TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Univerzální servisní databáze výrobků

Universal products database for service

Bakalářská práce

Autor: **Martin Paroubek**
Vedoucí práce: Ing. Tomáš Martinec, Ph.D.
Konzultant: Ing. Přemysl Svoboda

V Liberci 14. 5. 2012

ORIGINAL ZADANI PRACE
CISLOVANI ANO NEPÍŠE SE

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum

Podpis

Abstrakt

Tato bakalářská práce se zabývá vytvořením programu pro ukládání dat o výrobcích a servisních zásazích a rozšiřuje tak bakalářský projekt Servisní databáze výrobku. V úvodní části je stručně shrnut obsah projektu, náplň spočívající ve vytvoření funkčního programu a popsána motivace práce.

První kapitola je věnována uvedení do problematiky. Je definován pojem výrobek a tato práce je zařazena do odvětví Product Life Management (PLM) a pododvětví Maintenace, repair and overhaul (MRO). Obě tato odvětví jsou představena a u MRO je provedena rešerše existujících řešení.

V samostatné části jsou stanoveny cíle práce, jejichž součástí je také vytvoření funkčního programu umožňujícího hierarchicky ukládat výrobky a k těmto výrobkům přidávat vlastnosti, soubory a informace o servisních zásazích.

Poté je upraven návrh databáze z bakalářského projektu a je vytvořen návrh programu i uživatelského prostředí s využitím frameworku DockingFrames.

V předposlední obsahové kapitole je realizováno řešení v programovacím jazyce JAVA s využitím funkcí vývojového prostředí NetBeans IDE. Toto řešení je zdokumentováno uživatelským manuálem a dokumentací zdrojového kódu (javadoc).

Nakonec je ověřena funkčnost programu a nejrizikovější části jsou podrobeny zátěžovým testům s velikostí databáze až půl milionu řádků.

Přínosem této práce je umožnění uživatelům ukládat informace o výrobcích do jednoho přehledného místa.

Klíčová slova:

databáze, program, uživatelské rozhraní, výrobek, servisní zásahy

Abstract

This bachelor thesis deals with creation of a computer program designed to store data about products and service customers, thus advancing on the previous project, Products servicing database. In its introduction, a brief summary of the project is given along with the actual working program and a description of the motivation leading to the completion of this project.

The first chapter is introductory. The term "product" is defined and this project is assigned into the Product Life Management (PLM) branch and the Maintenance, repair and overhaul (MRO) subbranch. Both branches are briefly introduced and a research of already existing solutions is given in the case of MRO.

Objectives of the thesis are established in a separate part of the paper. Creation of a functional program that would allow for a hierarchic products database and the assignment of properties for these goods, as well as a database of service work performed on them was also a part of the thesis.

The database design from a previous project is then adjusted and a program and user interface design is created using the DockingFrames framework.

In the next chapter, a solution is implemented in the JAVA programming language, using NetBeans IDE. This solution is documented by the means of a user manual, as well as with a source code documentation (javadoc).

Finally, the functionality of the program is tested and the most high-risk parts are subjected to tests with databases up to half a million lines in size.

The main contribution of this project is enabling users to store information about products into a single, easy-to-use database.

Keywords:

database, program, user interface, goods, maintenance

Obsah

Seznam obrázků

Seznam symbolů, zkratek a termínů

ASP	Application service provider
CAx	Computer Aided Authoring
MRO	maintenance, repair and operations
MTBF	střední doba mezi selháními
MTBR	střední doba mezi výměnami
MTTB	střední doba do poruchy
PLM	Product Life Management

Úvod

Svět není dokonalý, co bylo vyrobeno se může porouchat. Pro takové situace je potřeba mít k dispozici základní informace o výrobku a jeho předchozích poruchách.

Tato práce navázala na bakalářský projekt Servisní databáze výrobků, jehož součástí byla rešerše embedded databází, navrhnutí databáze a vytvoření funkčního prototypu programu v Javě.

V této práci byla provedena rešerše existujících řešení. Program z bakalářského projektu byl přestrukturován: změnilo se uživatelské rozhraní, byla doplněna struktura databáze a k výslednému programu byla vytvořena dokumentace v podobě uživatelského manuálu a dokumentace zdrojového kódu. Důležitou součástí bylo dodefinování hierarchických vztahů mezi výrobky a jejich revizemi.

Motivací k této práci byla absence dostupného softwaru řešící skladování dat o výrobku spolu s hierarchickými vztahy a s ukládáním dat o servisních zásazích.

1 Uvedení do problematiky

V následujících kapitolách jsou definovány termíny používané v oblasti, do které se řadí tato práce. Dále jsou představeny nejvýznamnější firmy, jejich produkty a technologie.

1.1 Výrobek

Sääksvuori a Immonen [19] konstatovali, že výrobek je obvykle chápán jako něco hmotného, co může být vlastněno, vyměněno nebo distribuováno. A to samé je dnes možné i s nehmotnými věcmi, například softwarem, algoritmem, vzorcem nebo informací, které jsou také považovány za výrobky.

1.2 Product Life Management (PLM)

Pojem PLM není jednotně definován, proto byla jako definice zvolena odpověď v diskusi Tomáše Svobody obchodního ředitele Siemens PLM Software [22]:

> *„Hlavní myšlenkou PLM je spojení většiny informací o výrobku do jednoho místa, do jednoho centrálního úložiště dat, odkud jsou informace snadno dostupné a hlavně odkud jsou řízeny jednotlivé procesy s produktem spojené. Podstatné je, že spravovaná data vznikají po celou dobu životního cyklu produktu, tedy od fáze úvah o uvedení nového výrobku, přes sběr požadavků na nový produkt, nebo poptávky, přes fáze nabídkové, kontraktační, předvýrobní, přípravy výroby až po vlastní předání do výroby, distribuci, podporu a poskytování servisu zákazníkovi, až třeba po fáze správného stažení výrobku z trhu.“*

Obrázek 1.1: Rozdělení PLM převzato z [23]

V článku od Homoly [8] bylo také řečeno, že PLM software nepředstavuje jeden program, ale soubor navzájem spolupracujících programů, kde každý program je určen pro jinou část životního cyklu výrobku. Tyto programy jsou obvykle stavěny individuálně na míru zákazníkovi, kde míra přizpůsobení nezřídka dosahuje až devadesáti procent rozsahu celého řešení. Přesto se lze i setkat s tzv. out-of-the-box aplikacemi, které jsou určeny pro menší podniky a jsou svým uživatelům k dispozici ihned po nainstalování a napojení na související software (např. CAD a ERP).

1.3 Prodejci PLM řešení

Podle studie Andorky [2] patří mezi nejlépe hodnocené distributory PLM systémů firmy Autodesk, Dassault-Systemes, PTC, Siemens PLM a Oracle. První čtyři zmínění vévodí části trhu CAx PLM (Computer Aided Authoring), která obsahuje software pro design, obrábění a simulaci. V odvětví procesní výroby (jídlo, nápoje, benzín a plyn) si vede nejlépe Oracle. Výrobky je potřeba v tomto odvětví rychle nahrazovat, z toho plynou kratší výrobní cykly. Siemens PLM a Dassault-System také vévodí v prestižní a vysoce konkurenční oblasti diskrétního PLM, do kterého řadíme letectví, automobilový průmysl, zdravotnictví, těžké stroje a stavbu lodí. Mezi další distributory PLM softwaru patří podle přehledu na stránce [16]: Aras Corp, Arena Solutions, Ingenuus Software, Metafore a SAP.

1.4 Maintenance, repair and operations

Stark [21] konstatoval, že jednou z větví PLM je MRO (maintenance, repair and operations), která sleduje stav produktu, jeho konfiguraci, servisní zásahy a verzi výrobku. Podle [12] MRO software může také obsahovat:

- konfiguraci tzv. bills of material, v němž je uložen soupis součástek, u kterých upravuje jejich stav na: doručeno, v údržbě a použito
- plánování logistiky, například nalezení kritické cesty na soupisu úkolů, které mají být provedeny
- plánování oprav
- spravování spouštění událostí
- spravování součástek, nástrojů a zásob vybavení
- historii údržeb, sériová čísla součástek, data o spolehlivosti MTBF (střední doba mezi selháními), MTTB (střední doba do poruchy), MTBR (střední doba mezi výměnami), dokumentaci údržeb, oprav a osvědčených postupů, dokumenty o záruce.

1.5 Metody a řešení MRO

Přestože MRO software je jen částí PLM, jedná se o komplexní systém, který je, podle zveřejněných dat z článku [14], většinou řešen technologií klient-server. Toto řešení je vhodné kvůli množství dat, které musí být sdíleny mezi jednotlivými uživateli.

1.5.1 Existující MRO řešení v letectví

MRO je stejně jako PLM řešeno především na míru zákazníkovi a vzhledem k různorodosti odvětví nelze software jednoduše porovnávat. Proto bylo vybráno jedno odvětví, ve kterém problematiku MRO řeší již delší dobu, a tím je letectví. Následující firmy byly vybrány dle počtu zákazníků uvedených v článku [14]. S výjimkou IBM, která stejně jako mnoho jiných firem tento údaj nezveřejňuje, a proto výběr nemusí odpovídat skutečné situaci.

Cimber Air Data

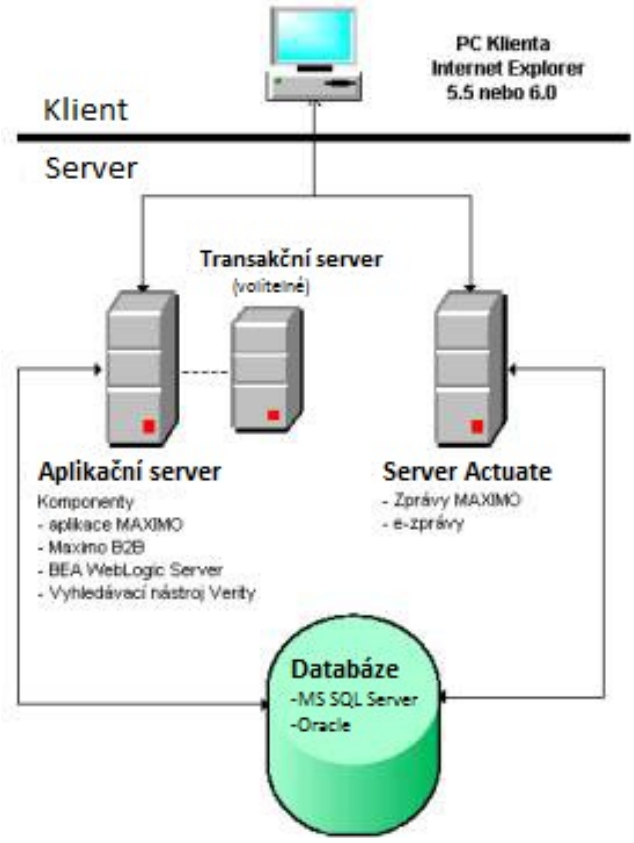
Podle [14] je Cimber Air Data dánská společnost s kancelářemi v Singapuru a Malajsii. Její program se nazývá AMICOS. Ten byl nejdříve implementován v letecké společnosti Cimber Air jako malý počítačový systém, pak se program rozvinul do klient-server modelu. Nyní je prodáván středně velkým aerolinkám. Jejich program má celou řadu modulů, aby splňoval požadavky kladené aerolinkami a MRO. Mezi tyto moduly patří: inženýrství, plánování, prodej, kontrola spolehlivosti a kvality, technické záznamy, nákup, kontrola cen, správa půjček a MRO (s moduly pro kontrolu ceny projektu a fakturaci). S 13 zaměstnanci se tato společnost řadí k těm nejmenším v oboru.

Pozn: Dne 3.5.2012 podala tato Cimber Sterling Group vlastnící Cimber Air Data návrh na prohlášení konkurzu.[4]

MRO Software (IBM)

IBM odkoupila MRO Software v roce 2006. Jejich produkt je znám pod názvem Maximo je kompletní nabídkou pro aerolinky nebo poskytovatele služeb a zahrnuje tak mnoho funkcí, že má svoji vlastní nezávislou uživatelskou základnu a internetovou diskusi pro podporu a konfiguraci.[14]

Podle českých oficiálních stránek [3] tato služba funguje na principu klient-server (viz Obrázek 1.2). Klient musí mít prohlížeč, pomocí kterého se připojí na aplikační server. Ten pak komunikuje s databází (MSSQL nebo Oracle) a volitelně i s transakčním serverem.



Obrázek 1.2: Architektura Maximo převzato z [3]

IFR

Podle [14] společnost IFR sídlí v Toulouse ve Francii. V roce 2006 měla 65 klientů z řad aerolinek, kterým poskytovala software AMASIS. Ten byl původně naprogramován v COBOL. A nyní je nabízen jako ASP (Application service provider) služba. Tudíž si klient pronajímá aplikaci jako službu a připojuje se k ní přes síť.

