžadoval, pro přístup do programu, zadání hesla. Pokud mají být data v systému zabezpečena proti neoprávněným osobám, musí být vstup do programu zabezpečen. Zadání hesla je jeden z nejpoužívanějších způsobů, jak povolit vstup do programu pouze oprávněným osobám. Tento způsob jsem použil i v novém systému a uživatel bez znalosti hesla se do programu nedostane. Heslo pro porovnání je uchováno v databázi v tabulce UZIVATEL. Hesla se většinou neukládají v otevřené podobě, ale používají se různé šifrovací algoritmy a ukládají se jejich šifry. Databázový server MySQL disponuje řadou funkcí pro šifrování dat. V novém systému jsem se rozhodl použít šifrovací algoritmus SHA-1, který dnes patří mezi nejpoužívanější hashovací funkce. SHA-1 vytvoří obraz zprávy dlouhý 160 bitů, což vyjadřuje hexadecimální číslo o délce 40 znaků. Heslo pro ověření je uchováno v šifrované podobě v tabulce. Ověření vstupního hesla je velice jednoduché. Zadané heslo je zašifrováno hashovaní funkcí SHA-1 a porovnáno s šifrou v tabulce UZIVATEL. Pokud se hexadecimální číslo shoduje, bylo zadáno správné heslo a uživateli je povolen přístup do programu. Ukázka šifrování pomocí SQL dotazu:

```
mysql> SELECT SHA1('abc');
       -> 'a9993e364706816aba3e25717850c26c9cd0d89d' // šifra
```

5.4.3 Šifrování zálohovaných dat

Nový systém bude provádět automatické zálohování dat nebo uživatel může sám zálohovat data v programu. Potup zálohování byl popsán v kapitole 5.3. Pomocí jednoduchého příkazu, uvedeného v kapitole 5.3, se vytváří záloha celé tabulky, avšak data jsou čitelná člověkem. Data obsahují citlivé informace, které musí být chráněny před odcizením nebo jiným zneužitím. Nejpoužívanější formou je šifrování, kdy útočník získá jen nesmyslnou změť bitů a bajtů.

Pro šifrování dat se používají různé algoritmy (AES, DES, Blowfish, 3DES, Serpent, atd.) a v programu jsem si zvolil AES (Advanced Encryption Standard), protože je jeden z nejpoužívanějších. AES je schválený standard, amerického úřadu pro standardizaci. Šifra využívá symetrického klíče, tudíž využívá pro šifrování i dešifrování stejný klíč. Klíč může mít délku 128, 196 nebo 256 bitů, hlavní výhodou algoritmu je velká rychlost a dá se použít pro šifrování velkého objemu dat.

Postup při šifrování zálohy:

1. Inicializace šifrovacího algoritmu.
2. Soubor zálohy je postupně načítán programem.
3. Program vytvoří nový, prázdný soubor pro ukládání šifry.
4. Data jsou šifrována a ukládána do nového souboru.
5. Původní soubor zálohy je smazán a zůstane pouze šifrovaný soubor.

Postup při obnovení zálohy:

1. Inicializace šifrovacího algoritmu.
2. Šifrovaný soubor zálohy je postupně načítán.
3. Program vytvoří nový, prázdný soubor pro ukládání dešifrovaných dat.
4. Data jsou dešifrována a ukládána do nového souboru.
5. Všechny data v tabulce databáze jsou smazána.
   (např.: `DELETE FROM ZAKAZNIK`)
6. Tabulka je naplněna daty ze zálohy.
   (např.: `LOAD DATA INFILE 'ZAKAZNIK.TXT' INTO TABLE ZAKAZNIK`)
7. Dešifrovaný soubor je smazán.

5.5 Konfigurační soubor programu

Počítačové programy využívají pro vlastní nastavení konfigurační soubory. V dnešní době se používají různé typy souborů, které jsou většinou typické pro konkrétní aplikace. Hlavním úkolem souborů je uchovávat data potřebná pro nastavení spouštěného programu. Ve většině konfiguračních souborů jsou data strukturována v podobě `<key>=<value>`, kde `key` představuje název proměnné a `value` představuje hodnotu v textové podobě.

