UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Vývoj firemních desktopových aplikací s využitím frameworku OpenSwing

Lukáš Voves

Bakalářská práce
2010

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2009/2010**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Lukáš VOVES**
Osobní číslo:	**I07837**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Vývoj firemních desktopových aplikací s využitím frameworku OpenSwing**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem bakalářské práce je zjistit, zda je framework OpenSwing vhodný pro vývoj desktopových aplikací v jazyce Java pro firemní využití, u kterého je kladen důraz na bezchybný, rychlý a kvalitní vývoj aplikací pro koncové uživatele. Důležitým kritériem bude kvalita propojení jednotlivých komponent a jejich spolupráce.
Teoretická část:
V teoretické části bakalářské práce budou popsány technologie pro vývoj firemních aplikací - .Net, Delphi, Java, C++, C atd. Možnosti použití databází - Apache Derby, Informix, Oracle. Výhody a nevýhody jednotlivých řešení.
Implementační část:
Aplikace bude naprogramována v Javě s použitím frameworku OpenSwing. Program s názvem Dealer bude využívat kombinaci databází Oracle a JavaDB Embedded (Apache Derby). V místní databázi bude uchovávat informace o objednávkách a fakturách a databáze Oracle bude sloužit jako databáze - sklad zboží.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

***Zakhour S. a kol.: Java 6 - Výukový kurz. Computer Press, 2007.**
***Prata S.: Mistrovství v C++. Computer Press, 2007.**
***Richter J.: .NET Framework - programování aplikací. Computer Press, 2002.**
***OpenSwing User Manual (English version)**

Vedoucí bakalářské práce: **Ing. Zdeněk Šilar**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. ledna 2010**

Termín odevzdání bakalářské práce: **14. května 2010**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2010

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 09. 05. 2010 Lukáš Voves

Poděkování

Chtěl bych tímto poděkovat panu Ing. Zdeňku Šilarovi za to, že mi umožnil zvolit si vlastní téma práce a pracovat na něm. Dále bych chtěl poděkovat panu Ing. Milanu Miklošovi, který mne seznámil s frameworkem OpenSwing a předal mnoho cenných rad. A nakonec bych chtěl poděkovat své rodině za podporu a trpělivost během studia i při zpracování této práce.

Anotace

Práce je věnována vývoji desktopových aplikací ve firmách. Jsou zde nastíněny některé programovací jazyky, které mohou být použity při vývoji aplikací. Déle jsou představeny databáze, se kterými mohou firemní aplikace spolupracovat a frameworky, které programování těchto aplikací zjednodušují. Praktická část je pak věnována aplikaci Dealer, která je naprogramována v programovacím jazyce Java s použitím frameworku OpenSwing.

Klíčová slova

OpenSwing, Java, Oracle, JavaDB, desktopové aplikace, GUI aplikace, Swing, MDI, SDI

Title

The development of corporate desktop applications using the framework OpenSwing

Annotation

Work is devoted to the development of desktop applications in companies. Outlined here are some programming languages that can be used in developing applications. Longer presents a database with which business applications can collaborate and frameworks that simplify programming of these applications. The practical part is devoted to the application of the dealer, which is programmed in Java using the framework OpenSwing.

Keywords

OpenSwing, Java, Oracle, JavaDB, desktop applications, GUI applications, Swing, MDI, SDI

Obsah

Seznam zkratek

API	Application Programming Interface. Rozhraní pro programování aplikací.
AWT	Abstract Window Toolkit. Rozhraní pro tvorbu GUI aplikací.
CIL	Common Intermediate Language. Byte-kód, do kterého se překládají jazyky pro .NET framework.
CLR	Common Language Runtime. Společné běhové prostředí je součást .NET frameworku.
CLS	Common Language Specification. Společná jazyková specifikace.
CTS	Common Type Systém. Unifikovaný typový systém.
GPL	General Public License. Licence pro svobodný software.
GUI	Graphical User Interface. Grafické uživatelské rozhraní.
IDE	Integrated Development Environment. Integrované vývojové prostředí. Software usnadňující práci programátorům.
J2EE	Java Platform, Enterprise Edition. Verze Javy určená k tvorbě a provozu podnikových aplikací a informačních systémů.
J2ME	Java Platform, Micro Edition. Verze Javy určená pro vývoj aplikací pro malá zařízení a zařízení s omezenými prostředky, jako jsou mobilní telefony a kapesní počítače.
J2SE	Java Platform, Standard Edition. Základní verze Javy určená pro tvorbu konzolových aplikací a aplikací s grafickým rozhraním.
JDBC	Java Database Connectivity. Rozhraní pro jednotný přístup k databázím.
JDK	Java Development Kit. Soubor základních nástrojů pro vývoj aplikací pro platformu Java.
JPA	Java Persistence API. Framework programovacího jazyka Java, který umožňuje objektově relační mapování. To usnadňuje práci s ukládáním objektů do databáze a naopak.
JRE	Java Runtime Environment. Sada knihoven Java API a běhového prostředí JVM pro spuštění aplikace napsané v Javě.
JVM	Java Virtual Machine. Virtuální stroj jazyka Java, který umí spouštět tzv. byte-kód, který je vytvořen ze zdrojových kódů jazyka Java.

LGPL	GNU Lesser General Public License. Licence svobodného software, která na rozdíl od GPL nevyžaduje, aby software, který využívá knihovnu, nemusel mít licenci (L)GPL.
LINQ	Language Integrated Query. Integrovaný dotazovací jazyk, který přináší podobné možnosti, jako SQL, do programovacího jazyka.
MDI	Multiple Document Interface. Metoda pro organizaci vnitřních oken aplikace, která nejsou individuální a mají společný rodičovský formulář.
MVC	Model-view-controller. Softwarová architektura, která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent tak, že modifikace některé z nich má minimální vliv na ostatní.
RAD	Rapid Application Development. Automatické vytvoření kostry zdrojového kódu na základě vizuálního návrhu GUI.
SDI	Single Document Interface. Metoda pro organizaci GUI aplikací do individuálních oken, která jsou od sebe oddělená.
SQL	Structured Query Language. Strukturovaný dotazovací jazyk používaný pro práci s daty v relačních databázích.
WPF	Windows Presentation Foundation. Součást .NET frameworku pro vytváření moderního uživatelského rozhraní pomocí značkovacího jazyka XAML.
XAML	Extensible Application Markup Language. Deklarativní jazyk založený na XML.

Seznam obrázků

1 Úvod

Desktopové aplikace jsou důležitou součástí každé větší firmy. Ať už jde třeba o vnitřní informační systém, seznamy skladových zásob nebo zprávu objednávek. Velmi moderní jsou dnes webové aplikace, které se snaží převzít úlohy desktopových aplikací a přenést je do webového prohlížeče. Některé firmy možná sáhnou po této volbě, ale ti, kteří se chtějí odpoutat od webového prohlížeče, zajisté sáhnou po desktopové verzi.

Webové aplikace mají výhodu v nezávislosti na platformě, ale zase na druhé straně musejí být schopné běžet správně v různých prohlížečích. V případě webové aplikace se ke klientovi stahuje pouze zobrazovací část a veškerá logika a datová část pracuje na straně serveru. Velkou výhodou je správa aktualizací a oprav, která je pouze na jednom místě. Webové aplikace nejsou většinou náročné na výkon počítače a tak je lze spustit prakticky kdekoliv. Nevýhodou může být závislost na webovém prohlížeči a jeho stabilitě.

Desktopové aplikace jsou naproti tomu ve většině případů závislé na platformě a jejich velikost je větší, protože mají v sobě i logickou a datovou část. Je možné ale logiku i datovou část také oddělit, takže se velikost aplikace nakonec také může zmenšit. Výhodou desktopových aplikací může být například stabilita, možnost tvorby složitějších programů nebo rychlost. Některé programovací jazyky mohou být také nezávislé na platformě, tudíž nemusíme aplikaci kompilovat pro každou platformu zvlášť. Záleží pouze na výběru.

Cílem této práce je ukázat, jak s pomocí OpenSwing frameworku vytvořit firemní desktopovou aplikaci napsanou v Javě. Možnosti budou demonstrovány na aplikaci Dealer, která jako hlavní databázi využívá JavaDB neboli Apache Derby a jako pomocný sklad databázi Oracle.

V práci budou představeny programovací jazyky, které je možné při programování desktopových aplikací využít, databáze, které s těmito programy mohou spolupracovat a frameworky, které v těchto programovacích jazycích usnadní vývoj.

2 Požadavky pro tvorbu firemní desktopové aplikace

Na rozdíl od běžných aplikací, které tvoří programátoři například na zakázku, kdy si může programátorský tým, který danou aplikaci tvoří, vybrat, jakým programovacím jazykem aplikaci naprogramovat, jakou databázi využít a jakým stylem a za jak dlouho aplikaci naprogramovat, musí IT oddělení ve větší firmě pracovat trochu jinak.

Programátorský tým si nemůže usmyslet, že bude dělat v jazyce, který se mu nejlépe hodí pro aplikaci, kterou má momentálně naprogramovat nebo si vybrat databázi, se kterou bude ten daný jazyk nejlépe spolupracovat.

Ve firemním prostředí je většinou už zaveden nějaký jednotný programovací jazyk, který by měli ovládat všichni programátoři a také databázový stroj. Součástí týmu mohou být ale i programátoři, kteří umí jiný programovací jazyk, který se využíval před tím současným a jsou i tak velice důležitou součástí týmu.

Většina aplikací, které se naprogramují, jsou totiž aplikace pro interní správu skladů, objednávek, fakturací atd., které je nutné průběžně aktualizovat, opravovat a udržovat v bezchybném stavu. Nemůže se stát, že například přestane fungovat aplikace pro kontrolu skladových zásob, která je naprogramovaná v prostředí Delphi, ale není nikdo, kdo by ji mohl opravit, protože ten, kdo ji vytvářel, už ve firmě nepracuje a nikdo jiný tomu nerozumí. Proto je nutné mít vždy ve svém týmu alespoň jednoho člověka, který si s daným problémem poradí a zprovozní chybnou aplikaci.

Velmi vhodnou volbou by se mohlo zdát vytvoření nové aplikace v aktuálně používaném jazyce, ale vzhledem k počtu požadavků na IT oddělení v informačním systému a například rozsáhlosti aplikace není vždy možné vydat se touto cestou. Pokud se však jedná o nějaký menší nebo pomocný program, v případě dostatku času je vhodné takovou aplikaci nahradit za novou, kterou bude možné snadněji spravovat.

Hlavními požadavky pro tvorbu firemních desktopových aplikací jsou tedy rychlost, spolehlivost, kvalita, bezchybnost a uživatelská přívětivost. Rychlost je zde uvedena hlavně proto, že je důležité, aby programátorský tým měl nastavena určitá pravidla a pracoval tak efektivně. Není žádoucí, aby v případě problému, programátor strávil dlouhou dobu jeho hledáním a odstraňováním. Proto jsou většinou součástí postupu určité šablony, návrhové vzory nebo jednotné postupy, které práci zjednodušují, zrychlují a hlavně minimalizují množství chyb.

Základy většiny programů jsou totiž stejné a tak je zbytečné vše programovat úplně od začátku, když je daleko rychlejší a jednodušší pouze upravit připravenou šablonu a předělat ji pro potřeby dané aplikace.

3 Programovací jazyky

Programovacích jazyků je celá řada. Ne každý se však hodí pro programování desktopových aplikací. Jelikož jsou firemní desktopové aplikace většinou s grafickým uživatelským rozhraním, nehodí se pro jejich programování strukturovaně orientované programovací jazyky. Naše volba by tedy měla padnout na jazyky objektově orientované. Každý jazyk má nějaká svá pro i proti a neexistuje žádný nejlepší programovací jazyk. Je jen na programátorovi, pro který jazyk se rozhodne a který bude pro jeho práci ten nejlepší. Každý jazyk se hodí na něco jiného a do jiné situace.

3.1 Jazyk C++

V 70. letech udaly jazyky C a Pascal směr v oblasti strukturovaného programování (PRATA, 2001). Kromě strukturovaného programování byla kladnou vlastností jazyka C i schopnost produkovat rychle běžící programy a možnost adresovat komunikační porty a ovladače disků. Jazyk C se tak stal oblíbeným jazykem 80. let. Začátkem 70. let pracoval Dennis Ritchie z Bell Laboratories na rozvoji operačního systému UNIX. Chtěl tento systém zpřístupnit většímu počtu lidí, a tudíž se rozhodl nepoužít assembler, který je svázán se strojovým kódem počítače. Ritchie chtěl vytvořit jazyk, který by uměl přistupovat k hardware na nízké úrovni, ale zároveň aby byl jazykem vyšší úrovně a byl přenositelný. A tak vytvořil jazyk C.

Jazyk C++ vznikl také v laboratořích Bell. Počátkem 80. let ho vytvořil Bjarne Stroustrup. Název C++ pochází z přírůstkového operátoru, který přičítá k původní hodnotě jedničku. Tím zachovává odkaz na jazyk C, který rozšiřuje. Jazyk C++ rozšiřuje jazyk C hlavně o rysy OOP a šablony. Většina kompatibility je však zachována. Pokud tedy napíšeme program v jazyce C, měl by být ve většině případů platný i v C++. Programy napsané v C++ mohou využívat knihovny jazyka C.

Jazyk C++ zachovává tradici efektivity, uceleného, rychlého a přenositelného jazyka C. Navíc ale přidává principy objektově orientovaného programování rozšířením o třídy C++ a programování pomocí šablon jazyka C++. Pokud se přechází z jazyka C na jazyk C++, je třeba se odnaučit některým programovým zvyklostem.

Jazyk C se zabývá daty a algoritmy, kdy data jsou informací, kterou program používá a zpracovává a algoritmy jsou metody, které používá. Jejich spojením vzniká program. Aby se zamezilo nepřehlednosti například při používání skoků, používá se strukturovaný styl programování.

Jazyk C++ přidává nový přístup k programování, OOP. Místo zdůrazňování algoritmů, jako v případě strukturovaného programování, OOP zdůrazňuje data. V C++ se setkáváme se slovy class a object. Class, neboli třída, obecně definuje všechny údaje a činnosti, které mohou být nad těmito údaji prováděny. Object je pak konkrétní instance třídy, která má už specifické vlastnosti. Třída může být například člověk, který se nějak

jmenuje a může provádět činnost jíst. Objekt potom může být například člověk, který se jmenuje Josef Novák a provádí konkrétní činnost jíst, když jí snídani.