SWISS Software

Podle [14] je SWISS Software jeden z dominantních prodejců softwaru pro aerolinky. Tato společnost zaznamenala značný vzestup mezi rokem 2003 a 2008, kdy se mezi její klienty přidaly významné aerolinky, například Ryanair, easyJet a TUI Group. Společnost se začala zabývat problematikou v roce 1989, ale první zákazník si jejich software koupil až v roce 2002. Program, který vytvořili v jazyce JAVA se zabudovanou modulární konstrukcí, se nazývá Amos. Tato společnost prodává aplikaci výhradně na komerčním trhu, tudíž nemá žádné vojenské zákazníky.

Podle oficiálních stránek [13] patří mezi vytvořené moduly správa materiálu, strojírenství, plánování, produkce, kontrola údržby, údržba komponent, kontrola kvality, lidské zdroje, finanční správa a rozhraní.

TRAX

Podle [14] byla TRAX jednou z prvních společností nabízejících alternativu k existujícím těžkopádným řešením s centrálním počítačem. V roce 2008 měla firma 66 zákazníků. Mezi jejich prestižní zákazníky patří Alaska Airlines, United Airlines a jetBlue. Mnoho funkcí jejich programu bylo vyvinuto ve spojení se zákazníky, což umožnilo vysokou využitelnost programu. Mezi tyto funkce patří údržba letky, plánování, záruka kvality, správa a plánování materiálů, dodržování a vykazování auditu, technické záznamy on line, manuál distribuce, správa MRO a další.

1.5.2 Open Source alternativy

Najít open source alternativu vzhledem ke specifičnosti této práce je problematické. Nejdříve byla alternativa vyhledávána pomocí klíčových slov (např. MRO open source software apod.). Po neúspěchu byly prohledány záznamy na stahovacích serverech www.stahuj.cz a www.slunecnice.cz.

Existují univerzální programy umožňující jednoduše ukládat informace o výrobcích, ale přehledná správa servisních zásahů by v nich byla problematická. Během průběhu rešerše nebyla nalezena open source alternativa vyhovující alespoň částečně požadavkům práce.

1.5.3 Shrnutí podkapitoly

MRO software zahrnuje obrovské množství často přizpůsobitelných funkcí, které se mezi jednotlivými distributory liší. Informace o vnitřním fungování a architektuře nelze citovat anebo jsou minimální. V současné době je trendem architektura klient-server, avšak její uplatnění se také liší. Některé firmy používají jako klienta internetový prohlížeč, jiné vytváří vlastní program. Server komunikuje s databází a jeho další možnosti jsou mezi firmami různé. Mezi nejčastěji používané programovací jazyky patří .NET a JAVA. Vzhledem k vysoké míře přizpůsobitelnosti programů vznikají cenové nabídky až na základě potřeb jednotlivých zákazníků.

2 Cíle práce

Cílem této práce je vytvořit pseudo-MRO řešení pro hardwarové výrobky splňující následující požadavky:

- výrobky je možno ukládat do skupin a podskupin
- k výrobkům a skupinám je možné přidávat vlastnosti (vypovídající o možnostech nebo stavu výrobku, přičemž výrobky dědí vlastnosti od předků)
- k výrobkům lze definovat servisní zásahy
- k výrobkům lze přiřadit soubory
- z těchto údajů (viz čtyři předchozí body) lze vygenerovat výstupní sestavy
- tyto informace je možné exportovat v nějakém obvyklém formátu
- k ovládání programu musí sloužit grafické uživatelské rozhraní
- k uživatelskému rozhraní musí být vytvořen manuál
- zdrojové kódy programu musí být zdokumentovány.

3 Návrh řešení

V této kapitole byla zdůvodněna volba prostředků a byl vytvořen návrh řešení pro databázovou, programovou a souborovou část.

3.1 Databázová část

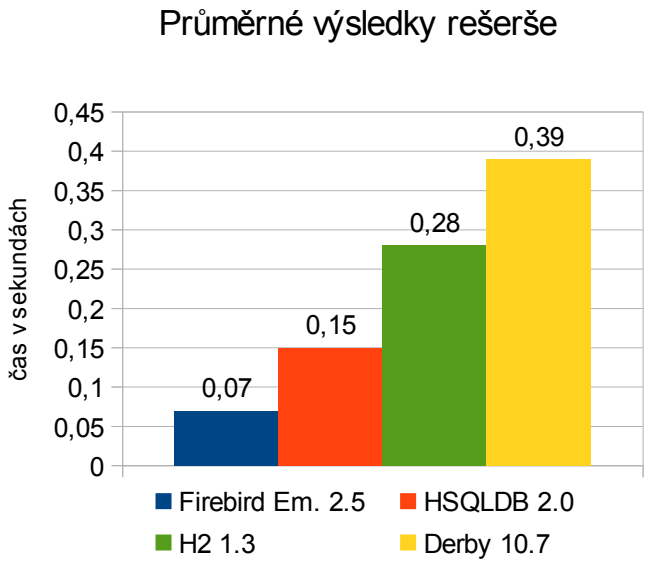
Během bakalářského projektu byly stanoveny podmínky pro výsledný program. V těchto podmínkách bylo také uvedeno, že program má běžet na localhost bez nutnosti instalace. Z tohoto důvodu byla jako úložiště zvolena embedded databáze, kterou Kolář [11] definoval takto:

> *„Embedded databáze tvoří přesný opak architektury klient-server. Místo aby využívaly pro přístup k datům vyhrazený databázový server, běží v adresovém prostoru aplikace, která je využívá“.*

3.1.1 Volba embedded databáze

Volba proběhla rešerší nekomerčních embedded databází během bakalářského projektu. Vybrané databáze (Firebird, HSQLDB, H2, Derby) byly podrobeny sadě testů obsahujících: výběr, upravování, mazání dat a spojování tabulek. V testovací tabulce bylo sedm sloupců, žádné relační závislosti a při nejnáročnějších testech obsahovala 17 tisíc řádků. Na základě zprůměrovaných výsledků všech testů dopadl nejlépe Firebird Embedded (viz Obrázek 3.1).

Na oficiálních stránkách Firebird databáze [1] je řečeno, že **Firebird** je relační databáze pracující s SQL příkazy. Tuto databázi lze provozovat na Linuxu, Windows a Unixových platformách a je šířena s open-source licencí IDPL (Initial Developer's Public Licence), která ukládá povinnost vývojářům zveřejnit modifikované zdrojové kódy pod stejnou licencí. Podrobněji je licence popsána na oficiálních stránkách a mimo jiné povoluje určení vlastní licence u nových modulů.



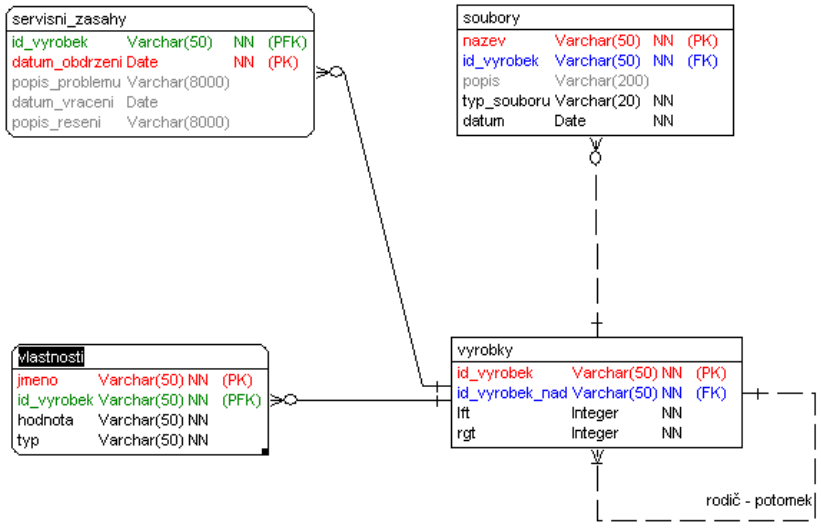
Obrázek 3.1: Výsledky rešerše bakalářského projektu

3.1.2 Návrh struktury databáze

První návrh byl vytvořen jako součást bakalářského projektu v programu CASE Studio 2, který umožňuje z vytvořeného diagramu vygenerovat SQL skript k vytvoření struktury databáze.

V této práci byla změněna tabulka vlastností a byl přidán sloupec *datum* k tabulce *soubory*, který umožňuje v aplikaci řadit soubory dle data přidání. U téže tabulky byl zjednodušen primární klíč z dvojice *nazev+id_vyrobek* na *nazev*, který musí být jedinečný, protože jsou soubory od všech výrobků ukládány do jedné složky. Také byla opravena referenční integrita tak, aby se změna v jedné tabulce promítla řetězově do dalších. Jako výchozí kódování bylo dodatečně zvoleno UTF8. Výsledný návrh je zobrazen na Obrázku 3.2.

Program CASE Studio 2 také vytvořil hlášení pro chyby vytvořené databází, a ty byly ve výsledném programu ponechány.



Obrázek 3.2: ERD diagram z programu CaseStudio 2

3.1.3 Tvoření hierarchie výrobků

Skupina může mít několik podskupin výrobků a přitom musí mít stejné možnosti jako výrobek, což vyžaduje umístění těchto dvou elementů do stejné tabulky, která bude odkazovat sama na sebe resp. na rodiče výrobku. V případě, že výrobek si je sám sebou rodičem, tak je výrobek nebo skupina kořenem stromu. K ulehčení práce s tímto modelem tabulky slouží sloupce *lft* a *rgt* označující obal dané skupiny nebo výrobku, resp. levé a pravé ohraničení. Výrobek se v databázi pozná tak, že má *rgt=lft+1*. Skupina má *rgt>lft+1* a vše mezi hodnotami *lft* a *rgt* jsou její potomci.

3.1.4 Tabulka vlastností

V první verzi měla tabulka dva sloupce hodnot (celočíselný a textový sloupec), kde právě jen jeden mohl být vyplněn. Takto postavená tabulka měla výhodu v možnosti použití řadících databázových funkcí. V případě rozšíření o nový typ vlastnosti (například datum) by ovšem bylo potřeba změnit program na několika místech a změnit strukturu databáze.

Další možností bylo mít tabulek více pro různé datové typy, tudíž by se v žádné tabulce nenacházel sloupec, který by byl nulový. Ale tímto řešením se připravíme o informaci, v jakém pořadí uživatel zadával vlastnosti. Pro uchování pořadí by bylo nutné vytvořit generátor a v každé tabulce přidat po jednom sloupci.

Zvolené řešení neumožňuje řazení dat vlastností pomocí databáze, protože ukládá všechny hodnoty jako textový řetězec a k identifikaci typu slouží sloupec *typ* (např. celé číslo nebo text). Další dva sloupce uchovávají odkaz na výrobek a název vlastnosti.

3.1.5 Tabulka souborů

Tato tabulka uchovává název souboru, který odpovídá jménu souboru v adresářové struktuře databáze. Dále uchovává odkaz na výrobek, ke kterému soubor patří, popis, typ souboru a datum, kdy byl soubor přidán. Aktuální datum je načteno pomocí SQL příkazu *NOW()*. Primárním klíčem tabulky je název souboru.

Z teoretického hlediska by mohlo více výrobků odkazovat na stejnojmenný soubor, ale nebylo by to praktické, protože by byla složitější adresářová struktura (každý výrobek by měl svůj adresář) anebo by se rozdělilo přidávání nových souborů na ty, které v adresářové struktuře už jsou a na ty, které tam ještě nejsou. V takovém případě by byla složitější funkčnost při mazání výrobku, kdy by bylo potřeba kontrolovat počet odkazů na jednotlivé soubory a na jejich základě je teprve mazat.

Pozn: Kvůli vybrané funkčnosti je doporučeno pojmenovávat soubory s předponou jména výrobku, kterou program sám předvyplňuje.

3.1.6 Tabulka servisních zásahů

Tato tabulka obsahuje odkaz na výrobek, kterého se servis týká, datum obdržení výrobku k provedení opravy, popis problému, datum vrácení a popis řešení. Oba popisy jsou typu *varchar* a jsou omezeny na osm tisíc znaků. Také by bylo možné využít datový typ BLOB, který ukládá data binárně a není u něj definováno omezení.

3.2 Část programová

V této části byla zdůvodněna volba vývojového prostředí a popsán návrh programu.