Nový systém musí uchovávat nastavení pro samotné spuštění programu a uživatelem definované nastavení. Mezi hlavní proměnné patří: heslo pro přístup k databázi, datum poslední zálohy, absolutní cesta poslední zálohy (definovaná uživatelem), absolutní cesta poslední obnovy

(definovaná uživatelem), absolutní cesta automatické zálohy, údaje uživatele, atd. Všechny tyto údaje jsou uloženy v konfiguračním souboru, který má příponu `properties`. Programovací jazyk Java využívá soubory `properties` jako konfigurační soubory. Získávání a ukládání hodnot je velice jednoduché. Konkrétní příklad použití:

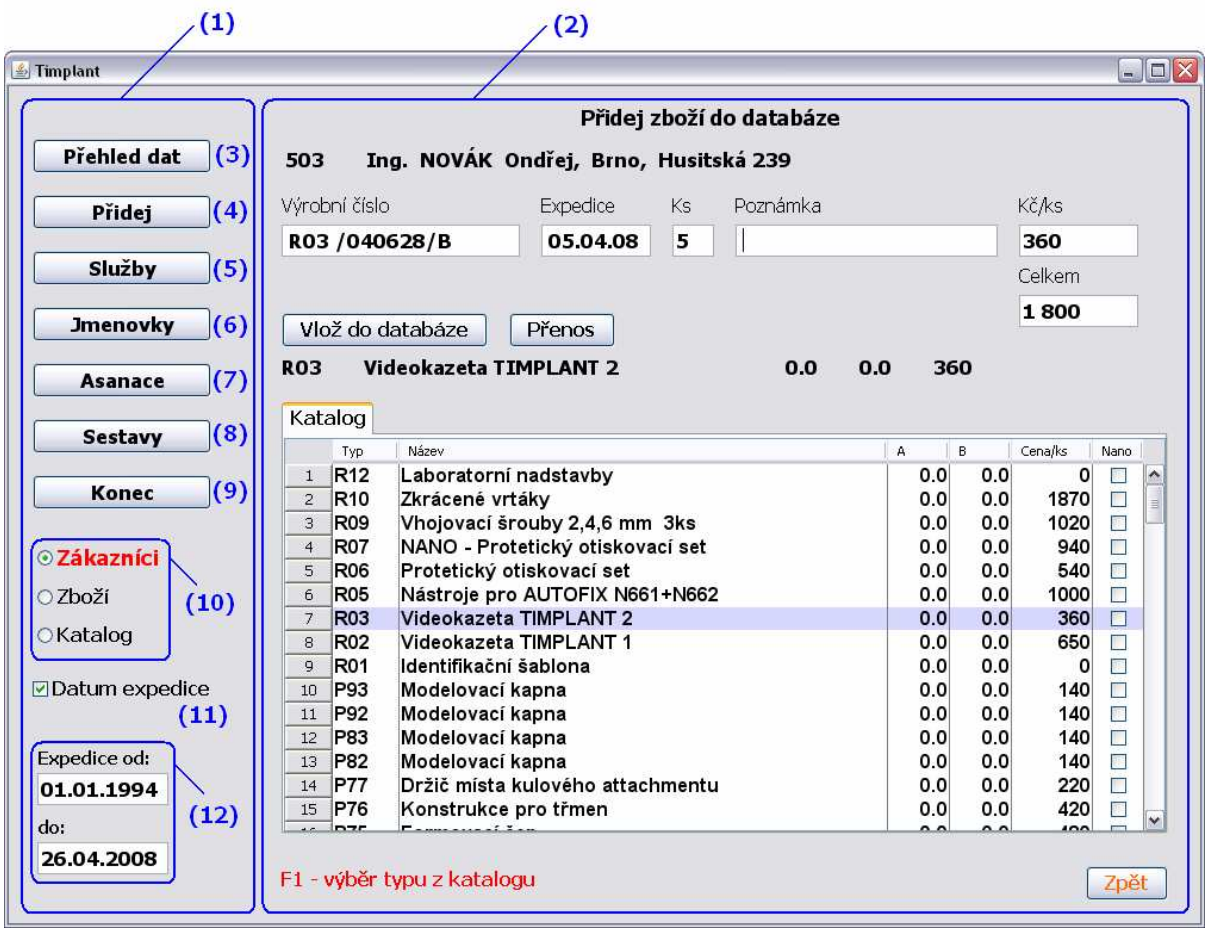
```
Properties configFile = new Properties();
// načtu konf. soubor "my_config.properties"
configFile.load(this.getClass().getClassLoader().getResourceAsSt
   ream("/my_config.properties"));

// uložím do souboru novou hodnotu s klíčem "key"
configFile.setProperty("key","value");

// načtu hodnotu ze souboru, kde klíč = "key"
some_var = configFile.getProperty("key");
```