Jazyk C++ přidává k OOP ještě další programovací postup, a tím je generické programování. Zatímco OOP se zaměřuje hlavně na data, programování pomocí šablon se zaměřuje na algoritmy. Programování pomocí předloh poskytuje typově nezávislé provádění společných úloh, jako je například třídění nebo slučování dat. Data se vyskytují v mnoha typech, a pokud bychom je například chtěli porovnat, museli bychom pro každý typ vytvořit odlišné funkce. Při programování pomocí předloh tak lze napsat jednu šablonu pro tzv. všeobecný typ. Při použití pro určitý typ pak jen stačí dát šabloně vědět, o jaký typ jde.

Když se řekne C++ a vývoj GUI aplikací, tak to v základu neznamená nic dobrého. C++ standard neposkytuje pro návrh GUI nic. Naštěstí je k dispozici mnoho frameworků, které tento nedostatek odstraňují a přidávají tak jazyku C++ podporu pro návrh GUI aplikací. Jsou to například Windows Forms[1], GTK[2], MFC[3], QT, WTL[4] nebo wxWidgets. Použití určitého frameworku ale závisí také na podpoře IDE, které používáme. Pokud používáme Microsoft Visual Studio ve verzi Standard a vyšší, můžeme v klidu využít frameworky .NET, MFC, QT nebo WTL, které mají podporu grafického návrháře GUI aplikací.

3.2 Jazyk C#

První verze jazyka C# (WIKIPEDIA, 2010c) byla vydaná v roce 2002 společně s .NET frameworkem, pro který bylo optimalizováno. Obsahovalo podporu objektově orientovaného programování a vycházelo z jazyků C++ a Java. Další verze přinesla podporu generik, neboli parametrizovatelných šablon jako v jazyce C++. Na rozdíl od něj jsou ale instanciovány za běhu a ne při kompilaci, takže je možné je použít i v jiném jazyce, než byly napsány. Možnosti C# byly dále rozšířeny i o iterátory a částečné a statické třídy. Částečná třída je taková, která je například rozdělená do více souborů, kdy jedna část může být například vygenerována automaticky a druhá napsaná programátorem. Dále pak byla přidána podpora pro nullovatelné typy a LINQ, jazyk podobný syntaxi SQL, rozšiřující metody, které umožňují přidat metody do již existujících tříd a ty potom volat jako metody instance tak, jako by byly součástí třídy samotné. Zajímavé je i použití klíčového slova var, které můžeme zapsat místo deklarace proměnné a ta se poté doplní podle pravé strany při překladu. Tím se nám zkrátí zápis pro inicializaci proměnných.

Název jazyka je odvozen od hudební notace a # označuje zvýšení o půl tónu. Je to jakýsi znak rozšíření podobně jako ++ u jazyka C++.

[1] API, které je součástí .NET frameworku
[2] GIMP Toolkit
[3] Microsoft Foundation Classes
[4] Windows Template Library

C# je moderní objektově orientovaný jazyk, který je určen nejenom pro zařízení se složitějšími operačními systémy, ale i pro omezenější zařízení. Jazyk poskytuje například podporu pro hlídání hranic polí, automatický garbage collector, jako jazyk Java, nebo kontrolu neinicializovaných proměnných. V C#, podobně jako v jazyce Java neexistuje vícenásobná dědičnost. Rozhraní ale můžeme implementovat libovolné množství. Dále neexistují žádné globální proměnné, ani metody. Místo nich jsou statické metody a proměnné veřejných tříd. C# velmi dbá na bezpečnost typové konverze, a tak je možné používat pouze bezpečné implicitní konverze, jako je například konverze z odvozené třídy na rodičovskou. V jazyce C# není potřeba dopředné deklarace metod. Jazyk C# také rozlišuje velká a malá písmena, takže identifikátory `Promenna` a `promenna` jsou rozdílné. C# používá, jako všechny jazyky pod .NET frameworkem, unifikovaný typový systém zvaný CTS. Pro vývoj GUI se používá součást .NET frameworku, WPF.

3.3 Object Pascal a Delphi

Historie jazyka Object Pascal (WIKIPEDIA, 2010a) sahá do roku 1985, kdy Niklaus Wirth a Larry Tesler vytvořili rozšíření programovacího jazyka Pascal o objektově orientovaný přístup. Object Pascal vznikl za účelem vývoje aplikačního rozhraní pro počítače Macintosh. V roce 1986 přidala firma Borland obdobná rozšíření do své implementace jazyka Pascal jménem Turbo Pascal pro počítače Macintosh a později v roce 1989 i do verze pro DOS. Jazyk poté přejmenovala na Object Pascal.

Delphi (KADLEC, 2001), jehož první verzi představila roku 1995 firma Borland, je vývojové prostředí založené na jazyce Object Pascal. Ten se od jiných implementací liší zavedením klíčového slova class místo object a zavedením konstruktoru Create a virtuálního destruktoru Destroy. Je to prostředek, který maximálně usnadní vývoj aplikace. Některé části kódu vygeneruje za programátora samo vývojové prostředí.

Delphi umožňuje:

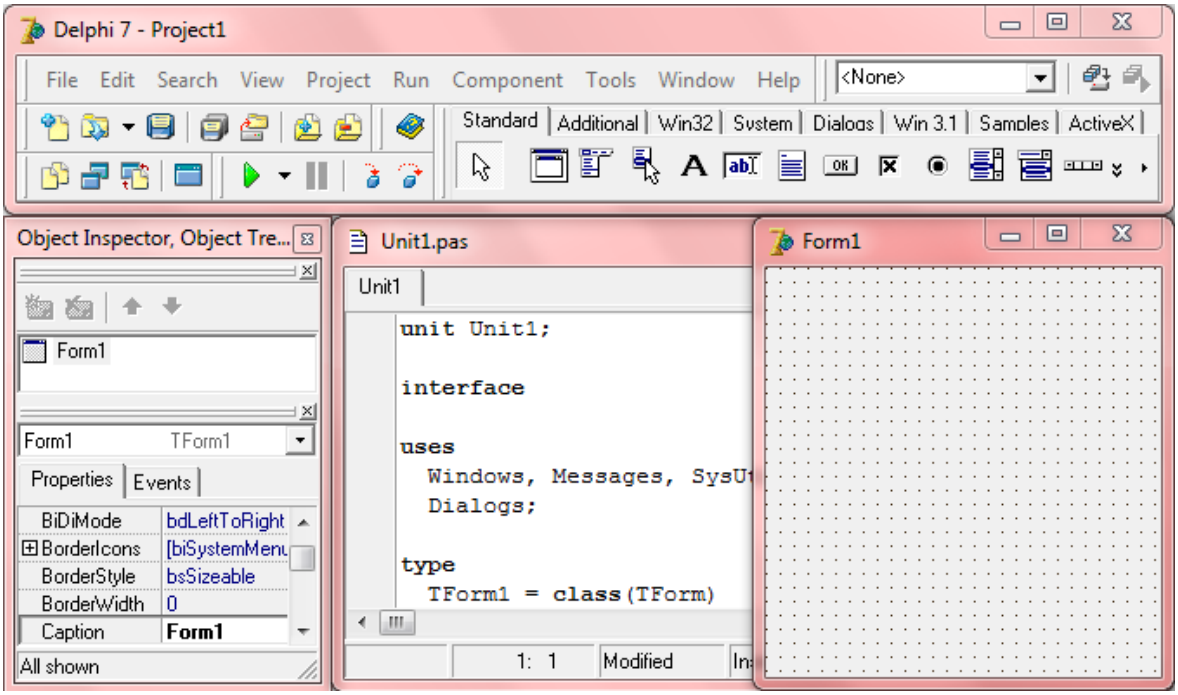
- co nejjednodušší a nejrychlejší návrh, díky možnosti použít velké množství komponent, které usnadní některé činnosti,
- co nejrychlejší a nejefektivnější výsledek práce, díky velmi výkonnému kompilátoru,
- co největší možnost znovuvyužití zdrojového kódu.

Dále nabízí velmi dobrou podporu databází, množství integrovaných nástrojů, jako je například TeamSource pro správu činností, možnost vytvářet aplikace klient/server, možnost vytvářet vlastní komponenty a od verze 8 nabízí také podporu pro kompilaci aplikací do .NET CIL.

Jednou ze slabých stránek Delphi (WIKIPEDIA, 2010b) je chybějící podpora pro vývoj nativních 64 bitových aplikací, takže není možné použít více než 4 GiB paměti. Také není možné psát doplňky pro jiné 64 bitové aplikace. Další slabou stránkou je, že Delphi jako takové není nezávislé na platformě, takže lze vyvíjet aplikace pouze pro Windows.

Stinnou stránkou pak může být lpění na zpětné kompatibilitě, která zajišťuje, že většina projektů, která byla vytvořena v dřívějších verzích Delphi, bude v nových verzích funkční. Tato kompatibilita může být nespornou výhodou, ale také krokem zpět v případném vývoji.

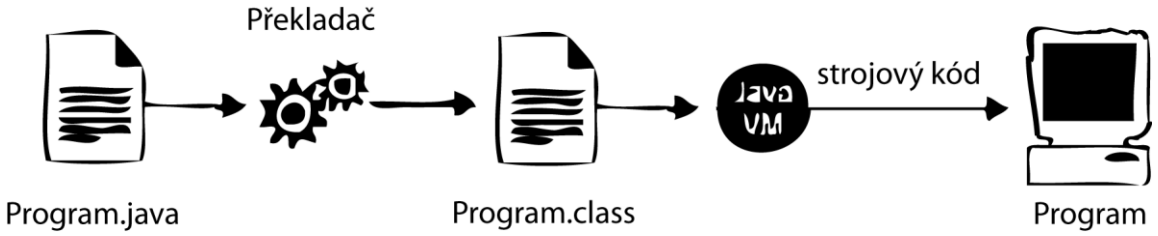
Jednou z nejoblíbenějších verzí Delphi se stala verze 7 (Obrázek 1), která přinesla vysokou stabilitu, rychlost a nízkou náročnost na hardware. Tato verze je velkým množstvím programátorů používána dodnes. Vývojové prostředí Delphi v současnosti vlastní a vyvíjí společnost Embarcadero Technologies a jeho zatím poslední vydaná verze je 2010.



Obrázek 1 – Prostředí vývojového prostředí Delphi 7. Zdroj vlastní.

3.4 Jazyk Java

Platformu Java (ZAKHOUR, a další, 2007) vyvinula a představila firma Sun Microsystems v roce 1995. Součástí technologie Java je programovací jazyk a platforma Java. Programovací jazyk Java je vyšší objektově orientovaný programovací jazyk, který je jednoduchý, nezávislý na architektuře, přenositelný, dynamický a zabezpečený.



Obrázek 2 – Schéma kompilace a spuštění aplikace. Zdroj vlastní.

V programovacím jazyce Java se zdrojový kód (Obrázek 2) ukládá do souborů s příponou `.java`. Kompilátor `javac` pak tyto zdrojové soubory kompiluje do souborů `.class`. Ty obsahují byte-kód, který je spustitelný v JVM, což je virtuální stroj jazyka Java. JVM je k dispozici pro mnoho různých operačních systémů a tak lze stejné `.class` soubory spustit jak v Microsoft Windows, tak i třeba v Linuxu nebo MacOS.

Platforma Java je softwarové prostředí, které umožňuje spouštět programy napsané v Javě. Platforma Java obsahuje 2 komponenty:

- JVM – virtuální stroj jazyka Java,
- API – Aplikační programové rozhraní.

Platforma Java je vlastně takový můstek mezi operačním systémem a programem. API je sbírka knihoven, které poskytují rozsáhlou sbírku komponent, tříd a rozhraní. Kvůli nutnosti běhu programu v JVM není jeho rychlost vyšší než rychlost programů zkompilovaných do nativního kódu, ale díky vývoji se tento rozdíl minimalizuje.

Technologie Java (KISZKA, 2003) nabízí spoustu přínosů, jako je jednodušší psaní kódu, které je jednodušší než programování v C nebo C++. Díky Garbage collectoru se programátor nemusí starat o práci s pamětí a její úklid. Ukazatele byly nahrazeny odkazy a nehrozí tak zápis do neplatné paměti. Díky serializaci je přenášení dat po síti nebo ukládání do souborů příjemnější. Systém objektových výjimek umožňuje nejen ošetření všech chyb, ale i zachycení v jediném bloku `try-catch`. Díky zabezpečení sandbox je možné zajistit bezpečnost dat, kdy žádný program stažený z internetu nemůže zformátovat disk nebo zavést viry. Velkou výhodou Javy je i její dokumentace. Díky technologii JavaDoc je dokumentace přehledná a lze se v ní velice snadno orientovat. Další nespornou výhodou je nezávislost a přenositelnost programu napsaného v Javě. Díky tomu, že se aplikace kompilují do byte-kódu, který je interpretován překladači JIT a je nezávislý na počítači, fungují programy prakticky všude, kde je k dispozici platforma Java.

Platforma Java se skládá z několika edic. Základní edicí je J2SE, Java 2 Standard Edition, která je určená k vývoji konzolových aplikací a aplikací s grafickým uživatelským rozhraním. Další edice je J2EE, Java 2 Enterprise Edition, která se používá pro tvorbu podnikových aplikací. Pro tvorbu aplikací pro mobilní telefony a kapesní počítače je pak určena J2ME, Java 2 Micro Edition. Pro tvorbu bohatých internetových aplikací, tzv. RIA, je k dispozici celkem mladý přírůstek do rodiny, který se jmenuje JavaFX.

Java platforma obsahuje množství grafických komponent, které mohou být využity k tvorbě grafického uživatelského rozhraní. Už v základu jsou k dispozici knihovny AWT (Abstract Window Toolkit) a Swing. Pro uživatele IDE Eclipse je pak k dispozici ještě knihovna SWT (Standard Widget Toolkit).

3.4.1 Knihovna AWT

Knihovna Abstract Widget Toolkit (KEOGH, 2005) je k dispozici od samého počátku, ale místo pochvaly se na ni snesla spíše vlna kritiky. Je to hlavně proto, že tato knihovna využívá nativních komponent operačního systému a tudíž porušuje základní vlastnosti Javy, kterou je přenositelnost a nezávislost na platformě. Při tvorbě GUI se tak už většinou grafické komponenty AWT moc nepoužívají. Ale jelikož knihovna AWT obsahuje podařené metody pro odchytávání a zpracování událostí, generovaných GUI komponentami, balíček `java.awt.event` můžeme najít velice často.