3.2.1 Volba vývojového prostředí

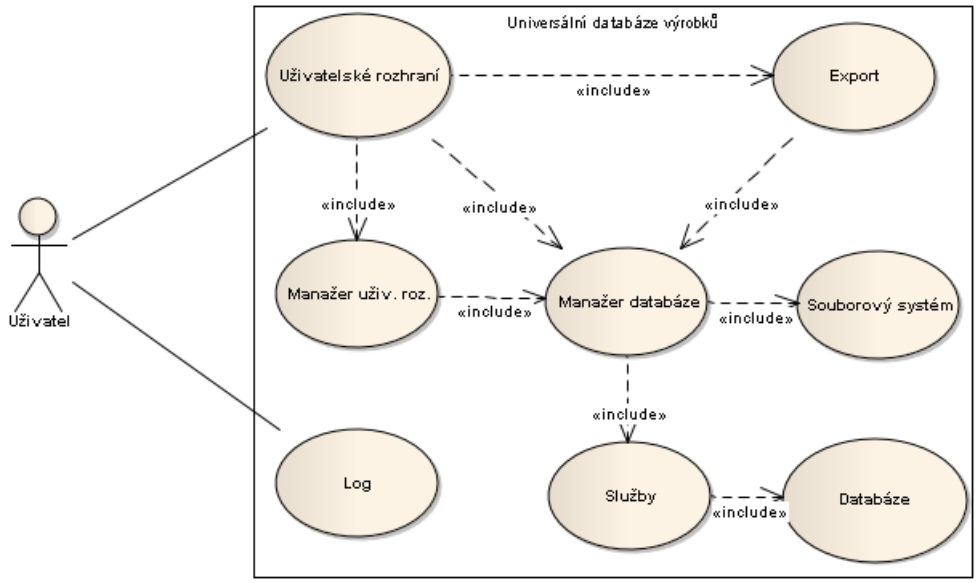
Během tvoření bakalářského projektu nedošlo k žádným problémům vyžadujícím změnu vývojového prostředí, proto byla práce dále vytvářena v Netbeans IDE.

Konkurence obsahuje podobné funkce, ale neobsahuje vestavěný Swing GUI Builder. Ten byl využit zejména pro vytvoření statických částí oken a pro jednoduchou správu vzhledů (např. GridBagLayout). Mezi další přednosti, které byly využity při vývoji aplikace je možnost sledovat změny kódu v jednotlivých třídách a grafické znázornění rozdílů mezi aktuálně uloženým stavem a předchozími stavy.

3.2.2 Návrh programu

Výstupem bakalářského projektu byl funkční prototyp programu, který byl navržen tak, že se veškerá problematika vyskytovala v jednom souboru s uživatelským rozhraním a jednotlivá dialogová okna byla vytvořena pomocí Swing GUI designer jako ostatní komponenty hlavního okna. Výsledná třída tak obsahovala tři tisíce řádků a bylo problematické se v ní orientovat.

Návrh programu byl vytvářen metodou přírůstku a probíhal postupným odštěpováním funkčních celků z rozměrné třídy hlavního okna a nabalováním nových funkcí. Po čase se stalo, že kód byl v některé z tříd nepřehledný anebo nevyhovující, a proto byl podle stejného principu upraven nebo rozdělen další soubor. Výsledek návrhu byl zobrazen v obrázku 3.3.



Obrázek 3.3: Use case diagram vytvořený v programu Enterprise architect

Stavebním kamenem programu je databáze, ve které je uložena většina dat zobrazovaných uživateli. Další data jsou uložena v souborovém systému. Na databázi jsou napojeny pouze služby postavené na návrhovém vzoru singleton. Ty mají za úkol převést řádky tabulek do objektů a naopak. Tento přístup je výhodný v tom, že se strukturou databáze se pracuje pouze v těchto službách a jinde se už není třeba starat o to, jaké sloupce má tabulka.

Na služby se napojují třídy manažeři databáze (též singleton instance), kteří zodpovídají za odchycení všech výjimek a také pracují se souborovým systémem, kvůli kterému se některé výjimky neodchytávají ve službách, protože v některých případech je při chybě programu nutné zotavení jak databáze, tak souborového systému. S manažerem databáze může komunikovat kterákoliv část programu.

Manažeři uživatelského rozhraní jsou mezistupněm mezi manažery a uživatelským rozhraním. Slouží k vyhnutí se duplicitě kódu v případech, kdy z různých tříd spouštíme stejný kód. Například, když přidáváme nový výrobek v horním menu nebo přes vyskakovací menu, tak se zavolá vždy tento manažer obstarávající zobrazení příslušného dialogu.

Většina informací pak přes tyto funkční celky putuje do uživatelského rozhraní. Chybová hlášení jsou zapsána do logu.

3.2.3 Použité knihovny

V programu bylo použito kromě JDK dalších pět knihoven. Připojení k databázi obstarává knihovna Jaybird (Jaybird-full-2.2.0.jar). Ke komunikaci s databází byla použita knihovna DbUtils (commons-dbutils-1.4.jar), která umožnila zjednodušení komunikace s JDBC. Vzniklé chyby ve všech částech programu jsou zapisovány do souboru pomocí knihovny LOG4J (log4j-1.-2.16.jar). Další knihovny zpříjemnily používání uživatelského rozhraní. Knihovna Docking Frames (dockingFramesCore.jar) umožnila částečně přizpůsobovat rozvržení aplikace (změna velikosti jednotlivých panelů, zavírání a překrývání částí) a knihovna Microba (microba-0.4.4.3.jar) umožnila použití kalendářů při obsluze servisních zásahů.

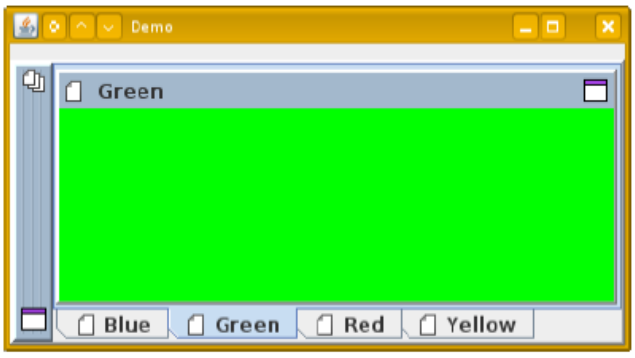
Licence:
Jaybird: LGPL [10]
DbUtils: Apache 2.0 [[17]
Docking Frames: LGPL 2.1 [5]
Microba: BSD [7]
LOG4J: Apache 2.0 [18]

3.2.4 Možnosti Docking Frames Core

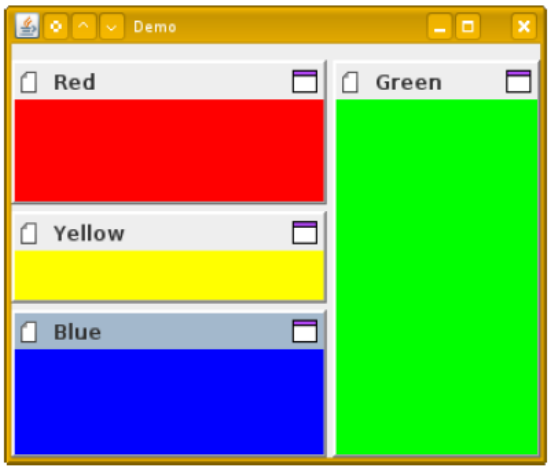
Podle dokumentace DockingFrames [20] je základní ideou mít jeden objekt, který spravuje celé rozhraní, jeden objekt pro každý posuvný panel a jeden objekt pro každé místo, kde se panel může nacházet.

Dockable [20] reprezentuje posuvný panel a také může obsahovat titulek a ikonu. Každý objekt tohoto typu může být posunut uživatelem na DockStation. Implementace může proběhnout pomocí interface Dockable a nebo jednodušeji použitím objektu DefaultDockable. Dockables se nemůžou přesouvat sami od sebe a potřebují DockStation k ukotvení. Vztah mezi DockStation a Dockable může být nejlépe popsán jako vztah mezi rodičem a potomkem.

Ve verzi Core existují čtyři různé druhy DockStation [20]. **StackDockStation** je organizována jako systém záložek, kde se pomocí tlačítek vybírá jeden potomek, který bude vidět. Framework automaticky vytvoří novou StackDockStation, kde je jeden Dockable přesunut nad druhý a naopak StackDockStation s jedním potomkem se automaticky nahrazují potomkem. Dalším druhem je **SplitDockStation**, u které jsou všichni její potomci vedle sebe. Uživatel kontroluje potomky tak, jako kdyby se stanice skládala z tabulky, jejímiž buňkami lze libovolně manipulovat. Vnitřně je stanice organizovaná jako strom, kde listy představují Dockable a uzly mezery mezi nimi. Mimoto stanice nabízí celoobrazový režim, kde jeden z potomků zabere celý prostor a ostatní jsou neviditelné.



Obrázek 3.4: StackDockStation se čtyřmi potomky. Převzato z [20]



Obrázek 3.5: SplitDockStation se čtyřmi potomky. Převzato z [20]

Další stanice jsou **FlapDockStation** a **ScreenDockStation** a jejich možnosti nejsou v práci využity.

DockController [20] drží prvky Dockable, DockStation a další elementy pohromadě. Většina úkolů neřídí DockController, ale jeden ze Sub-controller (například přesouvání oken je řízeno prvkem DockRelocator). Aplikace může mít víc než jeden DockController. Každý kontrolér má vlastní oblast, do které nezasahuje jiný. Většina aplikací si ale vystačí s jedním DockController.

DockController implementuje pouze základní funkcionalitu, zatímco **DockFrontend** [20] umožňuje programátorům přidávat nové vlastnosti. Také reprezentuje vrstvu před DockController a přidává sadu užitečných metod, například tlačítko na zavření Dockable a schopnost ukládat a načítat rozvržení prvků DockStation a Dockable. Inicializace se provede přidáním kořene, např. DockStation pomocí metody addRoot, do DockFrontend instance.

3.2.5 Uživatelské rozhraní (UI)

Původní uživatelské rozhraní bylo vytvořeno pomocí Swing GUI Builder, který je součástí NetBeans IDE. Všechny komponenty včetně dialogových oken byly v jedné třídě a to bylo pro další pokračování nepřehledné, proto bylo uživatelské rozhraní rozděleno na více částí (modulů), které reprezentují jednotlivé panely (seznam výrobků, seznam vlastností, obrázek výrobku, servisní zásahy, soubory, statistika) v hlavním okně a jiné funkční části (dialogová okna, vyskakovací menu). Toto rozdělení umožnilo jednoduchou implementaci variabilního prostředí pomocí knihovny Docking Frames ve verzi Core. Implementaci DockFrontend a SplitDockStation. Každý panel představuje jeden Dockable objekt.

3.2.6 Souborový systém

Souborovým systémem se rozumí nakládání programu s přidanými soubory. Umístění těchto souborů bylo zvoleno ve složce *Soubory*, které jsou ve stejném adresáři jako soubor s databází. Druhou možností bylo zvolit umístění přidaných souborů zvlášť do složek pojmenovaných dle výrobku, ke kterému náleží. Takové řešení by bylo více přehledné, ale neúsporné na velikost, neboť operační systémy si zpravidla vytvářejí v nových složkách neviditelné systémové soubory. Ke zlepšení přehlednosti prvního řešení bylo dodatečně doporučeno volit systematické pojmenování souborů, například s předponou výrobku.

4 Realizace řešení

Realizace probíhala ruku v ruce s návrhem programu. Během bakalářského projektu byl vytvořen funkční program, u kterého byla většina kódu nahromaděna v jednom souboru s hlavním oknem. Tato práce se snažila předejít obdobnému výsledku, a proto v ní byly postupně vytvořeny funkční celky.

4.1 Popis komponent

V této kapitole byla popsána funkčnost jednotlivých částí programu. Zdrojový kód se natolik změnil, že u většiny tříd nemělo smysl rozlišovat části z bakalářského projektu. Nejdříve byla popsána část databázová, která je základním prvkem programu. Pokračovalo se samostatně nečinnými částmi: exportem a souborovými nástroji. Potom byl popsán hlavní program, který vše spustí a načte uživatelské rozhraní, které bylo zařazeno nakonec.

Během rozvrhování uživatelských prvků byly hojně využívány layouty, které lze nastavit pomocí Swing GUI Designer. Nejčastěji užívaným byl GridBagLayout a GridLayout. Pro dynamické prvky uživatelského rozhraní byla rozvržení dopsána manuálně.

Všechna dialogová okna jsou spouštěna jako modální, tudíž se hlavní okno přesune do pozadí a je nepřístupné do doby než dialogové okno dokončí svou činnost.

4.1.1 Databázový modul

V této části byla popsána realizace databázové části programu, která byla složena z továrny vytvářející spojení pro databázové služby. Na služby jsou napojeni manažeři, na které bylo napojeno uživatelské rozhraní. Každá část dostala svůj specifický úkol.