5.6 Uživatelské prostředí nového systému

Požadavky na nový systém zahrnovaly zachování původního ovládání programu a celkovou strukturu. Původní systém byl postupně vyvíjen několik let, způsob ovládání programu a práce s databází byla za ty léta propracována do uživatelsky přívětivé podoby. Nový systém se tedy odráží od systému původního. Základem nového systému je hlavní okno (frame), na které jsou skládány další komponenty. Levou část okna představuje hlavní menu a zbytek okna tvoří panel, kde se zobrazují volby uživatele. Všechny GUI komponenty se umísťují do tzv. *kontejnerů* (frame, panel, dialog, label, buton, atd.). Umístění jednotlivých komponent v kontejnerové komponentě má na starosti tzv. *layout manager*. Je to objekt třídy `java.awt.LayoutManager`, který je zodpovědný za rozmísťování a zobrazování komponent ukládaných do objektů typu kontejner [13]. Hlavní panel, kde se zobrazují volby uživatele, používá *layout manager* s názvem *CardLayout*. Toto je zvláštní manager, protože má komponenty uspořádané do bloků a v jednu chvíli je viditelný pouze jeden blok. Jeden blok v mém případě představuje jeden panel, kde v jednu chvíli zobrazuji pouze jeden. V programu se může vytvořit až přes 20 panelů a každé zobrazení panelu předchází výběru z hlavního menu. Pokud chceme např. přidat nového zákazníka, klikneme na tlačítko přidat (musí být zaškrtnuta databáze zákazník), původní panel je odstraněn a je nahrazen panelem novým. V tom spočívá celé zobrazení programu.

5.6.1 Hlavní bloky systému

Jak již bylo zmíněno, hlavní okno je složeno z 2 panelů Obrázek 5.1 Ukázka nového systém, kde na pravý panel Obrázek 5.1 Ukázka nového systém jsou skládány panely další. Celý systém zahrnuje nepřeberné množství databázových funkcí, které patří ke standardům každého databázového programu, a proto nebudu všechny volby podrobně popisovat. Zaměřím se na hlavní bloky, kde popíšu jejich funkčnost. Pro představu nového systému nám poslouží následující obrázek (Obrázek 5.1 Ukázka nového systém



Obrázek 5.1 Ukázka nového systému

5.6.1.1 Popis prostředí

(1) **Hlavní menu** - Představuje jeden panel, kde uživateli nabízí nejrůznější volby s databází.

(2) **Hlavní panel** - Skládá se z jednoho panelu, na který jsou umísťovány panely další. Vždy je zobrazen pouze jeden panel, který závisí na uživatelské volbě. Pokud uživatel klikne na nějaké tlačítko z hlavního menu, právě v tomto panelu se zobrazí příslušné data.

(3) **Přehled dat** - Zobrazení v programu záleží na volbě databázové tabulky z hlavního menu (10).

a. Zákazník – Zobrazuje přehled zákazníků. Umožňuje zobrazit náhled zákazníka, upravovat zákazníka, zobrazit jeho odebrané zboží, náhled zboží, přidat zboží a další volby, které souvisí s příslušným panelem.
b. Zboží – Znázorňuje celkový přehled odebraného zboží v tabulce a k němu příslušné volby.
c. Katalog – Zachycuje v tabulce přehled všech výrobků z katalogu a k němu příslušné volby.

(4) **Přidej** - Uživatel může přidat do databáze nové data, výběr závisí na volbě databázové tabulky (10).

(5) **Služby** - Zobrazený panel umožňuje volbu z pěti služeb: záloha či obnova databáze, migrace dat, změna hesla, údaje o uživateli a kalendář narozenin.

(6) **Jmenovky** - Tisk vybraných parametrů z adresných údajů zákazníka.

(7) **Asanace** - Data v databázi jsou chráněna proti nechtěnému smazání. V programu máme možnost smazání označeného řádku tabulky, ale tento řádek není ve skutečnosti smazán a je pouze zvýrazněn červenou barvou. Vymazání označených řádků proběhne právě touto volbou.

(8) **Sestavy** - Tisk sestav, které jsou předem definovány uživatelem.