3.4.2 Knihovna Swing

Knihovna Swing, která může být známa i pod názvem Java Foundation Classes (JFC), byla představena po AWT. Obsahuje bohatší kolekci grafických tříd než AWT. Knihovny AWT a Swing se navzájem doplňují, takže se v programech můžeme setkat s oběma. Knihovna Swing eliminuje neduhy knihovny AWT, takže je napsána kompletně v Javě a již nespolupracuje s nativními komponentami operačního systému, vzhled byl oddělen od funkcionality, takže můžeme pomocí LookAndFeel definovat, jaký vzhled bude použit, nebo můžeme vytvořit vlastní. Knihovna Swing je ale oproti AWT více náročná na paměť.

3.4.3 Knihovna SWT

Standard Widget Toolkit (WIKIPEDIA, 2009b) je knihovna grafických komponent pro platformu Java. Původně byla vyvinuta firmou IBM, která ji později poskytla Eclipse Foundation spolu s IDE Eclipse. Je to alternativa k AWT + Swing, která byla vyvinuta hlavně kvůli snížení nároků na paměť. Náročnost je ale zhruba podobná, jako v případě Swing knihovny. K vykreslování komponent využívá na rozdíl od Swingu nativní knihovny operačních systémů pomocí JNI[5]. Knihovnu SWT lze najít hlavně v IDE Eclipse a jeho různých klonech.

[5] JNI, neboli Java Native Interface je rozhraní umožňující propojit kód napsaný v Javě s nativními programy a knihovnami napsanými v jiných jazycích.

4 Databáze

Moderní firemní desktopové aplikace se většinou neobejdou bez uložení informací, se kterými pracují, v relační databázi. Databáze jsou dnes všude. Na webu je nejoblíbenější databáze MySQL, která je sice nenáročná na výkon, ale pro firemní aplikace se nehodí kvůli nižší bezpečnosti a jednoduchosti. Ve firemním prostředí jsou kvalita, dostupnost, spolehlivost a bezpečnost standardní vlastnosti. Pro firemní aplikace se musí použít některá z pokročilejších databází, jako například Oracle Database. Pro jednodušší aplikace nebo aplikace využívající embedded databázi je v případě vývoje v IDE NetBeans výhodné použít databázi JavaDB, neboli Apache Derby.

4.1 Apache Derby (JavaDB)

První verze Apache Derby byla představena roku 1997 společností Cloudscape pod názvem Cloudscape. V roce 1999 přešel produkt pod firmu IBM a v roce 2004 poskytla firma IBM zdrojové kódy Apache Software Foundation, která nadále vyvíjí databázi pod názvem Derby. Apache Derby je velice úsporný relační databázový systém napsaný v Javě. Jeho největší předností je malá stopa. V paměti zabírá základní engine a JDBC Driver pouze 2 MiB. Derby je možné díky embedded JDBC Driveru použít jako vestavěnou databázi aplikací napsaných v Javě. Derby je možné používat i ve standardní variantě klient/server. Apache Derby je také součástí vývojového prostředí NetBeans, kde se vyskytuje pod názvem JavaDB.

4.2 Informix

IBM Informix Dynamic Server (WIKIPEDIA, 2009a) je rozšiřitelný relační databázový systém, původně vyvinutý firmou Informix Software, který nyní vlastní společnost IBM Software. Aktuální verze je 11.50 a je k dispozici v několika variantách:

- Informix Dynamic Server Express Edition je k dispozici pouze pro 32 bitové systémy s podporou maximálně 2 procesorů, 4 GiB operační paměti a bez podpory rozšiřujících modulů DataBlade, určená pro malé a střední firmy,
- Informix Dynamic Server Workgroup Edition, která je omezena na 4 procesory a 8 GiB operační paměti a je určena náročnějším uživatelům z řad středně velkých firem,
- Informix Dynamic Server Enterprise Edition je nejvyšší edice, která je plně funkční a je zaměřena na velké firmy.

4.3 Oracle

Oracle Database (WIKIPEDIA, 2010f) je relační databázový systém vyvinutý společností Oracle Corporation. Nejnovější verzí databáze je Oracle Database 11g. Společnost Oracle drží prvenství v uvedení první komerční SQL databáze (1979), první 64 bitové databáze (1995) nebo první databáze s podporou XML (1999).

Oracle nabízí svůj databázový systém v několika různých verzích:

- Express Edition, neboli Oracle XE, je nejnižší verze, která funguje pouze na 32 bitových systémech s využitím maximálně 1 procesoru, 4 GiB místa na disku a 1 GiB operační paměti,
- Standard Edition One je verze, která může být nasazena na stroji, který má maximálně 2 procesory, ale nemá žádné omezení paměti,
- Standard Edition (SE) můžeme nainstalovat na servery s maximálním počtem 4 procesorů bez omezení paměti; v této verzi můžeme využít i seskupení s Oracle RAC (Real Application Clusters), kdy můžeme mít například až 4 servery s jedním procesorem spojené do clusteru,
- Enterprise Edition (EE) je určena pro servery s více jak 4 procesory, má více funkcí než verze SE hlavně v oblasti zabezpečení a výkonu a může využívat seskupení serverů s podporou technologie Oracle RAC bez omezení

Databázové stroje Oracle vynikají svou spolehlivostí a je tudíž možné je bez problému nasadit i v náročnějších prostředích, kde je kladen důraz na spolehlivost, vysoký výkon, neustálou dostupnost, bezúdržbový chod a to hlavně ve verzi EE. Velkou výhodou databázových systémů od Oracle je jejich škálovatelnost a velmi snadná přenositelnost, kdy může systém běžet jak na strojích s operačním systémem Windows nebo Linux, tak i na sálových počítačích nebo serverech.

5 Frameworky

Aby bylo psaní kódu efektivní, vznikly struktury zvané frameworky. Ty v sobě mohou obsahovat různé návrhové vzory, postupy při vývoji nebo knihovnu API. Programátor se například díky použití frameworku nemusí zabývat pro něj nedůležitými věcmi, jako je řazení sloupců v tabulce nebo export obsahu tabulky do Excelu. Takovéto funkce nemusí zbytečně programovat a ztrácet tak čas, který potřebuje na řešení zadaného problému, a tím velice zefektivní svoji práci.

Ze začátku je použití frameworku časově náročnější, protože se s ním musí programátor nejprve seznámit a pochopit, jak pracuje. Prvních pár projektů je díky tomu pomalejších, ale jakmile si na framework zvykne a naučí se s ním pracovat, jeho produktivita se mnohonásobně zvýší. Občas bývají námitky směřovány k rychlosti aplikace při použití frameworku, jelikož je to kód, nad kterým programátor většinou nemá kontrolu. Tyto námitky jsou však pro firemní desktopové aplikace zbytečné, protože většina takových aplikací není tolik náročná, vzhledem k dostupnosti výkonného hardware.

5.1 .NET

Platforma .NET (PUŠ, 2004) byla poprvé představena v roce 2000 a první verze byla vydána společností Microsoft v roce 2002 společně s jazykem C#. .NET je platforma pro vývoj aplikací pro operační systém Windows, která používá k běhu aplikací řízené běhové prostředí podobné JVM. Hlavními důvody pro vývoj platformy byly:

- Nekompatibilita mezi jednotlivými jazyky a jejich obtížná spolupráce,
- Vysoká chybovost aplikací při práci s pamětí nebo při konverzi datových typů,
- Problémy s rozdílnými verzemi knihoven,
- Zastaralý a nepřehledný vývoj webových aplikací.

Všechny tyto problémy byly vyřešeny s příchodem .NET platformy a řízeného běhového prostředí. Standardní aplikace jsou zkompilovány kompilátorem do strojového kódu počítače a program je tak použitelný pouze na dané platformě. Pro jinou musí být znova zkompilován. Výhodou je ve většině případů rychlý běh aplikací. Problém ovšem může nastat, pokud je aplikace zkompilována pro určitou verzi operačního systému, ale na novější už neběží. Musí se tedy překompilovat nebo případně předělat i na novější verzi. Řízené běhové prostředí tento problém řeší tak, že mezi zdrojovým kódem a strojovým kódem je ještě byte-kód. Ten je spustitelný řízeným běhovým prostředím na dané platformě, kde je tento byte-kód běhovým prostředím převeden na strojový kód. Tento způsob je ale náročnější na výkon počítače a tak se nepoužívá například pro hry, kde jsou výpočetní operace složitější. Velice se hodí například pro vývoj firemních aplikací, kde náročnost programu většinou není tak vysoká a přenositelnost aplikace je důležitější než menší ztráta výkonu. U těchto aplikací se počítá s tím, že se spousta částí kódu používá opakovaně a jelikož se aplikace překládá pomocí JIT kompilace, při které se přeloží pouze používaná část a ne celá aplikace najednou, tak po znovupoužití již přeloženého kódu aplikace běží srovnatelně rychle s klasickými aplikacemi.

Byte-kód používaný platformou .NET se nazývá CIL. Tento jazyk je spouštěn součástí .NET frameworku zvanou CLR. V prostředí CLR je implementován Garbage Collector, podobně jako u platformy Java. Garbage Collector se stará o přidělování nebo uvolňování operační paměti a odpadá tak riziko špatné práce s ní, kdy občas dochází k pádu aplikace.

Důležitými součástmi platformy jsou také CLS, neboli společná jazyková specifikace a CTS, společný typový systém. Při dodržení specifikace CLS a CTS mohou ostatní výrobci přizpůsobit svůj programovací jazyk platformě .NET. Důležité je také, aby měl jazyk dostupný kompilátor schopný jazyk zkompilovat do byte-kódu CIL. Jazyky použitelné pro vývoj aplikací pro .NET platformu jsou C#, Visual Basic .NET, J# (rozšíření jazyka Java), managed C++, neboli řízené C++, nebo Delphi .NET.

S pomocí .NET frameworku můžeme vytvářet konzolové aplikace, aplikace využívající uživatelské rozhraní založené na Windows Forms, které využívá Win32 API, službu systému Windows, neboli aplikaci běžící na pozadí jako proces, nebo například webové aplikace založené na ASP.NET.

.NET platforma (WIKIPEDIA, 2010e) je dostupná v několika verzích:

- Microsoft .NET Framework – platforma pro osobní počítače s operačním systémem Windows 98 a vyšší,
- Microsoft .NET Compact Framework – platforma určená pro mobilní telefony a kapesní zařízení s operačním systémem Windows Mobile,
- Microsoft .NET Micro Framework – platforma určená pro embedded zařízení s omezeným výkonem a výpočetní kapacitou,
- Mono – nezávislá open-source platforma pro operační systémy na bázi UNIXu (Linux, MacOS).

5.2 OpenSwing

OpenSwing framework (CARNIEL, 2007) je sada pokročilých grafických komponent na základě Swing Toolkit, vyvíjená od roku 2005 Maurem Carnielem. Tyto komponenty jsou složitější než ty poskytované Swingem. Je možné s nimi pracovat uvnitř grafického návrháře v IDE. Framework je distribuován pod licencí open-source[6]. Pro komerční projekty je pak k dispozici pod licencí LGPL. První verze frameworku přišla na svět v roce 2007 a od té doby se neustále vyvíjí. V současné době je k dispozici verze 2.1.7, která vyšla na konci února 2010. Od verze 2.0.6, která vyšla na začátku srpna 2009 je k dispozici i můj vlastní český překlad.

Součástí OpenSwing frameworku ale není jen sada grafických komponent, ale i knihovny, které s nimi spolupracují. Je to také framework, který poskytuje mechanismus navázání spojení mezi komponentami a datovým modelem, založený na MVC. Datový

[6] Open source je počítačový software s otevřeným zdrojovým kódem.

model je založen na Java Beans[7] a je podporován ve všech OpenSwing komponentách, jako je například ovladač mřížky. OpenSwing framework poskytuje kompletní řešení pro rychlý a snadný vývoj bohatých desktopových aplikací.

OpenSwing obsahuje kolekci tříd, které mohou být použity:

- K vytvoření front-endu aplikace pomocí grafických komponent srovnatelných s tradičními RAD vývojovými prostředími, jakými je například Visual Basic nebo Delphi. Grafické komponenty obsahují více než 30 widgetů. Použití mřížky je obzvláště sofistikované a snadné. Vlastnosti jednotlivých sloupců je možné nastavovat přímo v grafickém návrháři v IDE. Je možné tak jednoduše nastavit třídění, filtrování, viditelnost nebo například možnosti importu a exportu. Grafické komponenty dodržují specifikace Java Beans, takže mohou být použity v grafickém návrháři IDE, jako je JDeveloper, NetBeans nebo Eclipse a je možné s nimi pracovat podobně, jako například v Delphi. Můžeme vytvářet aplikace založené jak na SDI, tak i na MDI, s menu a dalšími nastaveními.
- K vytvoření vrstvy business logiky a vrstvy pro přístup k datům, prostřednictvím souboru nástrojů, které zjednodušují proces vývoje. Tuto vrstvu však lze nahradit i jinými frameworky pro práci na straně serveru, jako je Spring, Hibernate nebo další. OpenSwing poskytuje soubor nástrojů pro snadnou integraci těchto frameworků. Dále také poskytuje nástroje pro zjednodušení integrace JPA.
- K vytvoření komunikační vrstvy, která bude umístěna mezi prezentační vrstvu (front-end) a vrstvu business logiky. Vrstva může být snadno rozšířena vytvořením vlastní komunikační vrstvy nad standardní vrstvou nabízenou frameworkem OpenSwing.

OpenSwing framework navíc poskytuje některé základní funkce eliminující mnoho problémů, které vznikají při vývoji firemních aplikací, jako je export dat z mřížky, import dat do mřížky, prohlížení dokumentů v různých formátech, jako je Excel, CVS, XML, RTF, PDF nebo HTML. Framework OpenSwing lze použít s Javou 1,4, 1,5 nebo 1,6.

[7] Java Beans jsou opakovaně použitelné softwarové komponenty pro Javu, kterými lze vizuálně manipulovat v GUI návrháři. Jsou používány, aby zapouzdřily více objektů do jednoho. Jsou serializovatelné, mají bezparametrický konstruktor a přístup k proměnným je realizován prostřednictvím getterů a setterů.