Továrna

Balíček pojmenovaný *Factory* má za úkol inicializovat připojení k databázi. Obsahuje abstraktní třídu pro databázi, od které musí každá třída s databází dědit. To umožňuje snadnou rozšiřitelnost o podporu dalších typů databází. V této abstraktní třídě byly definovány některé očekávané společné metody, z nichž nejdůležitější je *createDatabaseSchema*. Tato metoda načte soubor pomocí proměnné *SchemaResourceName* (u Firebird databáze je jejím obsahem text *DatabaseSchema.sql*), u které se předpokládá, že bude součástí balíčku implementace konkrétní databáze. Pomocí něho se vytvoří tabulky a vztahy uvnitř databáze.

Dále abstraktní třída obsahuje nedefinované proměnné a metody:

ConnectionNameDriver	jméno ovladače pro databázi
ConnectionPrefixURL	prefix z cesty použité z připojení k databázi
connectionURL	cesta použitá k připojení k databázi
DatabaseFileSuffix	přípona u souboru vytvořeného databází
databaseName	jméno databáze
databaseURL	absolutní cesta, kde je databáze uložena
password	heslo k databázi
SchemaResourceName	jméno nebo cesta k souboru obsahující schéma databáze
userName	uživatelské jméno k databázi
createDatabase(...)	vytvoří a inicializuje novou databázi
openDatabase(...)	připraví objekt s databází k vytvoření spojení
getNewConnection()	vytvoří nové spojení s databází.

Třída *FirebirdDatabase* dědí od výše zmíněné abstraktní třídy a definuje abstraktní proměnné a metody. Při vytváření nové Firebird Embedded databáze nelze nastavit její kódování, tudíž je kódování nastaveno dodatečně při vytváření struktury databáze v souboru *DatabaseSchema.sql*. Uživatelské jméno je nastaveno '*SYSDBA*' a heslo '*masterkey*'. Pozn.: Tyto přihlašovací údaje nechrání u embedded databází před vnějším vniknutím.

Poslední třídou v této části je *DatabaseFactory*, která zastřešila návrh připojení k databázi. Tato třída si udržuje referenci na vybranou databázi a otevřené spojení. Také obsahuje logiku k vytvoření nové databáze a zkontrolování jejího jména, u kterého byly nastaveny podmínky: délka jména je delší než tři znaky, možnými znaky jsou písmena z anglické abecedy, pomlčka a podtržítko. Náplní této třídy je poskytnout spojení třídám ze služební vrstvy. Také umožňuje mazat soubory ze souborového systému databáze a vytváří nové databáze.

V této části mohou nastat výjimky:

CannotCreateDatabaseException	nelze vytvořit databázi
CannotCreateDatabaseFileException	nelze vytvořit soubor s databází
CannotCreateDatabaseFolderException	nelze vytvořit složka, do které bude databáze uložena
CannotCreateDatabaseSchemaException	nelze inicializovat databázi
CannotFindDatabaseFileException	nelze najít soubor s databází
NoActiveDatabaseConnectionException	není spojení k databázi.

Objekty

Ke každé tabulce v databázi byla vytvořena třída, jejíž instance představuje jeden řádek databázové tabulky a jednotlivé sloupce jsou v objektu uloženy jako proměnné.

Dále byl vytvořen výčtový typ *PropertyType*, který rozlišuje mezi vlastností typu text a celým číslem. Pak je tu také objekt *ProductTreeNode*, jenž mimo jiné obsahuje SQL příkazy na výběr různých druhů stromů z databáze a pak tyto stromy reprezentuje.

Služby

Třídy v balíčku *Services* umožňují komunikaci s jednotlivými tabulkami, s výjimkou *statsService*, která slouží pouze ke čtení dat napříč více tabulkami. Každá služba se připojuje k továrně a žádá ji o spojení s databází. Pomocí tohoto spojení, knihovny *DButils* a předem definovaných SQL příkazů, je schopna vybírat, vkládat a upravovat data v tabulkách. K tomu je potřeba vytvořit instanci třídy *QueryRunner*. Té potom pomocí metody *query* nebo *update* (dle požadované funkčnosti) předat SQL příkaz, spojení s databází a v prvním případě i *ResultSetHandler* (v němž se také nachází tzv. *RowProcessor*, který zajišťuje převedení dat z řádků do objektů a je definován programátorem pomocí stejnojmenného interface). V případě jednoduchých výstupů (např. zjištění počtu řádků v tabulce) byl použit *IntScalarHandler*, protože *ScalarHandler* (součást DButils sloužící pro převedení datových typů z řádku tabulky do objektu programu) vrací v různých situacích odlišné datové typy.

Transakce byly vyřešeny na straně programu tak, že se vypne automatické ukládání pomocí metody u spojení s databází (*conn.setAutoCommit(false)*). To umožňuje použít sled SQL příkazů, které se buď v databázi projeví všechny (při potvrzení) a nebo se neprojeví žádný. Druhou a nepoužitou možností bylo vytvořit proceduru v databázi.

Ke správě výrobků byla vytvořena ***ProductService.*** Ta umožňuje vybrat výrobky několika způsoby: jeden výrobek, seznam všech výrobků, seznam potomků výrobku a seznam výrobků dle příkazu (pro využití *ProductTreeNode*). Také poskytuje metody pro přidání výrobku (viz Příloha A), což je jedna z nejsložitějších a nejnáročnějších funkcí programu, neboť ke správnému hierarchickému zařazení výrobku je potřeba přepočítat výrobky ostatní. Tato metoda byla řešena jako transakce a vytvořena tak, že se výrobek přidá vždy na konec vybrané skupiny (i kořenové skupiny). Při mazání a upravování výrobku je logika jednodušší, neboť stačí smazat výrobek a relační vztahy mezi tabulkami se přizpůsobí těmto změnám (např. smažou vlastnosti již neexistujícího výrobku). Po smazání výrobku je opět nutné přepočítat hierarchii výrobků, a tudíž je tato metoda také řešena pomocí transakce.

Ke správě ostatních tabulek slouží třídy *PropertyService, ProductServiceService a FileService,* které poskytují obdobné přístupové metody ke svým tabulkám jako *ProductService* služba.

Myšlenkou služby *StatsService* je využít možnosti SQL jazyka k vytvoření statistických dat, jako například průměrná délka opravy výrobku. Tato služba neumožňuje vytváření či upravování tabulek.

Manažeři

Tato vrstva navazuje na vrstvu služeb a obstarává funkce zahrnující: souborový systém, dědění vlastností, převod seznamu do objektu *ComboBoxModel a* dialogová okna. Také zpracovává neodchycené výjimky. Tato vrstva je již určena ke komunikaci s ostatními částmi programu.

ProductManager navíc obstarává: přidání vlastností od předka (při vytvoření výrobku), přejmenování hlavního obrázku (v případě změny jména výrobku) a smazání všech souborů od právě vymazaného výrobku i jeho potomků.

PropertyManager navíc obstarává přidávání a mazání vlastností u všech potomkům vybraného výrobku. Upravování vlastností je řešeno jednotlivě.

Pro potřeby *StatsManager* byl vytvořen *PropertyIntegerComparator* umožňující seřadit vlastnosti jejichž hodnoty jsou celočíselného typu.

V programu se dále používají třídy *FileManager, ProductServiceManager.*

4.1.2 Práce se soubory

Dalším samostatným funkčním celkem jsou třídy určené pro práci se soubory.

Filtry

V programu je použit *JFileChooser*, který umožňuje výběr souborů, a protože orientace mezi všemi soubory by mohla být nepřehledná, byly vytvořeny třídy (některé již v bakalářském projektu), jejichž instance umožňují filtrovat:

databázové soubory– přípona FDB
soubory obrázků – přípony jpg, jpeg, png, gif
komprimované soubory zip – přípona zip
html soubory – přípona html, htm, xhtml.

Konvertování obrázků

Při exportování výrobků do html bylo zjištěno, že by byl export jednodušší, pokud by všechny obrázky byly v jednom formátu. Z tohoto důvodu byla vytvořena třída *ConvertyAnyToJPG*, která umožňuje zkonvertovat následující formáty: PNG,BMP,WBMP,GIF do JPG. Konvertování obstarává třída *ImageIO*.

Souborové nástroje

Se soubory se pracuje napříč celým programem, a proto byla v bakalářském projektu vytvořena třída *FileUtils*, která umožňuje kopírování souborů. Do funkcí této třídy bylo přidáno zjišťování existence, přípony a typu souboru a vytváření složek.

Zabalení složky

Třída *ZipFolder* umožňuje zkomprimovat metodou zip vybranou složku a všechny soubory i složky v ní obsažené do jednoho souboru.

4.1.3 Export

Dalším samostatným modulem je Export dat o výrobku nebo výrobcích. Jako výstupní formát bylo v návrhu zvoleno html. V tomto formátu bylo vytvořeno sedm šablon, které byly uloženy do složky *Sablony* relativně k umístění programu. Šablona byla strukturována do dvou části. V hlavičce jsou informace o výrobku, která je ve všech šablonách stejná. V druhé části jsou informace o vlastnostech, servisech a souborech dle vybrané šablony. Jméno šablony je identifikátorem obsahu. Případné hlavní obrázky výrobku jsou zkopírovány k exportovanému souboru.

Vytvoření exportovaného souboru probíhá: čtením šablony po řádcích, doplněním dat do přečteného řetězce a následným zapsáním do výstupního souboru. Šablony lze měnit, ale musí se zachovat pozice prvků ve špičatých závorkách, které určují jaká data se mají doplnit (například * {jméno výrobku}).*

Šablony

vyrobek-Servis.html
vyrobek-ServisSoubory.html
vyrobek-Soubory.html
vyrobek-Vlastnosti.html
vyrobek-VlastnostiServis.html
vyrobek-VlastnostiServisSoubory.html
vyrobek-VlastnostiSoubory.html

4.1.4 Hlavní třída (Main)

Hlavní program je zde pouze v roli spouštěče. Inicializuje a spustí hlavní okno programu, kterému nastaví vzhled (*LookAndFeel*) podle operačního systému.

Do stejného balíčku byla vložena ikona a třída *Settings* (viz Příloha B), ve které se nachází statické proměnné specifikující nastavení programu a statické metody pro kontrolu dostupnosti souborů či pro odstranění nepoužitelných znaků v souborovém systému Windows.

4.1.5 Hlavní okno

Uživatelské rozhraní bylo rozděleno na část s hlavním oknem a na část s moduly. Hlavní okno bylo vytvořeno pomoci Swing GUI designer a obsahuje pouze horní menu. Uvnitř třídy byly definovány metody pro načtení a obnovení modulů a předávání jména vybraného výrobku. Také byla vytvořena chyba *ProductNotChoosedException*, která nastane když není vybraný žádný výrobek.

V práci bylo vytvořeno šest *Dockable* objektů: seznam výrobků,seznam vlastností, hlavní obrázek, seznam servisních zásahů,seznam souborů a statistiky. Tyto objekty jsou spojeny v jeden celek v hlavním okně pomocí *DockFrontend* a *SplitDockStation*.

Ve stejném balíčku se nachází třída l*ayout* se statickými metodami k ukládání a načtení souboru rozložení jednotlivých *Dockable* objektů. Pro tento úkon existují dva soubory, jež jsou uloženy relativně k umístění programu. *UserLayout.save* se vytvoří až po prvním spuštění programu a obsahuje takové rozvržení, které si nastavil uživatel. Do *defaultLayout.save* bylo uloženo základní rozložení, které je načteno v případě, kdy uživatelské chybí, anebo v případě uvedení rozvržení do základního stavu. Pokud uživatel omylem smazal i tento soubor, tak je v hlavním okně definována nouzová *SplitDockStation,* která umožňuje nouzové načtení všech modulů.

Další třídy v balíčku obstarávají uložení a zpracování informací o naposledy otevřených databázích, to umožní uživateli rychleji otevírat databáze. Pro tento účel je vytvořen objekt typu *Stack,* ve kterém je uloženo pět absolutních cest k naposledy otevřeným databázím. Tento *Stack* se po ukončení programu ukládá binárně do souboru *lastOpened.save* a při běhu programu je aktualizován.

4.1.6 Moduly UI

Druhou částí uživatelského rozhraní jsou moduly, které představují funkční a obsluhující celky k manažerům.

Balíček *Abstrakt* není modulem a nachází se v něm abstraktní třída rozšiřující *PopUpMenu* s metodou pro zablokování všech komponent, které jsou v této třídě uloženy. Tato třída slouží k vytvoření vyskakovacích menu, které lze jednoduše de/aktivovat.