(9) **Konec** - Ukončení programu. Zavření systému předchází automatická záloha celé databáze (20).

(10) **Výběr tabulky** - Volba databázové tabulky souvisí s body (3) a (4). Pokud chceme zobrazit přehled dat či přidat data nová, musíme mít zvolenu příslušnou tabulku.

(11) **Datum expedice** - Všechno zboží zobrazené v přehledu dat je filtrováno datem expedice (12). Pokud není položka zatržena, zobrazuje se všechno zboží bez ohledu a datum expedice.

(12) **Expedice** - Volba data expedice souvisí se zatržením data expedice (11). V přehledech zboží se nám zobrazí pouze zboží, které odpovídá dané době.

5.7 Nasazení systému do praxe

Při implementaci nového systému jsem se snažil o jeho rychlé nasazení do praxe, aby mohl být testován v provozu a aby uživatel mohl klást požadavky na úpravu. V polovině března 2008 byl program vytvořen do fáze, kdy byla splněna základní funkčnost programu, a proto jsem systém nasadil k testování do firmy. Uživatelům z firmy se systém na první pohled líbil, avšak některé volby a postupy při práci s ním byly uživateli kritizovány, protože byli zvyklí na starý systém, u něhož měli zažitý způsob ovládání. Požadavků na úpravu nebylo málo, a tak jsem místo přidělávání nových funkcí musel upravovat již samotný testovací systém. V praxi bývá obvyklé, že zpracování programátora se odlišuje od praktického používání uživatele. Během měsíce dubna 2008 byl systém doplněn o nové funkce a požadavky. Poté byl znova testován. Situace s nasazením nového systému byla zase podobná. Vznikly nové požadavky na úpravu, které jsem musel přijmout a systém upravit.

Nový systém byl testován zadáváním, úpravou, přidáváním, mazáním, exportováním položek do databáze a byly testovány všechny funkce v programu. Současně byly stejné úkony prováděny v původním prostředí a jejich výsledky byly porovnávány. Tento způsob provádění stejných úkonů v obou systémech a kontrola jejich výstupů byla nezbytná pro ověření správné funkčnosti nového programu. Původní systém Tilat je využíván asi od roku 1995 a vzhledem k tomu, že je povinností firmy archivovat každý expedovaný výrobek včetně jeho charakteristických údajů, dosahuje objem dat značnou velikost. Data z původního programu byla přenesena do nového pomocí volby *migrace dat* (5.2). Použití aktuálních dat z původního systému bylo nutné zachovat pro kontrolu správných výstupů nového systému.

Bylo zřejmé, že uživatelský pohled na vyvíjený software podnítil nové myšlenky a požadavky, které nebyly v minulosti akceptovány pravděpodobně z důvodu obtížných úprav, které by mohly mít za následek celkovou nespolehlivost systému. Při návštěvách firmy bylo evidentní, že původní program Tilat je nejčastěji používaným software firmy. Proto byl při veškerých konzultacích kladen velký důraz na jeho spolehlivost a předpokládaná doba paralelního testování byla odhadnuta na cca 3 měsíce. V této době by měla být ověřena veškerá funkcionalita systému a následně porovnána s původním systémem. Bylo dohodnuto, že funkce, které jsou používány jen asi jedenkrát za rok (roční výstupy pro účetní uzávěrku apod.), budou simulovány a odladěny.

Závěr

Tato bakalářská práce se zabývá vytvořením nového databázového systému pro firmu Timplant s využitím volně šiřitelného software. Téma jsem si zvolil proto, abych získal praktické zkušenosti ve vytváření nových databázových systémů a jejich nasazení. Hlavním přínosem pro mě byla praktická realizace systému, kde byly kladeny požadavky a já jsem se s nimi musel z programátorského hlediska vypořádat. V některých případech vznikaly ze zdánlivě jednoduchých požadavků programátorsky náročné úlohy a problémy při implementaci jsem musel řešit po svém. Tato práce pro mě byla velikým přínosem, zvláště pak při spolupráci s koncovými uživateli. Pro realizaci systému jsem si vybral programovací jazyk Java, protože se stává stále populárnějším a jeho znalost je nespornou výhodou pro přijetí do budoucího zaměstnání.