6 OpenSwing framework

6.1 Požadavky

Před samotnou instalací OpenSwing frameworku musíme splnit několik podmínek. Tou nejhlavnější je přítomnost Javy verze 1.4 a vyšší. Dále je to pak vývojové prostředí, které je podporováno grafickými komponentami OpenSwing frameworku. Pokud budeme chtít pracovat s grafickým návrhářem a mít ulehčeno vkládání komponent do formuláře, musíme si vybrat IDE, které umí s těmito komponentami pracovat. Je to například NetBeans od verze 4.1, JBuilder od verze 8, JDeveloper 10.1 a vyšší nebo Eclipse 3.0 a vyšší (s UI návrhářem Window Builder nebo Jigloo).

6.2 Instalace

Instalace OpenSwing frameworku je celkem snadná záležitost, ale vyžaduje pár kroků. Nejprve je potřeba stáhnout OpenSwing framework (CARNIEL, c2007). Poté musíme stáhnout podporované IDE, pokud ho ještě nemáme. Nainstalujeme IDE a pokud nám chybí, tak i Javu. Instalaci si předvedeme na NetBeans IDE 6.8. Nejprve je nutné rozbalit například do adresáře `C:\Temp\OpenSwing` stažený OpenSwing framework. Poté spustíme NetBeans a pustíme se do nastavení.

Vytvoření knihovny OpenSwing:

- Vyberte Tools | Libraries z menu,
- Vytvořte novou knihovnu po kliknutí na tlačítko New Library… a zadejte název OpenSwing,
- Poté vyberte nově vytvořenou knihovnu a zvolte Add JAR/Folder…,
- Vyberte všechny knihovny jar z adresáře `C:\Temp\OpenSwing\build`, kromě `BeanInfo.jar`.

Přidání komponent do UI návrháře:

- Vyberte Tools | Palette | Swing/AWT Components z menu,
- Klikněte na New Category… a zadejte název OpenSwing,
- Poté vyberte nově vytvořenou paletu a zvolte Add from JAR…,
- Vyberte z adresáře `C:\Temp\OpenSwing\build` archiv `BeanInfo.jar`,
- Na další obrazovce klikněte na komponentu v okně Available Components, stiskněte Ctrl + A a zvolte Next,
- Na další obrazovce vyberte paletu OpenSwing a potvrďte.

Nyní je již vše potřebné nastaveno a tak už jen po vytvoření projektu nesmíme zapomenout na přidání knihoven OpenSwing do projektu. To můžeme udělat označením projektu a přes pravé tlačítko myši volbou Properties | Libraries. Poté musíme zvolit Add Libraries a vybrat OpenSwing. Nebo můžeme rozvinout projekt a kliknout pravým tlačítkem myši na Libraries, zvolit Add Library… a vybrat OpenSwing.

7 Aplikace Dealer

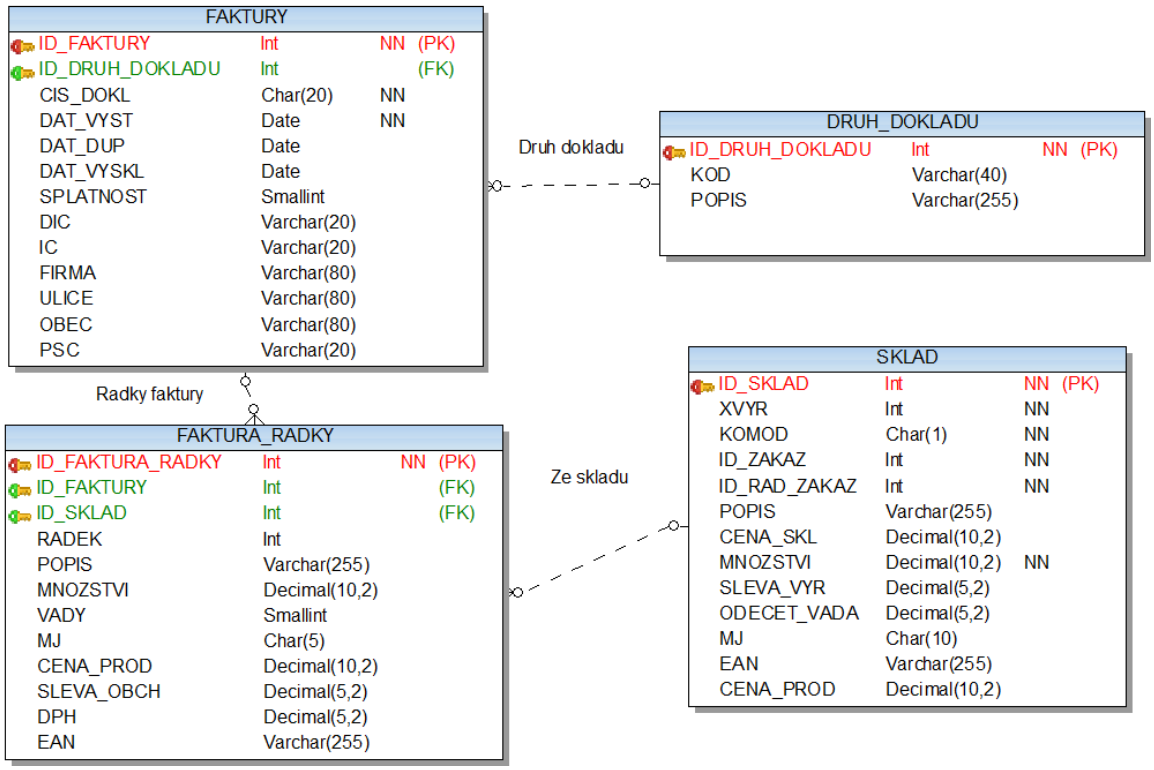
7.1 Analýza aplikace

Praktická část práce je zaměřena na tvorbu aplikace Dealer, která slouží ke správě a vydávání dokladů a faktur zákazníkovi. Aplikace dokáže spravovat druhy dokladů, které se dají vydávat. Dále umí načítat předem připravené zboží z centrálního skladu a vkládat ho do databáze, která představuje místní sklad dealera. Aplikace je vytvořena pro použití pouze jedním dealerem. Není ale nijak složité ji rozšířit o možnost přihlášení. Aplikace je postavena na platformě J2SE a využívá GUI rozhraní s komponentami z OpenSwing frameworku. Na propojení s databázemi využívá rozhraní JDBC. Jako místní (embedded) databáze byla zvolena databáze JavaDB (Apache Derby), která je součástí IDE NetBeans. Jako vzdálená, centrální databáze, byla zvolena Oracle Database.

7.2 ER diagram

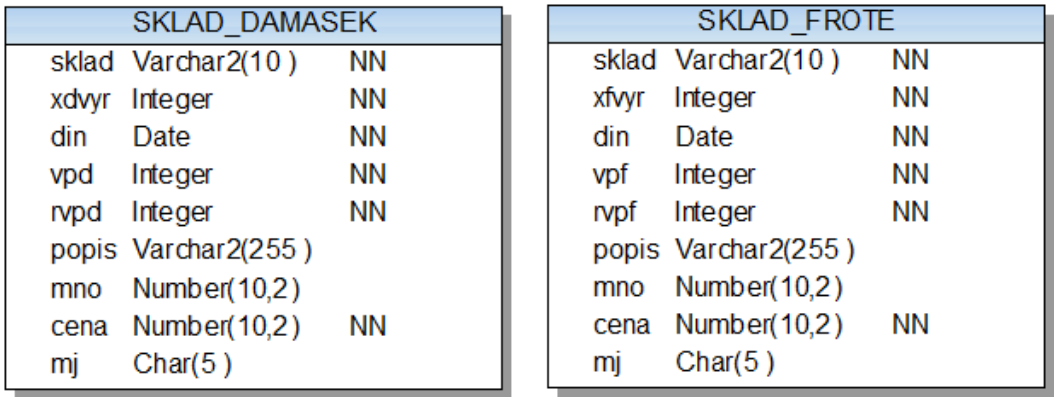
ER diagram je velice silný nástroj pro zobrazení struktury tabulek a jejich vazeb. Zobrazuje přehlednou formou sloupce jednotlivých tabulek a jejich vlastnosti. U zobrazení primárních a cizích klíčů se většinou používá jejich zvýraznění a odlišení od zbytku sloupců.

7.2.1 JavaDB



Obrázek 3 – Schéma tabulek databáze JavaDB. Zdroj vlastní.

7.2.2 Oracle



Obrázek 4 – Schéma tabulek databáze Oracle. Zdroj vlastní.

7.3 UML Class diagram

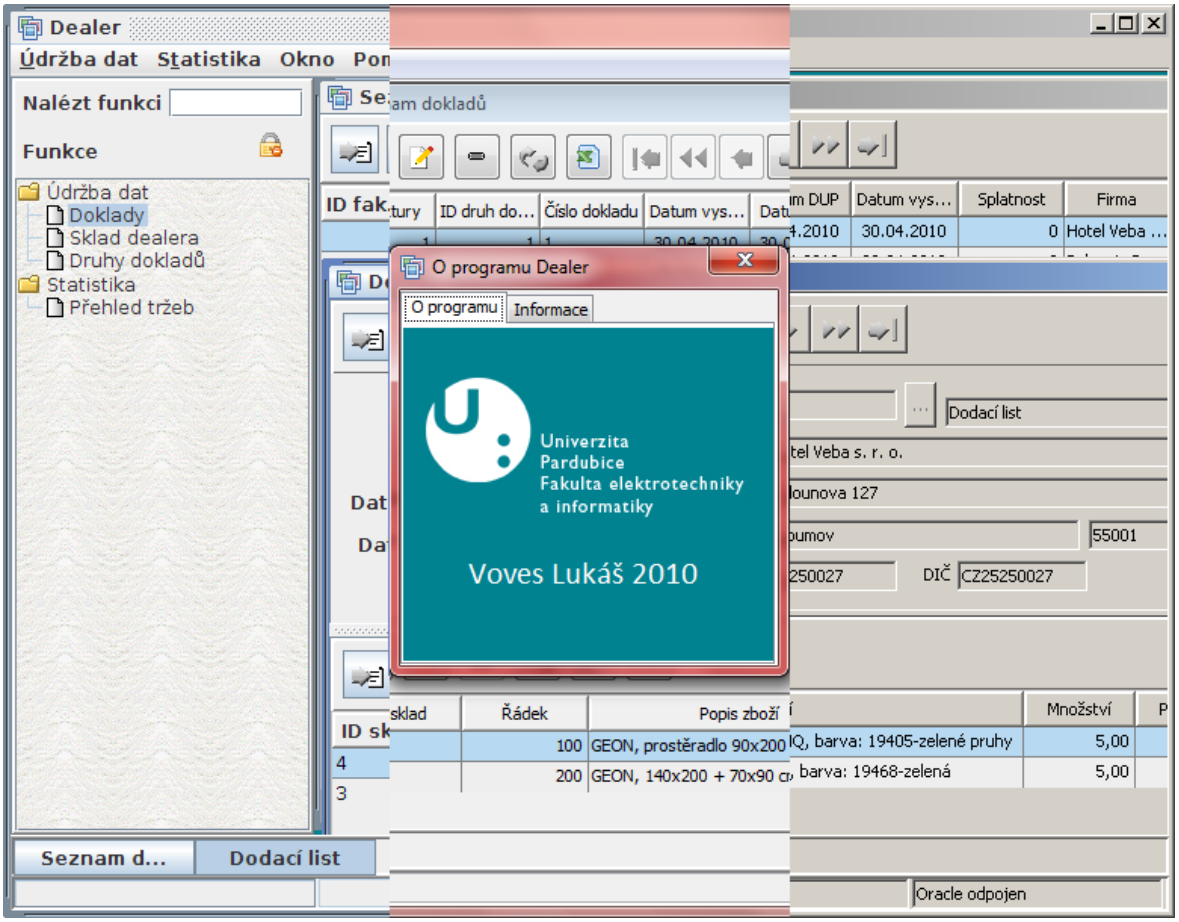
UML Class diagram s přehledem tříd je k dispozici v Příloze A. UML Class diagram nabízí vizuální přehled jednotlivých tříd a vztahů mezi nimi.

7.4 Implementace aplikace

Aplikace Dealer byla implementována jako desktopová MDI GUI aplikace. Jako vývojové prostředí bylo použito NetBeans ve verzi 6.8. Grafické uživatelské rozhraní, které využívá komponent Swing a OpenSwing, bylo vytvořeno v integrovaném grafickém návrháři. Pro vývoj byla použita knihovna OpenSwing ve verzi 2.1.7. Aplikace byla vyvíjena s pomocí J2SE JDK ve verzi 1.6.0_16. Pro připojení k databázi JavaDB byl použit Derby JDBC Embedded Driver ve verzi 10.4.1.3 a pro připojení k databázi Oracle byl použit Oracle JDBC Driver ve verzi 10.2.0.1.0. Výsledný program je v jediném jar souboru, ke kterému je ještě vytvořena složka s databází dealer a složka s knihovnami. Pro spojení s databázemi využívá aplikace navíc knihovnu `dealersettings.jar`, ve které jsou uloženy adresy databází a přihlašovací údaje k nim jako konstanty. Aplikace bude spustitelná všude tam, kde bude k dispozici JVM ve verzi 1.4 a vyšší.

7.5 Funkce a možnosti aplikace

Aplikace Dealer byla vyvíjena jak pro potřeby Bakalářské práce, tak i jako firemní aplikace pro ukázku možností frameworku OpenSwing při vývoji ve firemním prostředí. Aplikace Dealer je určena pro dealery zboží, kteří ji používají jako přenosnou aplikaci s hlavním propojením na místní databázi. Jelikož dealeři pracují hlavně v terénu, nebylo by vhodné volit pro ukládání dat nějakou vzdálenou databázi. Aplikace umožňuje přehledně spravovat zakázky, zobrazovat průběžné statistiky, vydávat doklady nebo exportovat do různých formátů. Díky tomu, že je aplikace naprogramovaná v jazyce Java, může být spuštěna v jakémkoliv operačním systému (Obrázek 5), pro který existuje běhové prostředí Javy.