Celkem bylo vytvořeno šest modulů. Z nichž každý obsahuje panel (zobrazený v hlavní okně), vyskakovací menu a několik obsluhujících dialogových oken.

Modul výrobek

Modul *Product* obsahuje dva panely: první *ProductTree* umožňuje vybírat mezi výrobky, druhý *ImagePanel* obstarává načtení hlavního obrázku k výrobku.

ProductTree je panel a obsahuje pouze prvek *jTree* pro zobrazení hierarchického stromu výrobků. Mezi jeho funkce patři různé možnosti řazení (abecedně bez skupin, podle posledního servisu, podle vlastnosti). A také umožňuje vyhledat výrobek dle části jména a zvolit ho v zobrazeném stromu.

Panel obrázek obstarává zobrazení a nahrávání obrázků. Pokud k výrobku neexistuje obrázek, tak se zobrazí výchozí obrázek uložený ve složce s programem pod názvem *unknown.jpg.* Tento obrázek může uživatel změnit. Před výběrem nového obrázku je vložen filtr typu *OrbFilter*

do objektu typu *JFileChooser* (zajišťuje výběr souborů). Tímto je dosaženo zobrazení souborů jen s příponou .jpg .jpeg .gif .png. Po potvrzení výběru se obrázek přeformátuje do jpg, aby se vyhnulo případným problémům při exportování výrobku. Následně se přeformátovaný obrázek zkopíruje do adresářové struktury programu pojmenovaný stejným jménem výrobku s příponou *HlavniObrazek.jpg*. Pokud jméno výrobku nesplňuje kritéria pro vytvoření souboru ve Windows nebo obsahuje háčky a čárky, tak se tyto znaky převedou na slovní ekvivalent nebo na znak bez diakritiky. Filtrované znaky: *mezera +&@#/%?=~|!:,.;ěščřžýáíéóúůťďĚŠČŘŽÝÁÍÉÓÚŮŤĎ.* Další možnosti výskytu znaků nebyly opomenuty, neboť jména výrobků a další vstupy do databáze jsou též filtrovány. Tudíž se předpokládá, že databáze nebude upravována jiným programem. V některých případech může dojít ke konfliktu. Změna formátování a kopírování probíhá pomocí metody *write* třídy *javax.imageio.ImageIO*. Podporované typy obrázků jsou png, jpeg a gif.

Dále modul výrobek obsahuje dialogová okna pro přidání výrobku a výběr vlastnosti, podle které bude seřazen strom výrobků. Dále tento balíček obsahuje ovladač pro vyvolání okna při přidání a vymazání výrobku. Tím se předchází duplicitnímu kódu.

Modul vlastnosti

V modulu vlastnosti se nachází jeden panel vlastností, ve kterém dochází k dynamickému generování objektů typu *propertyComponent*, které umožňují zobrazení jména a hodnot vlastností a také umožňují jejich změnu. V záhlaví panelu je uvedeno jméno výrobku. Výběr výrobku se provádí v panelu *Seznam výrobků*.

Grafická část *propertyComponent* je vytvořena v Swing GUI Builder. Objekt této třídy rozlišuje dva stavy. Stav pro čtení a stav, kdy uživatel chce změnit hodnotu. V prvním případě jsou zobrazeny dva objekty: *Jlabel s* jménem vlastnosti a *EditText* s hodnotou. V druhém případě se pod dříve zmíněnými objekty zobrazí dvě tlačítka: uložit a zrušit.

Dále obsahuje dialogová okna na přidáni, upravování a mazání vlastností. Při přidání a upravování vlastností rozlišujeme dva typy hodnot, proto je nutné ošetřit vstupy regulárními výrazy. Pro číselný formát je použit regulární výraz *[0-9]**. Pro textový formát je použit regulární výraz *[-a-zA-Z0-9+&@#/%?=~_|!:,.;ěščřžýáíéóúůťďĚŠČŘŽÝÁÍÉÓÚŮŤĎ\\s]**. Toto omezení neplyne z možností databáze a uživatelského rozhraní, ale z toho, aby byl obsah databáze jednotný, resp. aby nebylo nutné používat různé regulární výrazy pro různé tabulky a aby se předešlo nečekaným scénářům (např. čínské znaky).

Tento modul také obsahuje ovladač pro vyskakovací menu a spouštění dialogových oken.

Servisní zásahy výrobku

Panel servisních zásahů tvoří jeden *combobox* umožňující výběr data obdržení servisního zásahu. Tři *jLabel* s popisy dat a datem vrácení. A dva *jTextPane* s popisem problému a řešení. Tyto komponenty jsou v panelu rozvrženy pomocí *AbsoluteLayout.*

Dále se v modulu vyskytují dialogy pro přidávání, upravování a odebírání servisních zásahů. Úplné přidání servisního zásahu probíhá ve dvou krocích. Při obdržení výrobku k opravě se přidá servisních zásah a při odevzdání funkčního výrobku se upraví, již dříve vytvořený, servisní zásah.

Soubory výrobku

Panel modulu tvoří objekt typu *jList* se seznamem souborů, dva *jLabel* určující typ a jméno souboru, tlačítko k otevření souboru a *jText* s popisem souboru.

Dále se v modulu nachází: vyskakovací menu pro přidání a smazání souboru, dvě dialogová okna ke správě souborů a ovladač vyvolávající dialogová oknům.

Statistika

Toto je modul o dvou třídách. Jedna třída je panelem v hlavním okně a obsahuje tři *jLabel* informující o nejporuchovějším výrobku ve skupině, počtu dnů v opravě a průměrné délce jedné opravy.

Druhá třída představuje dialogové okno zobrazující výstupní sestavy:

- deset naposledy opravených výrobků
- deset výrobků s nejdelší dobou strávenou v servise
- porovnání výrobků dle vlastnosti.

4.2 Shrnutí funkcí programu

Výsledný program umožňuje vytvářet a spravovat databáze výrobků. Ke každému výrobku lze uchovávat vlastnosti a soubory, údaje o servisních zásazích. Pokud je výrobek ve skupině, tak přebírá za své veškeré vlastnosti skupiny. Pokud se smaže vlastnost skupiny, tak se tím smaže i vlastnost u každého výrobku obsaženého ve skupině. Kvůli této dědičnosti není implementováno přesouvání výrobku mezi skupinami, protože není zřejmé, jak by se metoda, provádějící tento úkon, měla zachovat. Ze stejného důvodu se provádí upravení vlastnosti jednotlivě, tudíž změna vlastnosti skupiny neovlivní výrobky v ní obsažené. Další funkce programu se také provádí jednotlivě.

Data uložená v databázi je možné zálohovat. Výstupem bude zazipovaný soubor obsahující všechna data, soubory i obrázky. Dále je možné data exportovat do HTML, a to buď jednotlivě anebo hromadně výběrem přes CTRL.

4.2.1 Přehled souborů

Konečná verze programu používá kromě databázových souborů, knihoven a šablon tyto soubory:

defaultLayout.save - pro uložení základního rozestavení prvků uživatelského rozhraní
lastOpened.save - pro uložení absolutních cest k naposledy otevřeným databázím
unknown.png - výchozí obrázek pro nový výrobek
userLayout.save - pro uložení uživatelského rozestavení prvků uživatelského rozhraní
log.log - pro uložení chybových stavů programu.

4.3 Tvorba dokumentace

Během tvorby zdrojového kódu probíhalo vytváření dokumentace dle pravidel vytváření komentářů pro Javadoc viz [9]. Z těchto komentářů se pomocí integrovaného nástroje Javadoc ve vývojovém prostředí NetBeans IDE vygenerovala dokumentace ve formátu HTML (viz příloha na CD - dokumentace kódu).

Fungování a chování výsledného programu bylo zdokumentováno v uživatelském manuálu (viz Příloha E). V tomto dokumentu jsou popsány kroky od instalace programu přes vytvoření první databáze až po zálohování naplněné databáze. Text v dokumentu je provázen obrázky z programu znázorňující popisované dění.

5 Vyhodnocení řešení

Funkčnost programu byla otestována na Windows XP 32bit a Windows 7 32bit a 64bit s nainstalovaným JRE 32. Program nefunguje pod JRE 64 bit kvůli nahrané 32 bitové externí knihovně obsluhující Firebird Embedded. Na základě používání programu ve Windows 7 64 bit bylo pomocí programu správce úloh zjištěno, že se využití paměti pohybuje mezi 60 MB až 150 MB v závislosti na velikosti databáze a uvolnění prostředků.

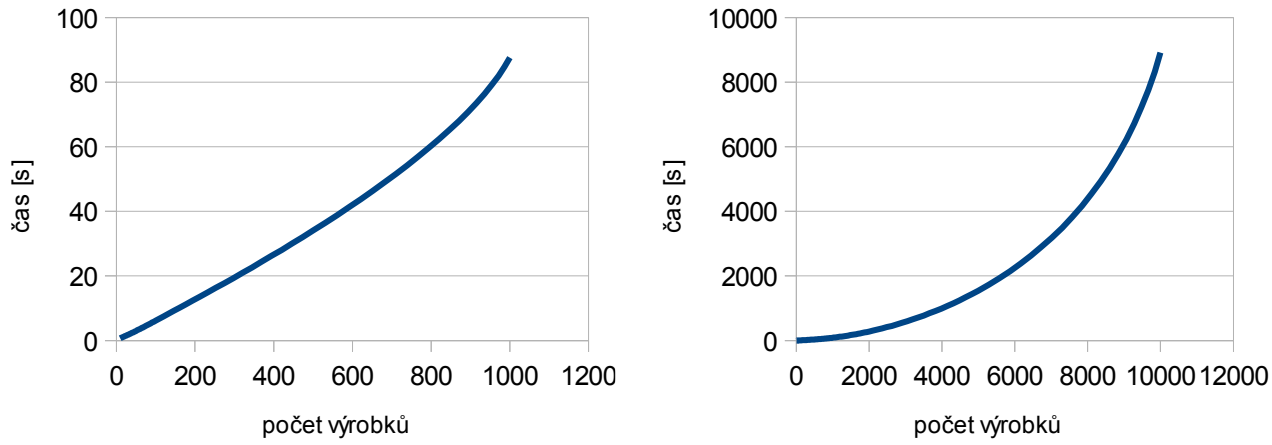
Složité výpočetní úlohy (např. seřazení dle posledního servisu) u příliš velké databáze se zdály být problematické, protože byly schopny spotřebovat většinu paměti v počítači, aniž by tato paměť byla viditelně (správce úloh) přiřazena některému z procesů. Toto chování nakonec bylo odstraněno opravením SQL příkazu (viz Příloha C).

5.1 Zátěžové testy

Po dokončení programu byla v testovacím balíčku vytvořena třída *Generator* pro automatické generování výrobků, jejich vlastností, servisních zásahů a souborů.

Pro generování musí být vždy použita nová databáze, aby nedošlo ke konfliktu. Data jsou generována pomocí služeb z hlavního programu. Jména a jiné názvy jsou generována dynamicky rovnající se číslu unikátní operace (pokud může dojít ke konfliktu, číslo unikátní operace se zvýší o 1). Každé použité datum je generováno náhodně a může u něj dojít ke konfliktu. V generátoru lze nastavit: počet výrobků ve vrstvě, hloubku hierarchického stromu, počet generovaných vlastností pro výrobek, počet servisních zásahů a souborů (v jedné proměnné).

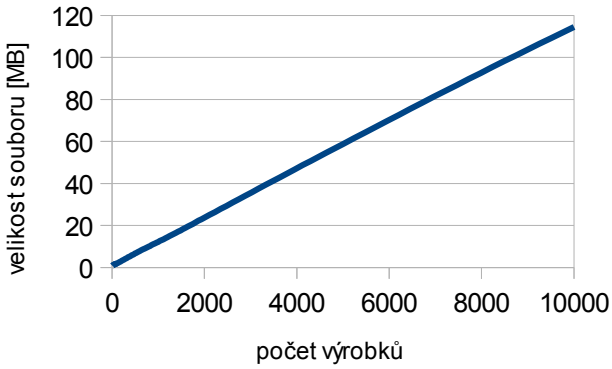
Nejdříve byly vytvořeny databáze o 10, 50, 100, 500, 1000, 2000, 5000 a 10 000 výrobcích a při vytváření byl měřen čas. Ke každému takto vytvořenému výrobku bylo přiřazeno sedm vlastností,



Obrázek 5.1: Grafy v různém rozlišení vyjadřující čas potřebný k vytvoření databáze s x výrobky

dvacet servisních zásahů a dvacet souborů. U 10 000 výrobků se v databázi nacházelo půl milionu záznamů.