Uživatelé předpokládají, že bude nezbytné tento software v budoucnu neustále modifikovat. Z toho jsem také usoudil, že pracovníci firmy jsou s novým software spokojeni nejen z důvodu zvýšeného komfortu obsluhy, ale také opticky je nový systém údajně značně přehlednější a „lidštější“.

V průběhu práce vznikaly nové nápady a myšlenky na rozšíření systému a to nejen s ohledem na připravované změny ISO norem, ale také s ohledem na možnost alternativní úpravy archivace dat. Jednou z mnoha inovačních myšlenek byla možná příprava systému pro síťovou verzi v souvislosti s požadavkem na nižší úroveň znalostí počítačové problematiky obsluhujícího personálu a s tím snížit riziko možnosti poškození dat. Za zmínku stojí také budoucí rozšíření o nově vyvinutý implantační systém Nanoimplant, u kterého se předpokládá stejné, či dokonce širší indikační použití jako u stávajícího systému, avšak jeho evidence a výstupy bude potřeba v budoucnu oddělit [13]. Nové požadavky, které nebyly předmětem zadání a vznikly až v průběhu testování, budou uživateli sepsány a mohou být předmětem rozšíření programu.

Při ladění systému jsem byl nucen nahlédnout do veškerých podkladů a seznámit se se zajímavými detaily činnosti firmy. Na základě oboustranné dohody jsem byl požádán o naprostou mlčenlivost o údajích, které nesmí být v žádném případě poskytnuty třetí osobě. Z tohoto důvodu jsem mohl pracovat jen na velmi malé části souboru dat, se kterou jsem prováděl ladění systému. Na celém objemu dat mi nebylo umožněno mimo prostory firmy pracovat.

Celkově byla tvorba této bakalářské práce velice zajímavým seznámením s moderními volně šiřitelnými technologiemi, které pro mě budou do budoucna zcela jistě i velmi přínosnou záležitostí. Jazyk Java a databáze MySQL prochází velmi rychlým vývojem a jejich používání v oblasti tvorby informačních systému neustále narůstá a vše nasvědčuje tomu, že tento trend bude i nadále pokračovat.

Literatura

[1] Nařízení Vlády České republiky č. 336/2004 Sb.

[2] Norma ČSN EN ISO 9001:2001

[3] Norma ČSN EN ISO 13485:2003

[4] Dluhoš, L.: Příručka jakosti dle ČSN EN ISO 9001/2001 a ČSN EN ISO 13485/2003, Timplant®, Ostrava, 2005, s. 26

[5] FoxPro [online], poslední aktualizace 4. dubna 2008, Wikipedie. Dokument dostupný na URL http://cs.wikipedia.org/wiki/FoxPro (duben 2008).

[6] Zendulka J., Rudolfová I.: Databázové systémy, Brno, 2006, Vysoké učení technické v Brně. Dokument dostupný na URL http://www.fit.vutbr.cz/study/courses/IDS/private/IDS_predn.pdf (únor 2008).

[7] Stein, R.: Návrh aplikací v jazyce UML – Unified Modeling Language, 2003. Dokument dostupný na URL http://interval.cz/clanky/navrh-aplikaci-v-jazyce-uml-unified-modeling-language (únor 2008).

[8] Beneš, M.: Přehled OO notací a metodik. Dokument dostupný na URL http://objekty.vse.cz/Objekty/MetodikyANotace#M2 (únor 2008).

[9] Šenk, Z.: Technologie pro perzistenci objektů v Javě, [Diplomová práce], Vysoké učení technické v Brně.

[10] Hatina, P.: Programování v jazyku Java (1) – Úvod, 2004. Dokument dostupný na URL http://www.linuxsoft.cz/article.php?id_article=244 (březen 2008).

[11] MySQL [online], poslední aktualizace 30. ledna 2008, Wikipedie. Dokument dostupný na URL http://cs.wikipedia.org/wiki/MySQL (březen 2008).

[12] Herout, P.: Java, grafické uživatelské prostředí a čeština, ISBN 80-72332-237-0, Kopp, 2006.

[13] Arnold, C., Hrušák, D., Dluhoš, L.: Nanoimplantáty – vlastnosti a indikace. StomaTeam CZ, 2007, ISSN1214-147X, s. 26-30.

Seznam příloh

Příloha 1. Programová dokumentace generovaná nástrojem JavaDoc viz přiložené CD

Příloha 2. Zdrojové soubory viz přiložené CD

Příloha 3. CD