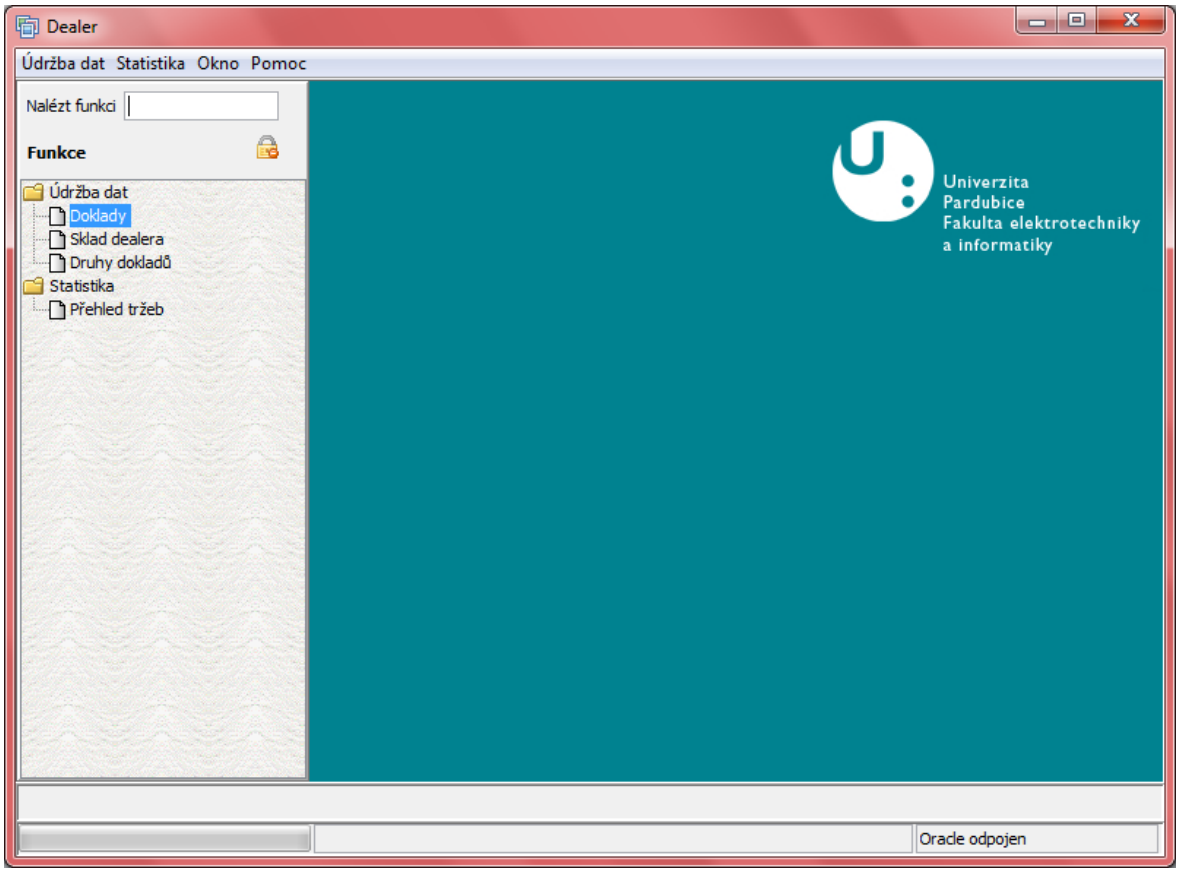
Obrázek 5 – Porovnání vzhledu aplikace v Linuxu, Windows 7 a XP. Zdroj vlastní.

7.5.1 Základní uživatelské rozhraní

Základní uživatelské rozhraní (Obrázek 6) je v aplikaci Dealer řešeno jako MDIFrame. Toto řešení je velmi užitečné pro aplikace, které používají vnitřní formuláře. Řešení jako SDI by mohlo být realizováno také, například při použití záložek. Aplikace Dealer ale pracuje s více okny, vhodnější je proto použít MDIFrame, který má v OpenSwing frameworku velké množství užitečných nastavení a možností. Jako jediný formulář se netvoří vizuálně v návrháři, ale pomocí definice vlastností a implementací MDIControlleru.

Aplikace Dealer je vyvíjena už od začátku jako vícevláknová aplikace s důrazem na odezvu GUI. Všechny formuláře jsou tedy samostatná vlákna a tím je zajištěno, že základní GUI může být aktivně k dispozici i v případě práce s velkým množstvím dat v mřížce. Komponenty OpenSwing frameworku jsou také spouštěny jako nová vlákna. Od verze 2.0.7 obsahuje OpenSwing framework na mé doporučení i oddělený export z mřížky nebo formuláře. Při exportu velkého množství dat z mřížky, například do PDF, zůstane GUI bez oddělení exportu do nového vlákna neaktivní až do doby, než export skončí. To je ale nežádoucí jev a ten je již eliminován.

Na základní stránce jsou aktivované 2 možnosti spuštění vnitřních oken. První způsob je přes vygenerovaný strom na levé straně. V něm může uživatel při větším množství formulářů snadno vyhledávat pomocí pole pro nalezení funkce. V našem případě je část se stromem funkcí ve výchozím nastavení zakotvena a je možné ji schovat. To je vhodné třeba v případě, že potřebujeme větší plochu programu pro vnitřní formuláře.



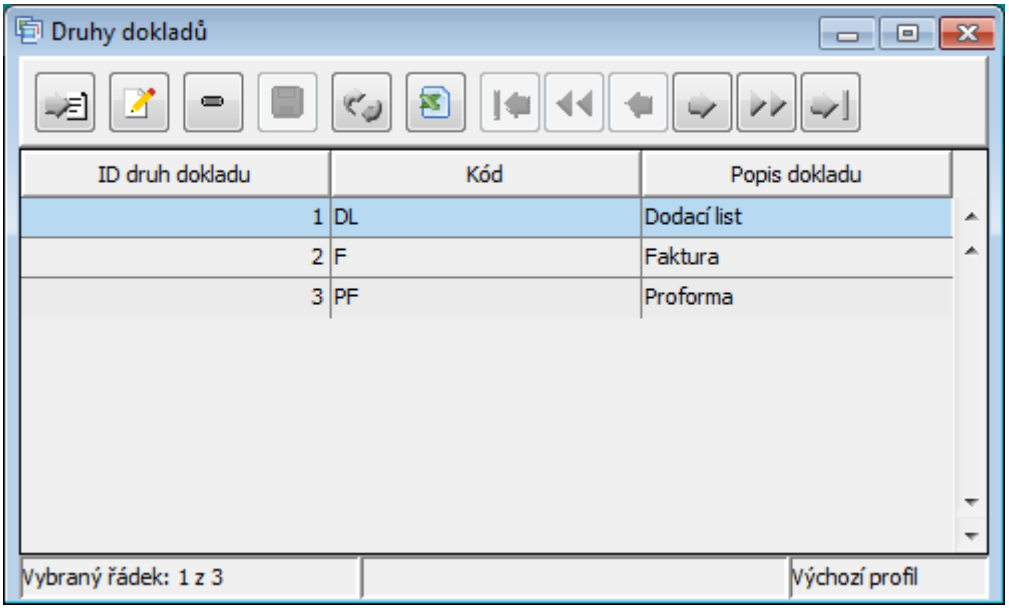
Obrázek 6 – Základní okno aplikace Dealer. Zdroj vlastní.

Další možností přístupu ke spuštění formulářů je menu. Všechny prvky menu se automaticky vygenerují a nemusíme je tak nijak přidávat. Pokud bychom ale chtěli přidat nějaké další volby do menu, je to také možné. Pokud máme otevřeno více vnitřních formulářů, pomocí volby Okno můžeme tato okna různě uspořádat a pracovat s nimi. Aplikace Dealer také poskytuje pro právě otevřená okna přehled na spodní liště. Zde se po spuštění okna zobrazí jeho ikona a můžeme tak mezi právě spuštěnými okny snadno přepínat. Aby bylo zamezeno zobrazení až moc velkého množství stejných oken, byla pro každý vnitřní formulář aktivována volba možnosti zobrazení pouze jedné instance okna. Aplikace je ve výchozím stavu v češtině, ale je možné ji po malé úpravě kódu v hlavní třídě předělat i do angličtiny nebo jiného dostupného jazyka, který je obsažen v balíčku, který má název `org.openswing.swing.internationalization.java.*`. Další možností je vytvořit vícejazyčnou aplikaci pomocí XML souborů, mezi kterými lze přepínat a vytvořit tak skutečně vícejazyčnou aplikaci. U této aplikace je také možná změna pozadí, pokud se nám přednastavené nelíbí. K přednastavenému se můžeme i po výměně bez problému vrátit.

Pokud by chtěl dealer přístup ke své aplikaci zabezpečit, stačí implementovat rozhraní `LoginController` a doplnit příslušné metody.

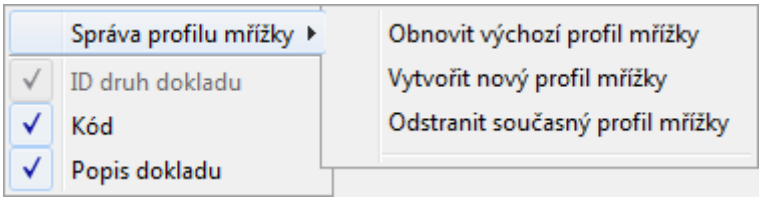
7.5.2 Formulář druhy dokladů

Součástí aplikace je formulář pro definování druhu dokladů (Obrázek 7). Pokud nám nestačí předdefinované druhy dokladů, můžeme si definovat své vlastní. Druhy dokladů můžeme vkládat, odebírat, exportovat nebo upravovat podle potřeby. V mřížce je možné se pohybovat nejen s pomocí myši nebo klávesnice, ale i pomocí navigačních tlačítek nad mřížkou.



Obrázek 7 – Formulář pro definici druhu dokladu. Zdroj vlastní.

Po klepnutí pravým tlačítkem myši na mřížku se nám zobrazí menu (Obrázek 8). V něm si můžeme označit nebo odznačit sloupce, které chceme zobrazit nebo schovat. V podmenu Správy profilu mřížky se potom skrývají možnosti pro obnovení, vytvoření nebo odstranění profilu mřížky. Po vytvoření se nám přidají další volby.

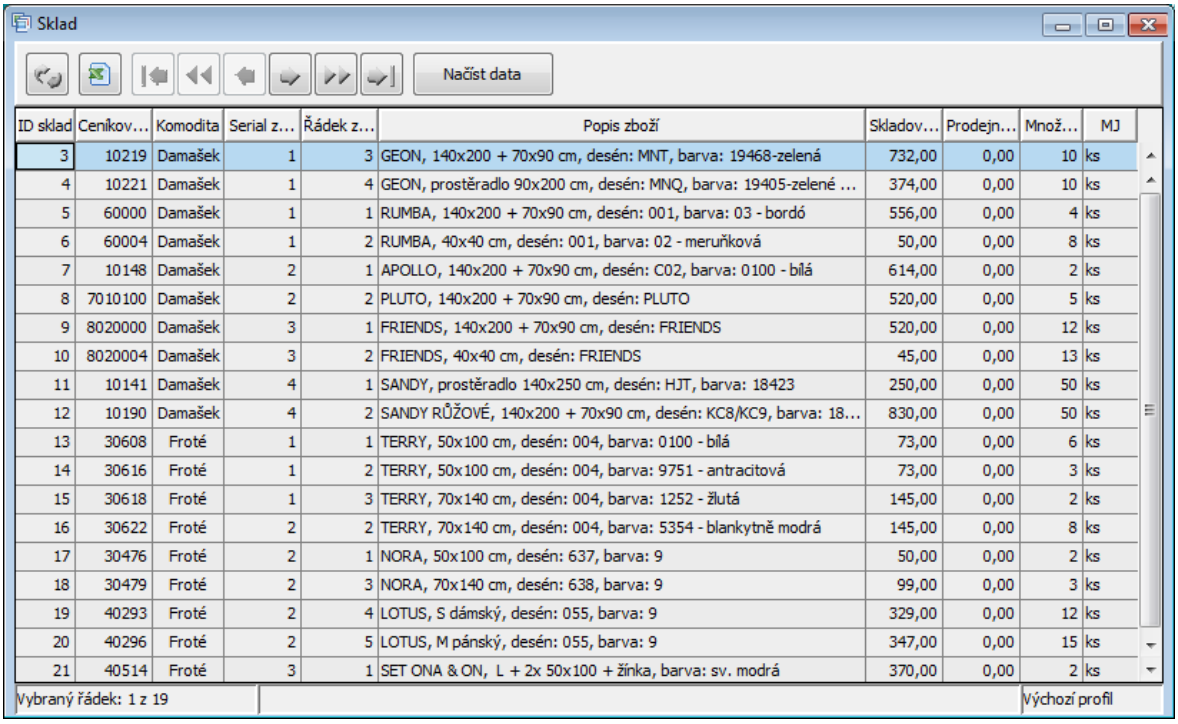


Obrázek 8 – Správa zobrazení mřížky. Zdroj vlastní.

7.5.3 Formulář sklad

Formulář sklad (Obrázek 9) obsahuje přehled zboží v místním skladu dealera. Formulář nelze z bezpečnostních důvodů nijak upravovat a slouží hlavně k přehledu zboží, které má dealer k dispozici na skladě. Pro zpracování zakázek skladníky a účetními je

k dispozici export dat z mřížky, takže si může dealer kdykoliv vyexportovat přehled například do Excelu.