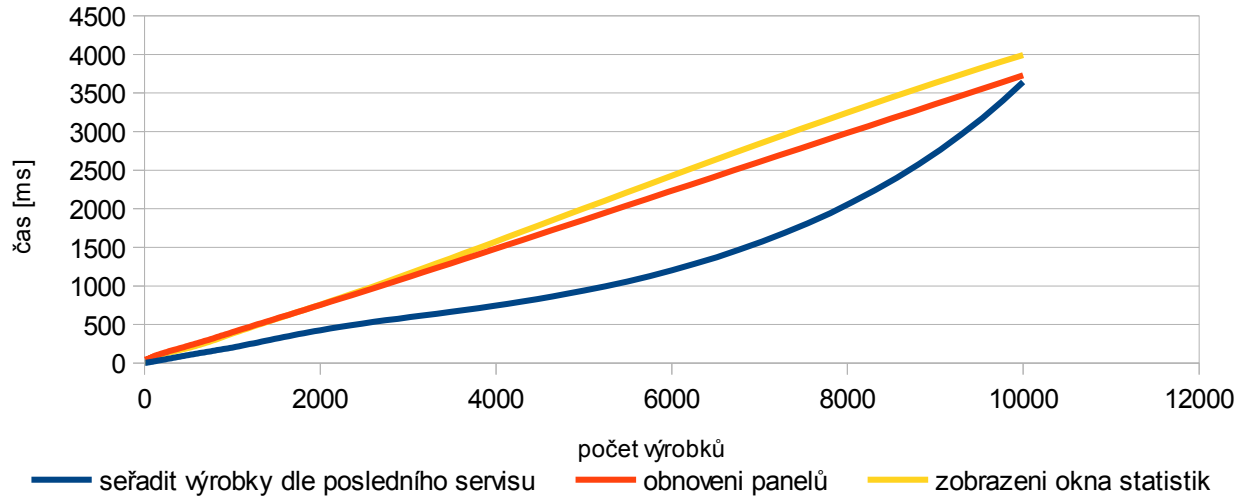
Výsledky první části testu zjistili, že náročnost vytvoření nového výrobku roste s počtem výrobku do 1000 zhruba lineárně (viz Obrázek 5.1 první graf) a přes 1000 výrobků se blíží k mocninnému růstu (viz Obrázek 5.1 druhý graf). Toto zpomalení je způsobeno zejména databází. v testu nebyla odzkoušena hierarchie výrobků, která by vytvoření nového výrobku zpomalila ještě o trochu více vzhledem k tomu, že by se musela část tabulky výrobků přepočítávat. Velikost takto vytvořené databáze se zvyšuje lineárně (viz Obrázek 5.2).



Obrázek 5.2: Graf velikosti souboru s databází v závislosti na počtu výrobků

V dalším kroku byl přidán kód k měření času výkonu sekvence příkazů: obnovení stromu výrobků (seřazených dle posledního servisu), výběr výrobku a inicializace okna statistik. Tato místa jsou považována za nejrizikovější, neboť pracují s nejvíce daty. Z testu vyplynulo, že nároky programu na výpočetní výkon rostou s počtem dat v programu (viz Obrázek 5.3) a tento růst nemusí být pouze lineární. Optimální maximální počet výrobků při rozumném čekání se pohybuje mezi 1000 až 2000 výrobky v závislosti na výkonnosti počítačové sestavy. V případě překonání této hranice je možné databázi rozdělit na dvě a tím program zrychlit.

Na základě výsledků testů bylo do rizikových míst programu přidáno rozlišení zaneprázdněného stavu programu pomocí kurzoru myši, což obstarává dodatečně vytvořená třída *CursorController*.



Obrázek 5.3: Graf růstu času vypočtu v rizikových místech v závislosti na počtu výrobků

Závěr

V této práci byla představena komplexní problematika PLM a MRO softwaru. Cílem bylo vytvořit jednoduchou náhražku MRO software, umožňující uchovávat na jednom místě údaje o výrobcích a servisních zásazích. Vytvořený prototyp z bakalářského projektu byl kompletně přestrukturován: pozměnil se návrh databáze, funkční části programu byly vloženy mimo třídy s uživatelským rozhraním a uživatelské rozhraní bylo navrženo pomocí frameworku DockingFrames. Dále se přidaly nové možnosti ve formě přívětivějšího uživatelského rozhraní, možnosti exportu dat a další.

Nakonec byl výsledný program odzkoušen na operačním systému Windows 32bit i 64bit s předinstalovaným JRE 32bit (kvůli externí knihovně) a byl podroben zátěžovým testům. Z těch vyplynulo, že program je použitelný pro jednotky tisíc výrobků, což odpovídá zhruba 70 tisícům řádků v databázi (ke každému výrobku jsou definovány servisní zásahy, vlastnosti a soubory). Zdrojové kódy a funkce programu byly zdokumentovány.

Z funkcionality programu a zátěžových testů bylo usouzeno, že cíle práce byly naplněny, a tudíž je výsledný program připraven k použití. Práce by mohla být rozšířena o modulární pojetí výstupních sestav a o podporu vícero databází.

Seznam použité literatury

[1] About Firebird. *Firebird* [online]. 2000 [cit. 2012-05-05]. Dostupné z: http://www.firebirdsql.org/en/about-firebird/

[2] ANDORKA, Frank. New Study Highlights Best PLM Vendors. *Industry week* [online]. [cit. 2012-04-07]. ISSN 0039-0895. Dostupné z: http://www.industryweek.com/articles/new_study_highlights_best_plm_vendors_24131.aspx

[3] Architektura Systému. *Maximo* [online]. 2008 [cit. 2012-05-05]. Dostupné z: http://www.maximo.cz/maximo/index.php?index=submenu/maximo5.php

[4] Bankrot Cimber Sterling. *Amicos* [online]. 2012 [cit. 2012-05-05]. Dostupné z: http://www.amicos.com/

[5] Docking Frames. *Docking Frames* [online]. 2001 [cit. 2012-04-24]. Dostupné z: http://dock.javaforge.com/

[6] Firebird: Initial Developer's Public Licence. [online]. [cit. 2012-03-26]. Dostupné z: http://www.firebirdsql.org/en/initial-developer-s-public-license-version-1-0/

[7] Home. *Microba Controls* [online]. 2005 [cit. 2012-04-24]. Dostupné z: http://microba.sourceforge.net/

[8] HOMOLA, Jan. PLM - Product Lifecycle Management. *PLM.cz* [online]. [cit. 2012-04-06]. Dostupné z: http://plm.caxmix.cz/definice-plm/

[9] How to Write Doc Comments for the Javadoc Tool. *Oracle Technology Network* [online]. 2004 [cit. 2012-05-11]. Dostupné z: http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html

[10] Info:licensing. *Jaybird wiki* [online]. 2005 [cit. 2012-04-24]. Dostupné z: http://jaybirdwiki.firebirdsql.org/jaybird/doku.php?id=info:licensing

[11] KOLÁŘ, Radim. Root.cz: Seriál Embedded databáze. [online]. [cit. 2012-03-26]. Dostupné z: http://www.root.cz/serialy/embedded-databaze/

[12] Maintenance, repair, and operations. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2012-04-08]. Dostupné z: http://en.wikipedia.org/wiki/Repair_and_maintenance

[13] Modules. *Swiss AviationSoftware* [online]. 2012 [cit. 2012-05-05]. Dostupné z: http://www.swiss-as.com/modules.do

[14] MRO IT market suppliers survey. *Aircraft Commerce* [online]. Issue No. 56 [cit. 2012-04-08]. Dostupné z: http://www.o-sys.com/pdf/aircraft-commerce_mrosurvey_march08.pdf

[15] PAROUBEK, Martin. *Servisní databáze výrobků*. Liberec, 2011. 25 s. Bakalářský projekt. Fakulta mechatroniky, informatiky a mezioborových studií. Technická univerzita v Liberci. Vedoucí bakalářského projektu Ing. Tomáš Martinec, Ph.D.

[16] PLM Vendors. *PLM Technology Guide* [online]. c2008 [cit. 2012-04-07]. Dostupné z: http://plmtechnologyguide.com/site/?page_id=12

[17] Project License. *Apache Commons* [online]. 2002 [cit. 2012-04-24]. Dostupné z: http://commons.apache.org/dbutils/license.html

[18] Project License. *Logging services* [online]. 2002 [cit. 2012-05-13]. Dostupné z: http://logging.apache.org/log4j/1.2/license.html

[19] SÄÄKSVUORI, Antti a Anselmi IMMONEN. *Product lifecycle management*. 3rd ed. Berlin: Springer, 2008, s. 1-6. ISBN 9783540781738.

[20] SIGG, Benjamin. *DockingFrames 1.1.0 - Core*. 2011. Dostupné z: http://dock.javaforge.com/dockingFrames_v1.1.0/core.pdf, s. 9-28.

[21] STARK, John. *Product lifecycle management*: *21st century paradigm for product realisation*. 2nd ed. New York: Springer, c2011, s. 185. ISBN 0857295454.

[22] SVOBODA, Tomáš. Informace o výrobcích jako na dlani. Eva Vaculíková. *Automa*: *časopis pro automatizační techniku*. Praha: FCC Public, 2009, roč. 2009, č. 1, s. 8-9. ISSN 1210-9592.

[23] Úvod. *TD-IS, s.r.o.* [online]. 2006 [cit. 2012-05-06]. Dostupné z: http://www.td-is.cz/cs/uvod.htm

Příloha A: Metoda pro přidání výrobku

```
public boolean addProduct(DatabaseProduct dp) {
  boolean ouputStatus = true;
  DatabaseFactory fd = DatabaseFactory.getInstance();
  QueryRunner qr = new QueryRunner();
  try {
   fd.getConnection().setAutoCommit(false);
   if (!dp.getParentProductName().isEmpty()) { // existuje parent
    DatabaseProduct productParent = this.getProduct(dp.getParentProductName());
    int right;
    try {
     right = productParent.getRgt();
    } catch (NullPointerException ex) {
     right = 0;
    }
String sql1="UPDATE \"vyrobky\" SET \"rgt\" = \"rgt\" + 2 WHERE \"rgt\" >= ? ";
String sql2="UPDATE \"vyrobky\" SET \"lft\" = \"lft\" + 2 WHERE \"lft\" >= ? ";
    qr.update(fd.getConnection(), sql1, right);
    qr.update(fd.getConnection(), sql2, right);
    int lft = right;
    int rgt = lft + 1;
    String createSql = "INSERT INTO \"vyrobky\" (\"id_vyrobek\",
                \"id_vyrobek_nad\",\"lft\",\"rgt\") " + "VALUES (?,?,?,?);";
    qr.update(fd.getConnection(), createSql, dp.getProductName(),
    dp.getParentProductName(), lft, rgt);
   } else { // bez rodiče
    String sqlLftMax = "SELECT max(\"vyrobky\".\"rgt\") FROM \"vyrobky\";";
    ResultSetHandler<Object> h = new IntScalarHandler();
    int lft = 1;
    try {
     lft += (Integer) qr.query(fd.getConnection(), sqlLftMax, h);
    } catch (NullPointerException ex) {//OK zadne vyrobky v databazi}
    int rgt = lft + 1;
    String createSql = "INSERT INTO \"vyrobky\"
                    (\"id_vyrobek\", \"id_vyrobek_nad\",\"lft\",\"rgt\") "
      + "VALUES (?,?,?,?);";
    qr.update(fd.getConnection(), createSql, dp.getProductName(),
    dp.getProductName(), lft, rgt);
   }
   fd.getConnection().commit();
  } catch (SQLException ex) {
   log.error("Nepodařilo se přidat výrobek.", ex);
   try {
    fd.getConnection().rollback();
   } catch (SQLException ex1) {
    log.error("Nepodařilo se navrátit z transkace.", ex1);
   }
   ouputStatus = false;
  } finally {
   try {
    fd.getConnection().setAutoCommit(true);
   } catch (SQLException ex) {
    log.error("Nepodařilo se obnovit automatické potvrzování dotazů...", ex);
   }
  }
  return ouputStatus;
}
```

Příloha B: Zjednodušený obsah třídy *Settings*

```
final boolean DEBUG = false;
final String newLine = System.getProperty("line.separator");
final String fileSep = System.getProperty("file.separator");
String DatabaseSaveURL = System.getProperty("user.dir")
                         + fileSep +"Databaze";
String SablonySaveURL = System.getProperty("user.dir") +fileSep +"Sablony";
String DatabaseSaveFilesURL = fileSep +"Soubory";

String imageSuffix = "HlavniObrazek.jpg";
SimpleDateFormat datumUzivatel = new SimpleDateFormat("dd.MM.yyyy");
SimpleDateFormat datumDatabaze = new SimpleDateFormat("yyyy-MM-dd");
String treeRoot = "Seznam výrobků";
String layoutSaveNameDefault="defaultLayout.save";
String layoutSaveNameUser="userLayout.save";
String defaultImageUrl="unknown.png";

String regularLettersNumbers="[-a-zA-Z0-9+&@#/%?=~_|!:,.;
                              ěščřžýáíéóúůťďĚŠČŘŽÝÁÍÉÓÚŮŤĎ\\s]*";
String regularNumbers="[0-9]*";
String regularLetters="[-a-zA-Z]*";
String regularFiles="[-a-zA-Z0-9_]*";

int productNameMaxSize = 50;
int propertyNameMaxSize = 50;
int fileNameMaxSize = 50;
int fileDescriptionMaxSize = 200;
int serviceDescriptionsMaxSize=8000;

int lastOpensSizeStack = 5;

void programInit()
String transferStringToSafe(String retezec)
void checkFileSystem()
```

Příloha C: Příklad vadného a opraveného SQL kódu

```
SELECT * FROM "vyrobky" LEFT JOIN "servisni_zasahy" ON
"servisni_zasahy"."id_vyrobek"="vyrobky"."id_vyrobek" WHERE
"vyrobky"."rgt"="vyrobky"."lft"+1 ORDER BY "servisni_zasahy"."datum_obdrzeni"
DESC
```

doba provádění příkazu se zobrazením do UI 120 sec

```
SELECT "vyrobky".* FROM "vyrobky" LEFT JOIN "servisni_zasahy" ON
"servisni_zasahy"."id_vyrobek"="vyrobky"."id_vyrobek" WHERE
"vyrobky"."rgt"="vyrobky"."lft"+1 ORDER BY "servisni_zasahy"."datum_obdrzeni"
DESC
```

doba provádění příkazu se zobrazením do UI 5 sec

Rozdíl v době provádění je způsobem tím, že jsou zbytečně načítány nepotřebné sloupce, které zabírají hodně paměti.

Příloha D: Obsah CD

dokumentace zdrojového kódu

elektronická verze bakalářské práce

uživatelský manuál

výsledný program

zdrojové kódy

Příloha E: Uživatelský manuál

Uživatelský manuál pro program: Univerzální servisní databáze výrobků

Martin Paroubek

Obsah

1 Úvod

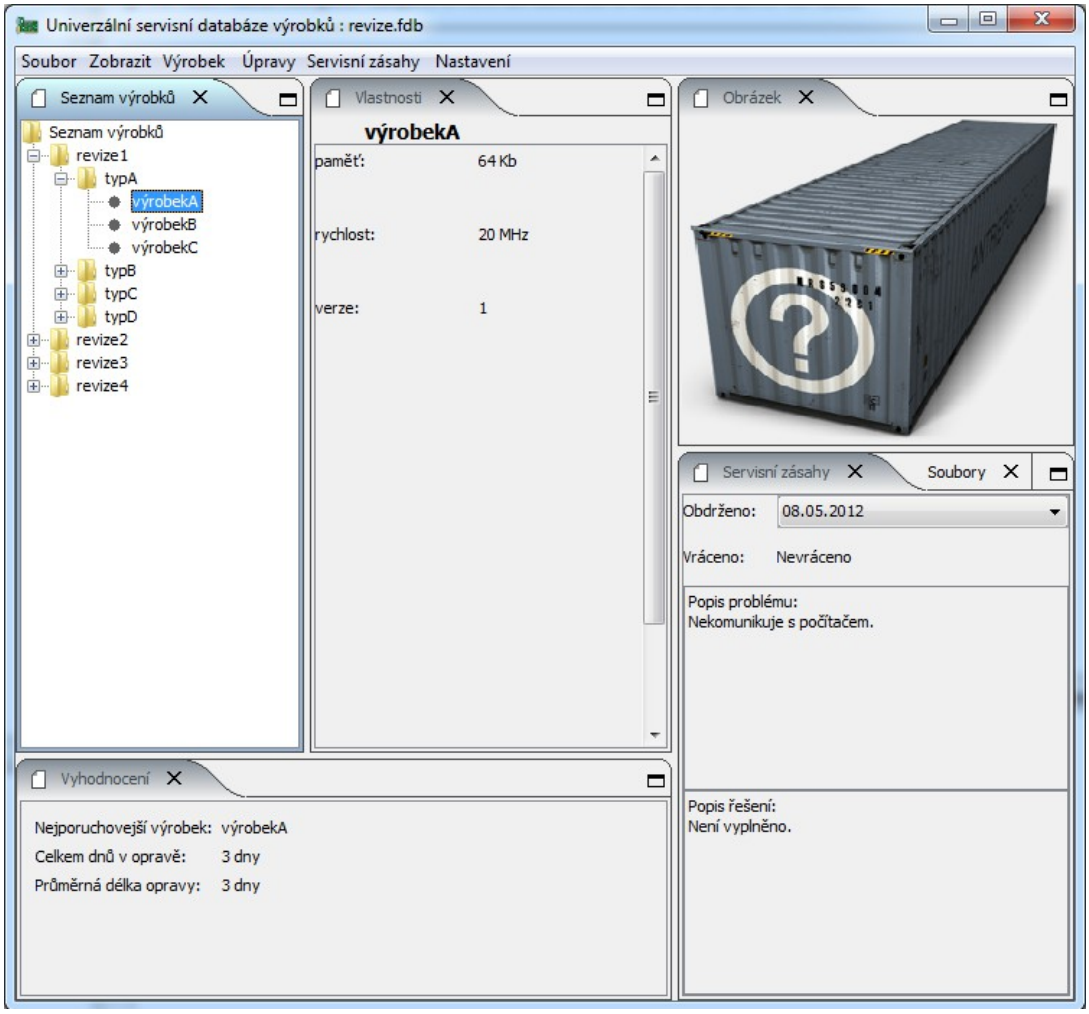
Tento program slouží k uchovávání údajů o výrobcích a o jejich opravách. Jednotlivé výrobky lze řadit do skupin. Skupinu si můžeme představit i jako první verzi výrobku. Ke každému výrobku a skupině definujeme vlastnosti, evidujeme opravy a můžeme k nim přiřazovat soubory (např. Manuál). Veškeré změny v programu se ihned promítají do databáze, proto není třeba databázi ukládat.

1.1 První spuštění

Ke spuštění tohoto programu musíme mít nainstalováno JRE verze 6. Lze stáhnout na této webové stránce http://java.com/en/download/index.jsp. Samotný program poté rozbalíme např. pomocí programu Total Commander a spustíme soubor s názvem *DatabazeVyrobku.jar*.

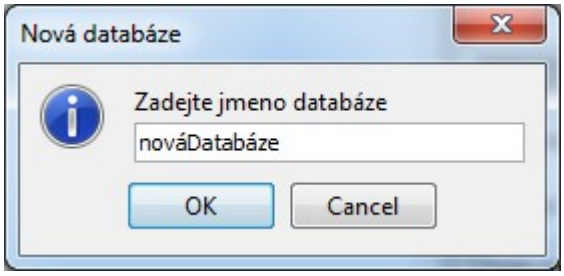
1.2 Základní rozhraní

Základní rozhraní programu tvoří 7 částí. Horní menu, ve kterém provádíme veškeré úkony, dále *seznam výrobků*, *obrázek*, *vlastnosti*, *servisní zásahy*, *soubory*, *vyhodnocení* (viz obrázek č. 1). Většinu částí lze přesouvat nebo skládat na sebe stisknutím levého tlačítka myši na nadpisu podokna a následným přetažením. Modré ohraničení značí, kam se dané podokno přesune po uvolnění tlačítka. Program automaticky ukládá zvolené rozhraní a pro příští spuštění si ho bude pamatovat. Přednastavené rozvržení načteme kliknutím na *nastavení → základní rozvržení.*



1.3 Vytvoření databáze

Novou databázi vytvoříme tak, že klikneme v menu na *Soubor* a z nabídky vybereme *Nová databáze.* Poté se nás program zeptá, jak se má databáze jmenovat. Jakmile potvrdíme jméno, vytvoří se nová složka v umístění hlavního programu ve složce *Databáze*.



1.4 Otevření databáze

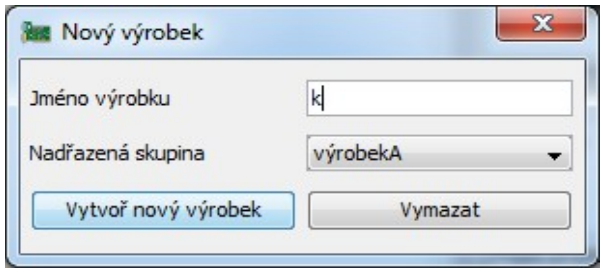
Pokud databázi již máme vytvořenou z dřívějšího spuštění programu, tak ji otevřeme. V horním menu v nabídce *Soubor* vybereme otevřít *databázi* a v novém okně vybereme umístění databáze a zvolíme soubor s příponou *.FDB*.

2 Výrobek

Za výrobek můžeme považovat nějaký produkt jakéhokoliv druhu, o kterém si chceme vést záznamy.

2.1 Vytvoření výrobku

Nový výrobek lze vytvořit dvěma způsoby. Buď si vybereme v horním menu *Výrobek → nový výrobek* a nebo klikneme pravým tlačítkem do oblasti *seznamu výrobku* a opět vybereme *nový výrobek.* V obou případech se objeví nové dialogové okno, do kterého se vyplní jméno výrobku a vybere se do jaké skupiny patří. Po potvrzení se nám nový výrobek objeví ve výběru *Seznam výrobků.*



2.2 Smazání výrobku

Výrobek, který chceme smazat, musíme nejprve označit. To provedeme tak, že na něj klikneme levým tlačítkem v seznamu výrobků. Po označení vybereme z horního menu *Výrobek → smazat výrobek* anebo klikneme pravý tlačítkem do seznamu výrobků a vybereme *smazat výrobek.* Tuto akci je potřeba potvrdit, aby nedošlo k nechtěnému omylu.

2.3 Změna jména výrobku

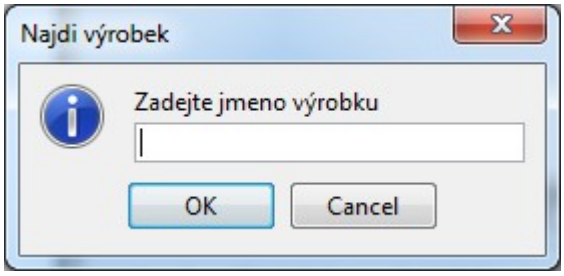
Opět musíme nejdříve označit výrobek, u kterého chceme změnit jméno. Pak klikneme na menu V*ýrobek* a vybereme možnost *změnit jméno výrobku*.

2.4 Obrázek výrobku

Ke každému výrobku můžeme přiřadit obrázek, který se zobrazí v panelu *obrázek*. V horním menu v nabídce *Výrobek* klikneme na *Změnit obrázek*. V novém okně obrázek vybereme. Obdobně to lze udělat přímo kliknutím na obrázek.

2.5 Vyhledávání výrobku

V případě velkého množství výrobků či zapomenutí jeho zařazení do skupin můžeme použít funkci hledání, kterou nalezneme v horním menu *Výrobek → vyhledej*. V novém oknu napíšeme jméno nebo část jména výrobku a potvrdíme. Po nalezení první shody s hledaným jménem se výrobek označí v *Seznamu výrobků*.



2.6 Zobrazení bez skupin

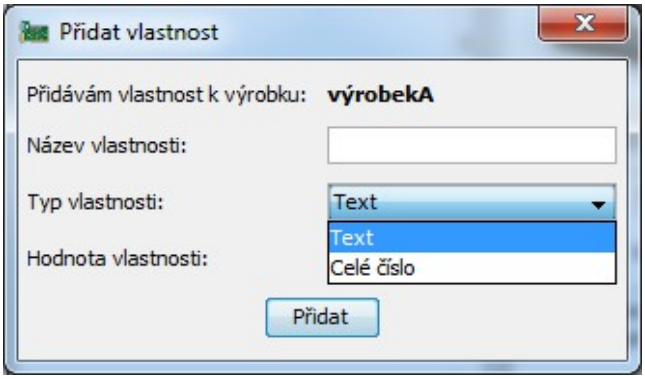
Pokud nás nezajímají skupiny tak v horním menu vybereme *Zobrazit → pouze výrobky*. V tomto formátu lze výrobky řadit opět pomocí horního menu *Zobrazit → Seřadit* a z podnabídky vybereme jednu z možností: abecední řazení, řazení dle poslední opravy, zobrazení pouze výrobku právě v opravě a nebo dle nějaké vlastnosti, kterou sami zvolíme. Jak spravovat vlastnosti výrobku se dozvíme v následující kapitole.