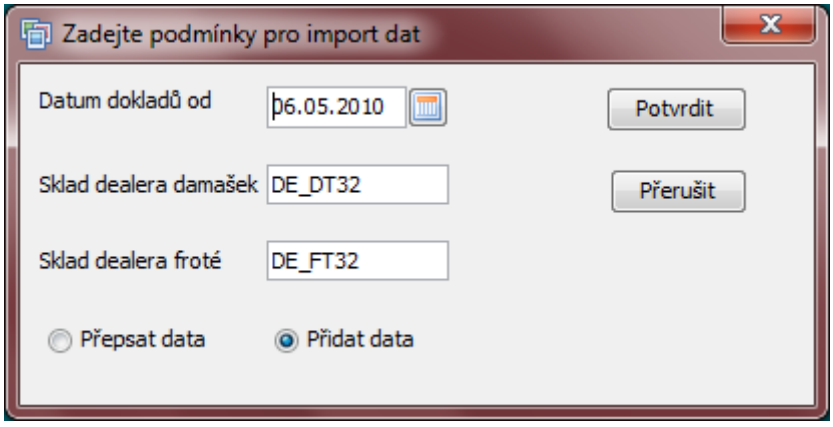


ID sklad	Ceníkov...	Komodita	Serial z...	Řádek z...	Popis zboží	Skladov...	Prodejn...	Množ...	MJ
3	10219	Damašek	1	3	GEON, 140x200 + 70x90 cm, desén: MNT, barva: 19468-zelená	732,00	0,00	10	ks
4	10221	Damašek	1	4	GEON, prostěradlo 90x200 cm, desén: MNQ, barva: 19405-zelené ...	374,00	0,00	10	ks
5	60000	Damašek	1	1	RUMBA, 140x200 + 70x90 cm, desén: 001, barva: 03 - bordó	556,00	0,00	4	ks
6	60004	Damašek	1	2	RUMBA, 40x40 cm, desén: 001, barva: 02 - meruňková	50,00	0,00	8	ks
7	10148	Damašek	2	1	APOLLO, 140x200 + 70x90 cm, desén: C02, barva: 0100 - bílá	614,00	0,00	2	ks
8	7010100	Damašek	2	2	PLUTO, 140x200 + 70x90 cm, desén: PLUTO	520,00	0,00	5	ks
9	8020000	Damašek	3	1	FRIENDS, 140x200 + 70x90 cm, desén: FRIENDS	520,00	0,00	12	ks
10	8020004	Damašek	3	2	FRIENDS, 40x40 cm, desén: FRIENDS	45,00	0,00	13	ks
11	10141	Damašek	4	1	SANDY, prostěradlo 140x250 cm, desén: HJT, barva: 18423	250,00	0,00	50	ks
12	10190	Damašek	4	2	SANDY RŮŽOVÉ, 140x200 + 70x90 cm, desén: KC8/KC9, barva: 18...	830,00	0,00	50	ks
13	30608	Froté	1	1	TERRY, 50x100 cm, desén: 004, barva: 0100 - bílá	73,00	0,00	6	ks
14	30616	Froté	1	2	TERRY, 50x100 cm, desén: 004, barva: 9751 - antracitová	73,00	0,00	3	ks
15	30618	Froté	1	3	TERRY, 70x140 cm, desén: 004, barva: 1252 - žlutá	145,00	0,00	2	ks
16	30622	Froté	2	2	TERRY, 70x140 cm, desén: 004, barva: 5354 - blankytně modrá	145,00	0,00	8	ks
17	30476	Froté	2	1	NORA, 50x100 cm, desén: 637, barva: 9	50,00	0,00	2	ks
18	30479	Froté	2	3	NORA, 70x140 cm, desén: 638, barva: 9	99,00	0,00	3	ks
19	40293	Froté	2	4	LOTUS, S dámský, desén: 055, barva: 9	329,00	0,00	12	ks
20	40296	Froté	2	5	LOTUS, M pánský, desén: 055, barva: 9	347,00	0,00	15	ks
21	40514	Froté	3	1	SET ONA & ON, L + 2x 50x100 + žínka, barva: sv. modrá	370,00	0,00	2	ks

Vybraný řádek: 1 z 19 | Výchozí profil

Obrázek 9 – Formulář pro správu zboží na místním skladu. Zdroj vlastní.

Hlavní funkcí formuláře skladu je import zboží (Obrázek 10) z centrálního skladu, který se vyvolá kliknutím na tlačítko Načíst data. Ten probíhá tak, že si nejprve dealer na centrálním skladě vybere výrobky, které mu byly přiděleny, nasnímá si jejich kódy, a podle nich si je potom přesune do databáze na centrálním skladě. Jsou mu přiděleny identifikátory skladu z databáze damašků a froté.



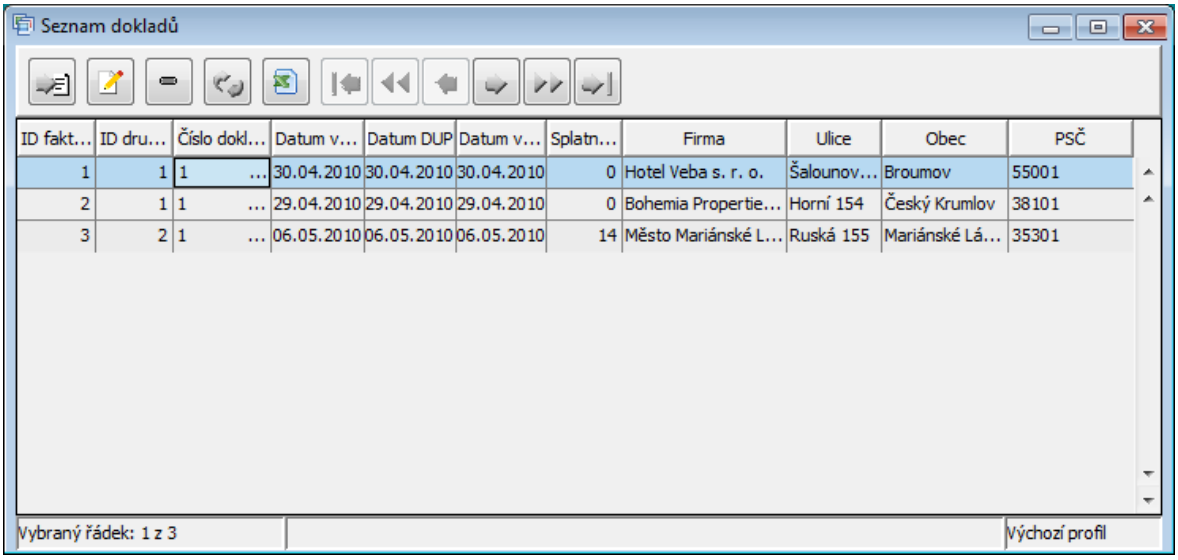
Obrázek 10 – Okno pro import z centrálního skladu. Zdroj vlastní.

Poté si v aplikaci Dealer zvolí datum, od kterého chce zboží z centrálního skladu načíst, identifikátor skladu damašků a identifikátor skladu froté. Musí si vybrat, jestli data do místního skladu pouze přidá, a tím se při vyhovujících podmínkách zvýší počty zboží, které už na skladě je, nebo zvolí možnost přepsání dat a u této volby se zboží z místního

skladu smaže a nahradí se zbožím novým. Po stisknutí tlačítka Přerušit se import přeruší beze změn. Po stisknutí tlačítka Potvrdit se u přepsání dat ještě aplikace zeptá, zda chceme data opravdu nahradit, a poté už proběhne samotný import[8]. V případě neúspěchu jsou zobrazena varovná okna s vysvětlením.

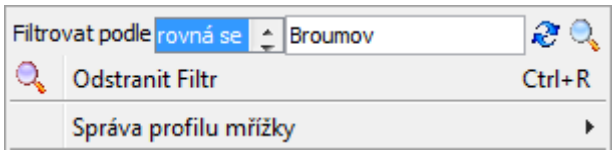
7.5.4 Formulář seznam dokladů

Formulář seznam dokladů (Obrázek 11) slouží jako hlavní formulář pro přehled a správu zakázek. Zakázky je možné vkládat, upravovat, mazat nebo exportovat. První 2 možnosti se otevírají do nového okna. Tento formulář slouží jako přechodný formulář, přehled o uskutečněných zakázkách, k formuláři s detailními údaji o zakázkách.



Obrázek 11 – Formulář seznam dokladů. Zdroj vlastní.

Jako u ostatních formulářů je i u tohoto formuláře možné upravit si zobrazení jednotlivých sloupců a jejich umístění. Sloupce lze libovolně přesouvat uchopením jejich záhlaví. Dále je také možné si vytvořit profily mřížky a přepínat mezi nimi. Podle některých sloupců, jako například ID faktury, lze data setřídit a podle některých sloupců, jako například Obec, lze data vytřídit (Obrázek 12). Setřídění lze provést kliknutím na záhlaví sloupce a vytřídění kliknutím pravého tlačítka myši, zvolením podle čeho a jak chceme třídit, a kliknutím na ikonku lupy. Pokud budeme chtít třídění zase vrátit zpátky, stačí kliknout na volbu Odstranit Filtr a data se zobrazí tak, jako před aplikací filtru.

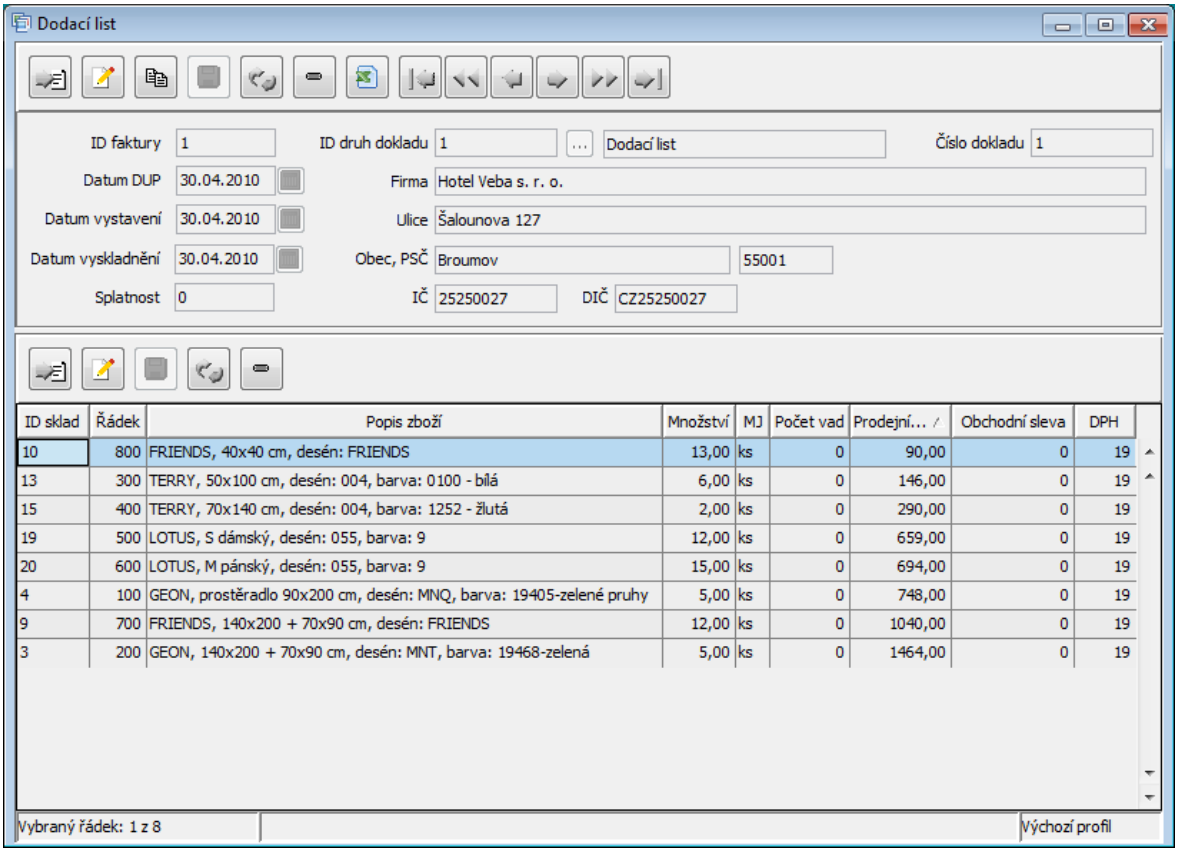


Obrázek 12 – Možnosti filtrování. Zdroj vlastní.

[8] V této práci je jako vzdálený sklad použita školní databáze Oracle a tudíž pro správné fungování je potřeba být buď připojen přes VPN nebo aplikaci spouštět v intranetové síti UPa.

7.5.5 Formulář dodací list

Formulář dodací list (Obrázek 13) je nejsložitější z představených formulářů. Je to formulář typu master-grid a je propojený s formulářem dodací listy, ze kterého jedině může být vyvolán, a to pro vložení nového dodacího listu, nebo pro upravení starého. Při procházení formuláře navigační lištou nahoře je listováno i formulářem dodací listy. To samé funguje i naopak, pokud jsou otevřeny oba formuláře. Při vytváření nové hlavičky dodacího listu jsou spodní tlačítka neaktivní a nemůže tak dojít k nechtěnému přiřazení zboží k jinému nebo neexistujícímu dokladu. Pokud je formulář v módu čtení, můžeme stiskem tlačítka pro vložení nebo editaci vložit nový doklad nebo upravit starý.

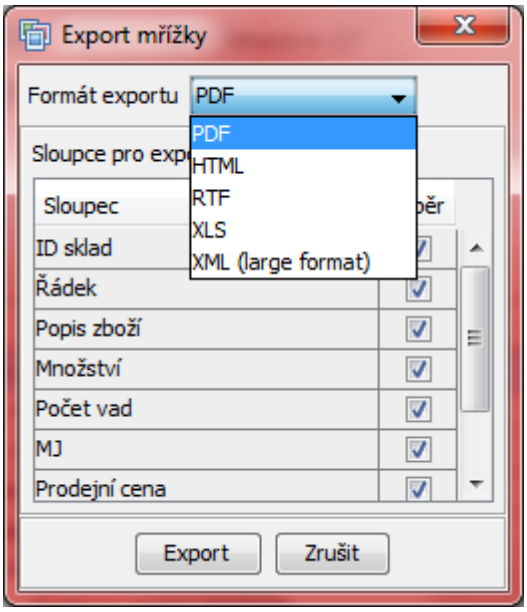


Obrázek 13 – Formulář dodací list. Zdroj vlastní.

Ve spodní části formuláře jsou zobrazeny informace o vybraném řádku a celkovém počtu řádků v levé části. V pravé části je pak zobrazen název právě použitého profilu mřížky. Při vkládání a editaci jednotlivých řádků se zbožím se kontroluje, zda jsme nevybrali více kusů, než je na skladě. V případě, že jsme zadali větší množství, aplikace sama doplní, jaké maximální množství můžeme vyplnit. Při zadání správného množství nás kontrola pustí dále. Velmi silnou zbraní frameworku OpenSwing jsou jeho možnosti exportu. Jediný vážnější problém je při exportu do formátu PDF, který bohužel špatně zobrazuje české znaky. Řešení tohoto problému je ale velmi jednoduché. Stačí vytvořit potomka třídy `ExportToPDFAdapter` a upravit metody pro volbu fontu a znakové sady. Ve třídě `Main` se poté přidá například řádek:

```
ClientSettings.EXPORT_TO_PDF_ADAPTER = new ExportDoPDF();
```

Samotný export může být do několika formátů (Obrázek 14). Při exportu si můžeme zvolit, které sloupce chceme exportovat a které vynechat. V mřížce pro export jsou pouze ty sloupce, které jsou viditelné v detailní mřížce. Pokud některý sloupec schováme, ale budeme ho chtít exportovat, tak je nutné si vytvořit více profilů mřížky nebo sloupec vždy dočasně zviditelnit. Ostatní formuláře mají také možnost exportu dat z mřížky, ale tento master-detail formulář je zvláštní v tom, že u něj probíhá export 2 kroky. Nejdříve se musejí vypsat hodnoty z master části, které se musí vytvořit ručně. Po vypsání hlavičky dokumentu se vygenerují automaticky i detailní informace ze spodní detail mřížky. Ukázkový příklad, jak může vypadat vyexportovaný doklad, lze najít v příloze B.



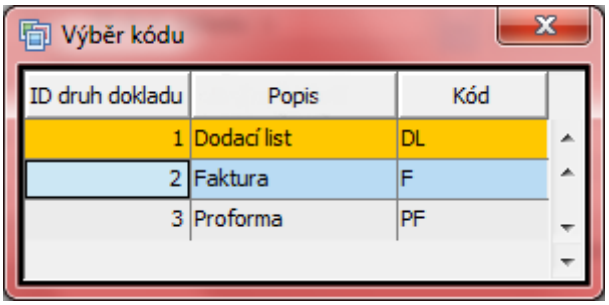
Obrázek 14 – Možnosti exportu mřížky. Zdroj vlastní.

Master formulář používá pro vkládání datumů komponentu `DateControl` (Obrázek 15), která využívá funkcí knihovny `jcalendar`.jar a pro výběr datumu třídu `JDateChooser`.

Obrázek 15 – Pole pro výběr datumu. Zdroj vlastní.

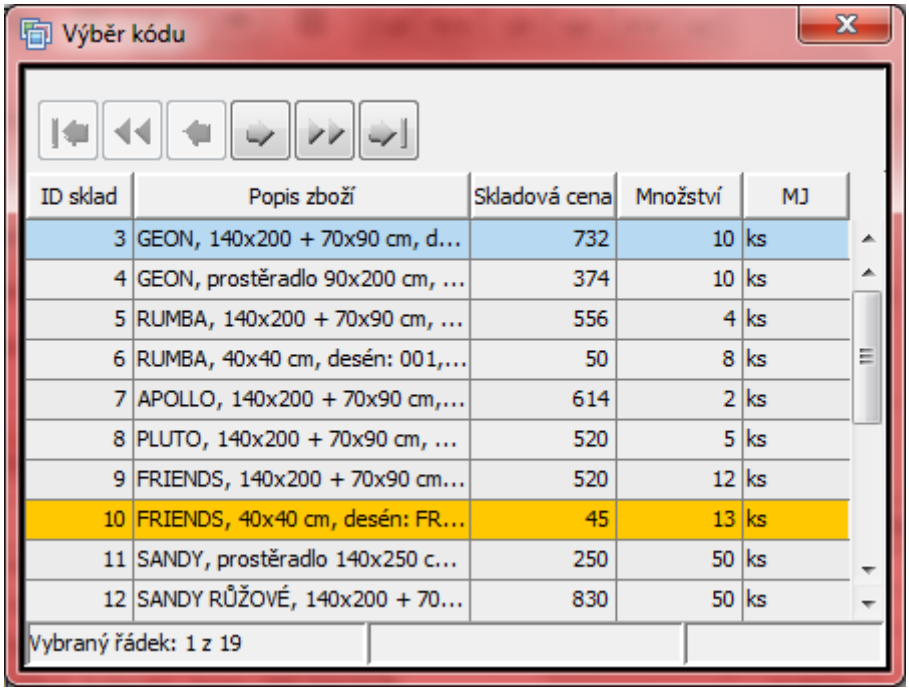
Dále je pak využit `CodLookupControl` (Obrázek 16) pro výběr dokladu. Pokud si uživatel nechce vybrat typ dokladu pomocí lookupu, tak má také možnost zapsat ID

dokladu do pole a pokud toto ID existuje a je mu přiřazen typ dokladu, je do vedlejšího pole vepsán jeho název. V případě chyby vyskočí varovné okno. Lookup je propojen s master formulářem a tak se po výběru typu dokladu zapíše ID a popis do patřičných políček v master formuláři.



Obrázek 16 – Lookup pro výběr dokladu. Zdroj vlastní.

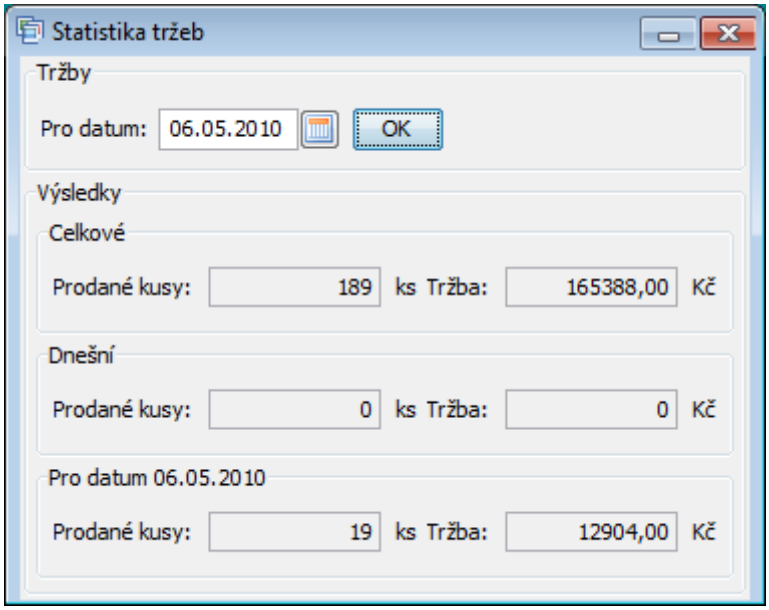
V detail mřížce je pak využit `CodLookupColumn` na zobrazení lookupu pro výběr zboží v mřížce (Obrázek 17). Po výběru se informace z lookupu doplní do detail mřížky. V lookupech je aktivně vybraný řádek zobrazen oranžovým podbarvením a námi nově zvolený řádek je podbarven modře. Lookup pro výběr zboží je navíc doplněn o navigační lištu.



Obrázek 17 – Lookup pro vložení zboží. Zdroj vlastní.

7.5.6 Formulář statistika tržeb

Formulář statistika tržeb (Obrázek 18) se od těch ostatních vnitřních formulářů liší jednou hlavní vlastností. Není možné ho maximalizovat ani nijak roztáhnout. Je to proto, aby byla zajištěna co nejlepší přehlednost statistik. Formulář zobrazuje přehledně statistiky celkové, dnešní a také pro den, který zadáme v horní části a potvrdíme tlačítkem OK.

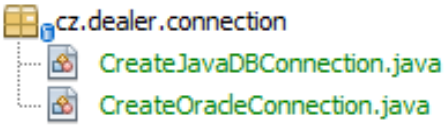


Obrázek 18 – Formulář statistika tržeb. Zdroj vlastní.

7.6 Struktura aplikace – popis jednotlivých balíčků a částí aplikace

Aplikace Dealer je kvůli přehlednosti rozdělena do několika balíčků. V každém balíčku jsou třídy, které patří k sobě. Většina balíčků dodržuje uspořádání tak, že obsahuje soubory `<Nazev>.java`, `<Nazev>Controler.java` a také `<Nazev>VO.java`. Framework OpenSwing je postaven na MVC a většina balíčků tento styl dodržuje. První soubor obsahuje formulář, který zobrazuje data. Druhý soubor obsahuje ovladač, slouží pro práci s modelem a poskytuje metody formuláři. Třetí soubor je potom samotný model, se kterým ovladač pracuje a formulář zobrazuje.

7.6.1 Balíček cz.dealer.connection



Obrázek 19 – Balíček cz.dealer.connection. Zdroj vlastní.

Tento balíček obsahuje 2 třídy, které obsluhují připojení k databázím pomocí JDBC Driverů. `CreateJavaDBConnection` má navíc i kontrolu existence databáze, takže v případě neexistující databáze se vytvoří a poté se zavolá metoda `createTables()`, která vytvoří tabulky potřebné k běhu aplikace a pohled potřebný pro zobrazení tržeb. Do tabulky `DRUH_DOKLADU` se vloží výchozí 2 řádky.

Pro ukázku, konstruktor třídy `CreateJavaDBConnection` vypadá takto:

```
public CreateJavaDBConnection(String connect, String user, String
password) {
  Properties p = new Properties();
  try {
      Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
```

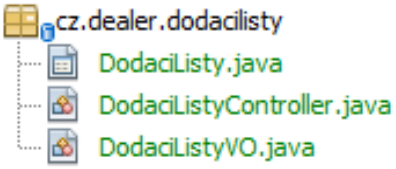
```
      p.setProperty("user", user);
      p.setProperty("password", password);

      conn = DriverManager.getConnection(connect, p);
  } catch (Exception ex) {
    if (conn == null) {
      try {
        p.setProperty("create", "true");
        conn = DriverManager.getConnection(connect, p);
        createTables();
      } catch (SQLException ex1) {
      }
    }
  }
}
```

7.6.2 Balíček cz.dealer.dodacilist

Obrázek 20 – Balíček cz.dealer.dodacilist. Zdroj vlastní.

V tomto balíčku se nacházejí třídy pro obsluhu formuláře `DodaciList`. Jelikož je formulář typu master-detail, tak má také k dispozici 2 modely VO, které obsahují strukturu, se kterou se pracuje, a 2 ovladače. `DodaciListDetailController` obsahuje metodu `validateCell()`, která má na svědomí kontrolu množství přidávaného zboží. Dále obsahuje metodu `getExportingFormats()` pro upřesnění formátů, do kterých je možné exportovat a metodu `exportGrid()`, která se stará o vypisování hlavičky při exportu. Ta se přidá před samotný výpis mřížky. Dále jsou zde metody `getID()` a `getCisloRadku()`, které mají na starosti správné generování použitelných čísel pro nové záznamy. `DodaciListMasterController` obsahuje také metodu `getID()`.

7.6.3 Balíček cz.dealer.dodacilisty

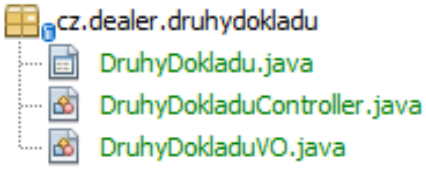


Obrázek 21 – Balíček cz.dealer. dodacilisty

Třída `DodaciListy` z tohoto balíčku neobsahuje napojení tlačítek pro vložení a editaci na mřížku. Místo toho používá funkce pro zavolání nového okna buď s vyplněným parametrem primárního klíče pro editaci, nebo s parametrem null místo klíče pro vložení

nové zakázky. Třída `DodaciListyController` obsahuje metodu `doubleClick()`, která zobrazí formulář `DodaciList` v režimu čtení.

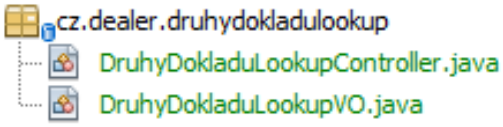
7.6.4 Balíček cz.dealer.druhydokladu



Obrázek 22 – Balíček cz.dealer.druhydokladu

Balíček obsahuje model VO a ovladač pro formulář `DruhyDokladu`. Ovladač obsahuje metodu `getID()` pro vygenerování primárního klíče.

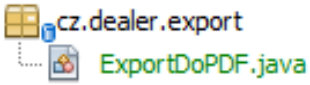
7.6.5 Balíček cz.dealer.druhydokladulookup



Obrázek 23 – Balíček cz.dealer.druhydokladulookup

Tento balíček nemá žádný formulář, protože lookup se generuje pomocí parametrů v ovladači. Třída `DruhyDokladuLookupController` má metodu `getBackgroundColor()`, která se stará o správné vybarvení řádků. Pokud řádek odpovídá hodnotě uvedené v políčku, odkud se lookup zavolal, vybarví se oranžově. Pokud je vybrán myší, obarví se modře.

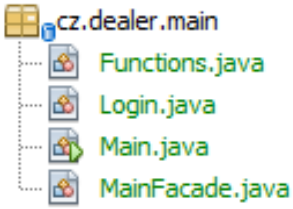
7.6.6 Balíček cz.dealer.export



Obrázek 24 – Balíček cz.dealer.export

Třída `ExportDoPDF` je potomkem třídy `ExportToPDFAdapter`, která se používá k nastavení písma, ve kterém se generuje PDF dokument. Protože sama rodičovská třída si neporadí s českými znaky, `ExportDoPDF` nastavuje písmo a znakovou sadu tak, aby bylo PDF vygenerováno se správnými českými znaky.

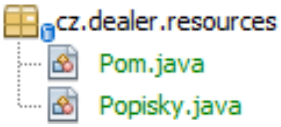
7.6.7 Balíček cz.dealer.main



Obrázek 25 – Balíček cz.dealer.main

Balíček obsahuje spouštěcí třídu `Main`, která implementuje rozhraní `MDIController` a příslušné metody pro konstrukci `MDIFrame`. V této třídě se navazuje spojení s databází JavaDB, pomocí `ClientSettings` se nastavuje rozhraní na české, statické proměnné se nastavují podle požadavků na funkce aplikace a v implementovaných metodách se nastavují potřebné údaje či informace. Jednotlivé odkazy na formuláře jsou definovány ve třídě `Functions`. Třída `MainFacade` obsahuje metody pro spuštění formulářů, na které se odkazuje ve třídě `Functions`. Třída `Login` je zde jen pro ukázku, jak je možné vytvořit přihlašovací dialog buď do aplikace, nebo v tomto případě do databáze.

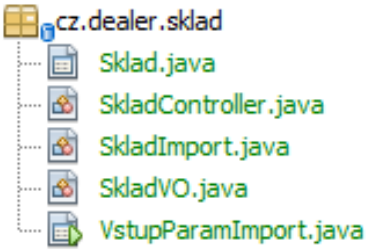
7.6.8 Balíček cz.dealer.resources



Obrázek 26 – Balíček cz.dealer.resources

Ve třídě `Pom` se skrývá sada statických metod, které jsou používány napříč balíčky a zjednodušují práci. Metody jsou celkem jednoduché, ale protože jsou použity vícekrát, bylo výhodné je vložit do samostatné třídy. Třída `Popisky` je potomkem třídy `Properties` a uchovává překlady k jednotlivým klíčovým slovům, která byla použita v celém projektu. Překladem této třídy můžeme získat texty k jiným jazykům.