3 Vlastnost

Ke každému výrobku je možné přiřadit vlastnosti vypovídající o jeho složení, struktuře a možnostech.

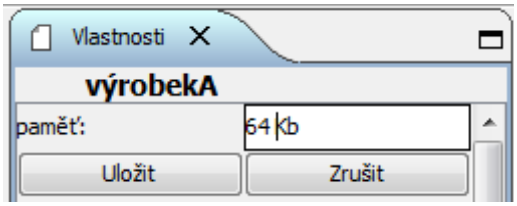
3.1 Vytvoření vlastnosti

Novou vlastnost můžeme přidat obdobně jako v předchozích případech. Nejdříve v seznamu výrobků vybereme výrobek, ke kterému chceme vlastnosti přidat. Potom v horním menu klikneme na *Úpravy → Přidat vlastnost* nebo klikneme pravým tlačítkem na název výrobku v panelu V*lastnosti* a poté na *Přidat vlastnost*. V obou případech se objeví nové dialogové okno, žádající nás o pojmenování, typ a hodnotu vlastnosti.



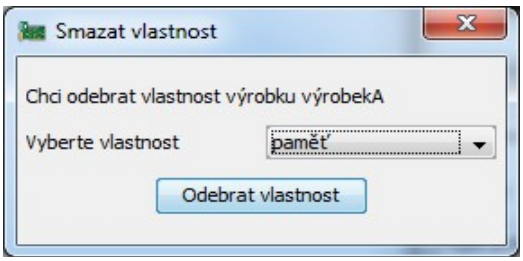
3.2 Upravení vlastnosti

Upravit vlastnost můžeme dvěma způsoby. V obou případech nejdříve v seznamu výrobků vybereme výrobek, ke kterému je úprava vztažena. První možností upravení vlastnosti je přes pomocné okno, které otevřeme z horního menu *úpravy → upravit vlastnost.* Stejný efekt bude mít, když klikneme pravým tlačítkem myši na jméno výrobku a vybereme *upravit vlastnost.* Druhou a rychlejší možností je kliknout na vlastnost v prostředním panelu. Po kliknutí můžeme začít rovnou upravovat hodnotu vlastnosti. Jakmile jsme hotovi, uložíme změny pomocí tlačítka *uložit.*



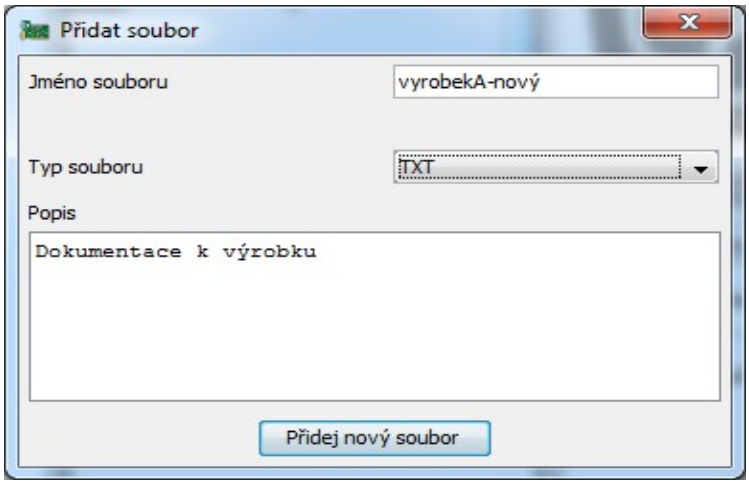
3.3 Smazání vlastnosti

Vlastnost smažeme obdobně jako jsme ji přidali. Vybereme výrobek ze seznamu výrobků a v horním menu, nebo ve vyskakovacím okně při pravém kliknutí v prostředním sloupci na jméně výrobku, vybereme *Smazat vlastnost.* V novém okně vybereme jméno vlastnosti a potvrdíme.



4 Soubory

Jakýkoliv výrobek nebo skupinu je možné spojit se souborem, který se uloží do adresářové struktury databáze. V této verzi program rozlišuje tři typy souborů: textový, obrázek a ostatní.

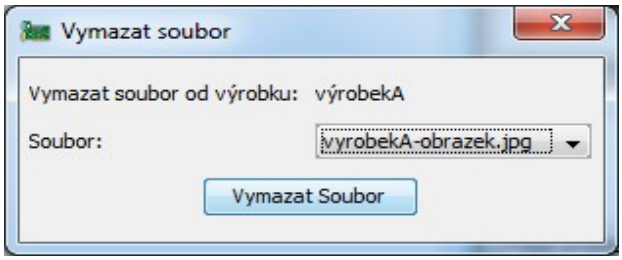


4.1 Přidání souboru

Nový soubor přidáme kliknutím v horním menu na *úpravy → přidat soubory.* V novém okně vybereme soubor, který k výrobku chceme přiřadit. Po potvrzení nás program požádá o typ souboru a jeho popis. Soubor obvykle pojmenováváme *názevVýrobku_jmenoSouboru*.

4.2 Smazání souboru

Soubor smažeme tak, že z horního menu vybereme *úpravy – smazat soubor.* V novém okně vybereme požadovaný soubor a potvrdíme.



4.3 Otevření souboru

Soubor můžeme otevřít v panelu souborů, kde vybereme kliknutím soubor a napravo od výběru se objeví tlačítko otevřít. Soubor lze také otevřít v adresářové struktuře v umístění: *cesta k databázi/Databáze/jmeno databáze/soubory/.*

4.4 Řazení souborů

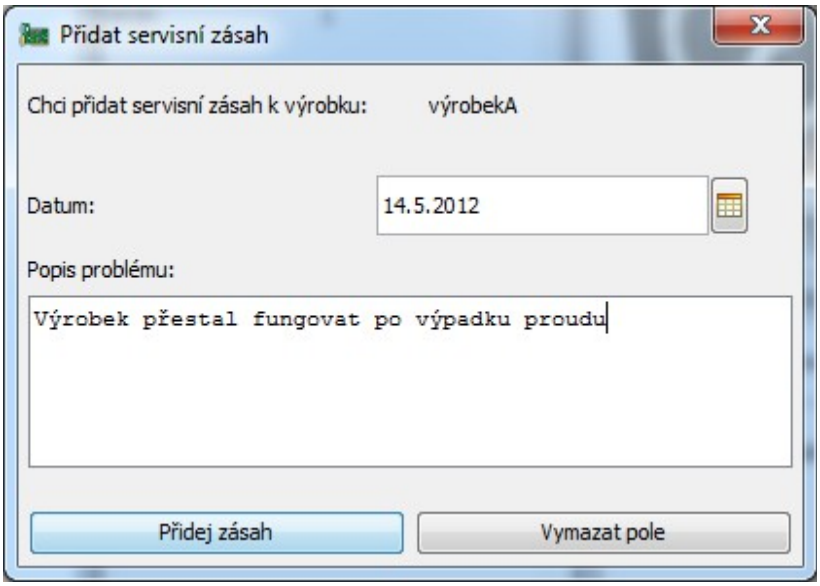
Když klikneme v panelu souborů pravým tlačítkem na výběr souborů, tak se dostaneme do nového menu, pomocí něhož můžeme řadit soubory abecedně nebo dle data. Jako datum se bere v úvahu datum přidání souboru do databáze.

5 Servisní zásahy

Tento program také umožňuje evidovat poruchy a opravy k jednotlivým výrobkům.

5.1 Vytvoření servisního zásahu

Nový zásah vytvoříme v horním menu, když klikneme na *Servisní zásahy → nový servisní zásah*. V novém okně vyplníme, kdy jsme výrobek obdrželi, popíšeme problém a potvrdíme.



5.2 Úprava servisního zásahu

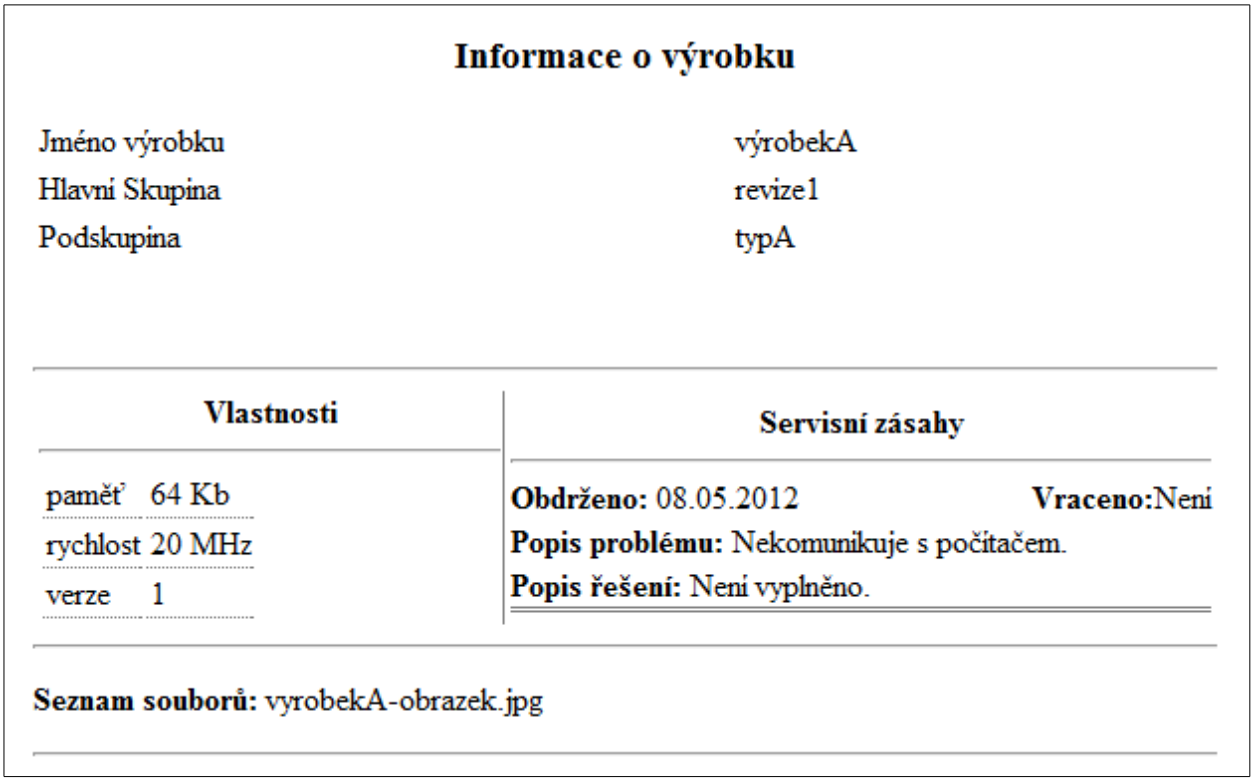
Jakmile se nám podaří problém s výrobkem vyřešit, můžeme upravit servisní zásah. V horním menu klikneme na *Servisní zásahy → upravit servisní zásah.* V novém okně doplníme informace: datum vrácení a popis řešení problému.

5.3 Smazání servisního zásahu

Pokud potřebujeme servisní zásah smazat, klikneme v horním menu na *servisní zásahy → smazat servisní zásah* a v novém okně vybereme podle data, o který zásah se jedná. Potvrdíme a zásah se smaže.

6 Export do HTML

Informace o některém z výrobků anebo i o skupině můžeme prezentovat mimo program a třeba i vytisknout pomocí exportu do HTML. Vybereme výrobky ze seznamu výrobků (pomocí CTRL + kliknutím lze vybrat více výrobků). Jakmile máme výrobky vybrány, tak klikneme v horním menu na výrobek a na export do HTML. V novém okně zvolíme, které informace chceme vytisknout. Po potvrzení musíme vygenerovaný soubor uložit, potom se soubor automaticky otevře.



Informace o výrobku

Jméno výrobku	výrobekA
Hlavní Skupina	revize1
Podskupina	typA

Vlastnosti	
paměť	64 Kb
rychlost	20 MHz
verze	1

Servisní zásahy

Obdrženo: 08.05.2012 **Vraceno:**Není
Popis problému: Nekomunikuje s počítačem.
Popis řešení: Není vyplněno.

Seznam souborů: vyrobekA-obrazek.jpg

Náhled exportovaného výrobku

7 Zálohování

Celou databázi lze zazálohovat. V horním menu v nabídce *soubor* klikneme na *zálohovat*. Poté vybereme umístění zálohy, jméno souboru a dáme uložit. Tímto zálohujeme celou databázi včetně výrobků, souborů i obrázků. Pokud budeme chtít otevřít databázi ze zálohy, stačí tento soubor rozbalit, např. pomocí Total Commander, a otevřít jako jinou dříve uloženou databázi.