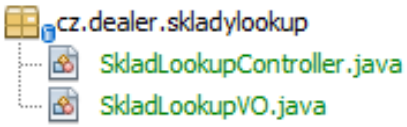
7.6.9 Balíček cz.dealer.sklad



Obrázek 27 – Balíček cz.dealer.sklad

Tento balíček v sobě ukrývá třídy pro obsluhu formuláře `Sklad` a třídy pro import dat ze vzdáleného skladu. Formulář `Sklad` volá po stisknutí tlačítka Načíst data nejprve dialog `VstupParamImport`, ze kterého zjistí potřebné údaje a poté volá třídu `SkladImport`, která se stará o obsluhu importu dat. Třída `SkladImport` obsahuje mnoho privátních metod pro práci s daty, settery a gettery pro nastavování a zjišťování parametrů importu. Obsahuje také jednu hlavní veřejnou metodu `importuj()`, která v případě úspěšného spojení a načtení vrací hodnotu true. V případě potíží třída oznamuje, co přesně je v nepořádku.

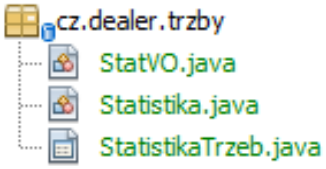
7.6.10 Balíček cz.dealer.skladylookup



Obrázek 28 – Balíček cz.dealer.skladylookup

Třídy v tomto balíčku mají podobnou funkci jako třídy `DruhyDokladuLookup`, ale formulář ovladače je nastaven na větší velikost, aby byl výběr přehlednější. Obsahuje také navigační lištu, díky které je procházení většího množství zboží jednodušší.

7.6.11 Balíček cz.dealer.trzby



Obrázek 29 – Balíček cz.dealer.trzby

Tento balíček obsahuje třídy pro zobrazení statistik tržeb. Třída `Statistika` obsahuje metody pro výpočet celkových tržeb, tržeb uskutečněných dnes a tržeb pro určité datum.

8 Závěr

Vývoj firemních desktopových aplikací se už v dnešní době velice často nahrazuje vývojem webových aplikací. Desktopové aplikace jsou ale i přesto velice důležité. Ve starších firmách není často možné z časových důvodů nahrazení dosavadních aplikací, které jsou určené pro desktop, aplikacemi webovými. Častým důvodem nasazení desktopových aplikací namísto webových může být i požadavek uživatelů na zachování funkcionality vyvíjené aplikace s aplikacemi již zaběhnutými. Koncoví uživatelé se velice neradi odnaučují již známé postupy nebo funkce, které mají zažité.

Cílem této práce bylo zjistit, zda je framework OpenSwing vhodný pro vývoj firemních desktopových aplikací. Zhodnocení této otázky je velice jednoduché. OpenSwing framework je pro tyto účely velice vhodný, protože poskytuje většinu komponent potřebných pro programátory již v základu a nemusí se tak doprogramovávat. Naučit se pracovat s tímto frameworkem chvíli trvá, ale po pochopení základní práce s mřížkami, formuláři nebo kombinovanými formuláři, je již programátor schopen v celkem krátkém čase vytvořit plně funkční aplikaci.

Při ucelení stylů přístupu k formuláři a databázi je možné si vytvořit návrhový vzor, který se pouze doplní správnými hodnotami a vývoj se tím značně urychlí. Pro vytvoření jednoduchého formuláře nám stačí 3 třídy. Datový model zvaný `ValueObject`, ovladač `Controller` a samotný formulář. Datový model se využívá v ovladači a je napojen také na formulář, kde pak jednoduše přiřadíme jednotlivé položky modelu například sloupcům v mřížce a to vše v grafickém návrháři. K tomuto směřuje jedna výtka. V prostředí IDE NetBeans občas při změně vlastností komponent dochází k zrušení vazeb mezi datovým modelem a grafickými komponentami. Vazba se dá poté zase navázat, ale při větších úpravách je to již trochu obtěžující chování.

Knihovnu OpenSwing mohu tedy vřele doporučit všem těm, kteří často vyvíjejí desktopové aplikace, které pracují s tabulkami a formuláři. Během práce jsem objevil spoustu dalších užitečných vlastností, které jsem předtím neznal. Velice pomohou metody v ovladači, které umožní kontrolovat a upravovat vlastnosti modelu prakticky v kterékoliv chvíli. Metody typu `createValueObject()`, `validateCell()`, `afterInsertGrid()` a další mohou programátorovi pomoci například s předvyplněním některých částí nově vytvářeného objektu.

Aplikace je plně funkční a může být i nasazena v běžném provoze. Byla však vytvářena hlavně za účelem prezentace některých možností frameworku OpenSwing a tento účel splnila. Pokud se najde někdo, koho by framework zaujal více, nejvíce se naučí ze zdrojových kódů Demo příkladů dodávaných s OpenSwing frameworkem nebo prozkoumáním zdrojových kódů aplikace Dealer.

Literatura

CARNIEL, Mauro. c2007. OpenSwing Framework. *SourceForge.* [Online] CARNIEL, Mauro, c2007. [Citace: 8. 5. 2010] Dostupné z WWW: <http://oswing.sourceforge.net/index.html>.

—. 2007. *OpenSwing: Release 1.9.4.* [PDF] místo neznámé: CARNIEL, Mauro, 2007. Dostupné po zaplacení poplatku z WWW: <http://oswing.sourceforge.net/moredoc.html>.

KADLEC, Václav. 2001. *Učíme se programovat v Delphi a jazyce Object Pascal.* Praha: Computer Press, 2001. ISBN 80-7226-245-9.

KEOGH, James. 2005. *Java bez předchozích znalostí: Průvodce pro samouky.* Brno: CP Books, 2005. ISBN 80-251-0839-2.

KISZKA, Bogdan. 2003. *1001 tipů a triků pro programování v jazyce Java.* Brno: Computer Press, 2003. ISBN 80-7226-989-5.

PRATA, Stephen. 2001. *Mistrovství v C++.* Praha: Computer Press, 2001. ISBN 80-7226-339-0.

PUŠ, Petr. 2004. Poznáváme C# a Microsoft .NET – 1.díl. *Živě.cz.* [Online] CPress Media, 23. 11. 2004. [Citace: 7. 5. 2010] Dostupné z WWW: <http://www.zive.cz/Clanky/Poznavame-C-a-Microsoft-NET--1dil/sc-3-a-120978/default.aspx>. ISSN 1212-8554.

WIKIPEDIA. 2010e. .NET In *Wikipedia: the free encyclopedia.* [Online] St. Petersburg (Florida): Wikipedia Foundation, 21. 4. 2010e. [Citace: 7. 5. 2010] Dostupné z WWW: <http://cs.wikipedia.org/wiki/.NET>.

—. 2010d. Apache Derby In *Wikipedia: the free encyclopedia.* [Online] St. Petersburg (Florida): Wikipedia Foundation, 28. 4. 2010d. [Citace: 7. 5. 2010] Dostupné z WWW: <http://en.wikipedia.org/wiki/Apache_Derby>.

—. 2010c. C Sharp In *Wikipedia: the free encyclopedia.* [Online] St. Petersburg (Florida): Wikipedia Foundation, 28. 4. 2010c. [Citace: 4. 5. 2010] Dostupné z WWW: <http://cs.wikipedia.org/wiki/C_Sharp>.

—. 2010b. Embarcadero Delphi In *Wikipedia: the free encyclopedia.* [Online] St. Petersburg (Florida): Wikipedia Foundation, 26. 4. 2010b. [Citace: 3. 5. 2010] Dostupné z WWW: <http://en.wikipedia.org/wiki/Embarcadero_Delphi>.

—. 2009a. IBM Informix Dynamic Server In *Wikipedia: the free encyclopedia.* [Online] St. Petersburg (Florida): Wikipedia Foundation, 15. 8. 2009a. [Citace: 8. 5. 2010] Dostupné z WWW: <http://en.wikipedia.org/wiki/IBM_Informix_Dynamic_Server>.

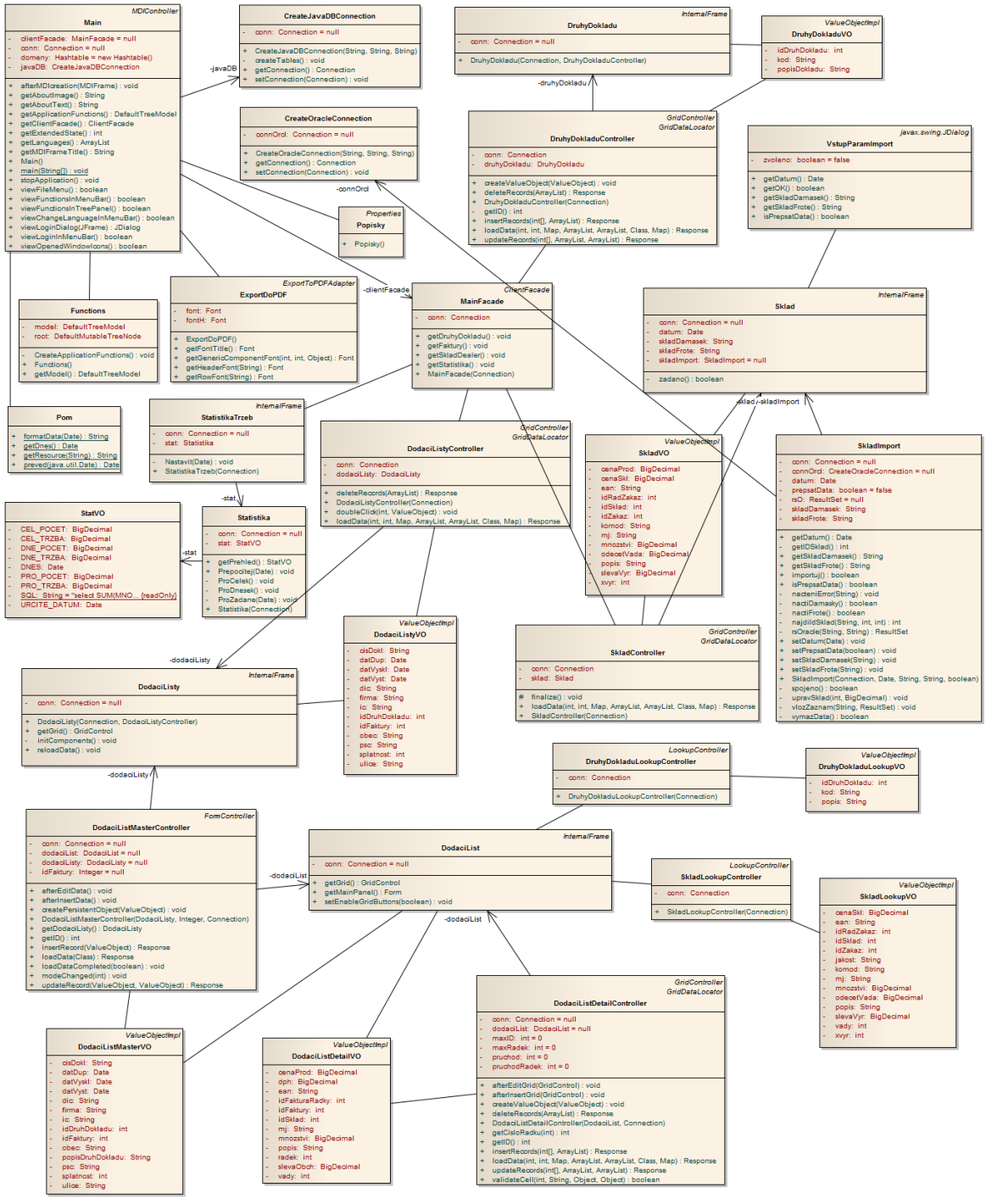
—. 2010a. Object Pascal In *Wikipedia: the free encyclopedia.* [Online] St. Petersburg (Florida): Wikipedia Foundation, 15. 4. 2010a. [Citace: 2. 5. 2010] Dostupné z WWW: <http://en.wikipedia.org/wiki/Object_Pascal>.

—. 2010f. Oracle Database In *Wikipedia: the free encyclopedia.* [Online] St. Petersburg (Florida): Wikipedia Foundation, 8. 5. 2010f. [Citace: 9. 5. 2010] Dostupné z WWW: <http://en.wikipedia.org/wiki/Oracle_Database>.

—. 2009b. SWT In *Wikipedia: the free encyclopedia.* [Online] St. Petersburg (Florida): Wikipedia Foundation, 25. 6. 2009b. [Citace: 9. 5. 2010] Dostupné z WWW: <http://cs.wikipedia.org/wiki/SWT>.

ZAKHOUR, Sharon, a další. 2007. *Java 6: Výukový kurz.* Brno: Computer Press, 2007. ISBN 978-80-251-1575-6.

Příloha A – UML Class diagram aplikace Dealer



Příloha B – Ukázka exportu dokladu

ID faktury	1	Druh dokladu	Dodací list
Datum DUP	30.4.2010	Datum vystavení	30.4.2010
Datum vyskladnění	30.4.2010	Splatnost	0
Firma	Hotel Veba s. r. o.	Ulice	Šalounova 127
Obec, PSČ	Broumov	55001	
IČ	25250027	DIČ	CZ25250027

ID sklad	Řádek	Popis zboží	Množství	Počet vad	MJ	Prodejní cena	Obchodní sleva	DPH
10	800	FRIENDS, 40x40 cm, desén: FRIENDS	13.00	0	ks	90.00	0.00	19.00
13	300	TERRY, 50x100 cm, desén: 004, barva: 0100 - bílá	6.00	0	ks	146.00	0.00	19.00
15	400	TERRY, 70x140 cm, desén: 004, barva: 1252 - žlutá	2.00	0	ks	290.00	0.00	19.00
19	500	LOTUS, S dámský, desén: 055, barva: 9	12.00	0	ks	659.00	0.00	19.00
20	600	LOTUS, M pánský, desén: 055, barva: 9	15.00	0	ks	694.00	0.00	19.00
4	100	GEON, prostěradlo 90x200 cm, desén: MNQ, barva: 19405-zelené pruhy	5.00	0	ks	748.00	0.00	19.00
9	700	FRIENDS, 140x200 + 70x90 cm, desén: FRIENDS	12.00	0	ks	1040.00	0.00	19.00
3	200	GEON, 140x200 + 70x90 cm, desén: MNT, barva: 19468-zelená	5.00	0	ks	1464.00	0.00	19.00