UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Název bakalářské práce

Jan Plecháč

Bakalářská práce
2010

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

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jan PLECHÁČ**
Osobní číslo:	**I07754**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Software pro návrh rozpočtu elektrikářských prací**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

V teoretické části práce bude vytvořen přehled typů databází, které budou následně jednotlivě popsány. Dále bude potřebné popsat dnešní nástroje pro tvorbu softwaru a jejich vazby na předcházející výčet databází.
Praktická část bude primárně zaměřena na návrh a implementaci aplikace pro tvorbu rozpočtu elektrikářských prací s využitím relační databáze.
Testování aplikace se provede nad daty středně velké firmy.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

1. **PÍCKA, M. Objektová databáze a její nasazení v praxi. Praha, ČZU, Dostupný z http://dsp2002.pef.czu.cz/pdf/dsp-165.pdf.**
2. **JEŽEK, K. Deduktivní databáze a jejich implementace v relačním prostředí. Plzeň, ZCU-FVA, Dostupný z http://www.kiv.zcu.cz/jezek_ka/vyuka/DB2%202005/DEDUC/datakon02.pdf**
3. **ŽÁK, K. Historie relačních databází. 2001 [19.10.2001]. Dostupný z http://www.root.cz/clanky/historie-relacnich-databazi/**

Vedoucí bakalářské práce: **Ing. Jan Fikejz**
Katedra softwarových 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 14. 5. 2009 Jan Plecháč

Poděkování

Zde bych chtěl poděkovat především svému vedoucímu Ing. Janu Fikejzovi za jeho cenné rady a připomínky při tvorbě bakalářské práce, že si na mne udělal čas, i když to pro něj nebylo snadné.

Anotace

Tato práce se zabývá vytvořením softwaru pro návrh rozpočtů elektrikářských prací a dále seznamuje čtenáře s tímto softwarem.

Klíčová slova

Programovací jazyky, java, C#, software, visual studio, csv

Title

Software for electrical works draft budget

Annotation

This work deals with creating software for the design of electrical works budgets and to acquaint the reader with this software.

Keywords

Programming Languages, java, C#, software, visual studio, csv

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Cílem mé práce je vytvořit software pro návrh rozpočtu elektrikářských prací, který by byl pro uživatele snadno ovladatelný a co nejvíce mu usnadnil práci.

V teoretické části bych vás chtěl seznámit s nástroji pro tvorbu softwarů. Tím je myšleno vývoj programátorských jazyků od počátku po současnost. Je zde také zmíněn vývoj způsobu programování od počátku strojového programování přes strukturované až po objektově orientované programování.

V další části, uživatelské, je popsán program a vysvětleno k čemu slouží. Dále je zde podrobný popis jeho funkcí a možností. Podle této části by měl každý uživatel dokázat tento program obsluhovat.

Závěrečná část se zabývá konkrétní implementací programu. Nejprve je zde zmíněno pomocí jakého programovacího jazyka a vývojového prostředí je aplikace zhotovena. Dále je zde nastíněn postup práce a na konec jsou zde popsány jednotlivé tříd a formuláře.

1 Nástroje pro tvorbu softwarů

Zde bych vás chtěl seznámit s některými programátorskými jazyky pro tvorbu softwaru.

1.1 Historie

Programovací jazyk je prostředek pro zápis programu, který je prováděn v počítači. Je to jakýsi komunikační nástroj mezi počítačem a uživatelem. Z počátku vznikl takzvaný strojový jazyk, který se skládal z posloupností instrukcí. Zde byla velká nevýhoda číselný zápis instrukcí, proto bylo potřeba vytvořit jakýsi překladač symbolických informací na numericky kódovanou informaci. Tento překladač se nazývá Assembler. Později v 50. letech 20. století zahájil éru vyšších programovacích jazyků jazyk Fortran.

Fortran byl jako první snadno naučitelným výkonným jazykem. Bohužel však původně byl vyvinut firmou IBM vyvinut pro vědeckotechnické výpočty a tak nemohl vyřešit úplně vše. Proto bylo potřeba vytvořit jazyk s ucelenou a jednotnou množinu jazykových prostředků pro popis algoritmů, což byl Algol 60. Zde teprve mohli být vytvářeny programy metodou shora dolů. To vedlo ke vzniku strukturovaného programování.

Díky strukturovanému programování se v krátké době objevily další jazyky, které z Algolu 60 vycházeli jako například jazyk Pascal. Ten byl navržen v roce 1971 profesorem Eidgenoessische Technische Hochschule Zurich Niklausem Wirthem pro výukové účely.[1] O rok později 1972 v Bellových laboratoří vzniknl jazyk C s mnoha datovými typy, který má jako velkou výhodu přímý přístup k hardwaru, proto se často používal na programování operačních systémů. Asi největší výhoda jazyka C je jeho rozšiřitelnost. To znamená že, si můžu napsat novou funkci, když ji potřebuju. Toto je možno díky snadno rozšiřitelné velikosti knihoven. Z jazyka C vzniknul roku 1983 jazyk C++.[2]

Jazyk C++ měl v sobě množství vylepšení. Jako je možnost psaní takzvaných šablon, kde si třídu nebo funkci parametrizujeme, aby byla univerzální pro jakýkoliv datový typ. Další novinkou jsou prostory jmen (označovány jako namespace) a možnost přetěžování funkcí a operátorů. A nakonec nejdůležitější novinkou byla podpora objektově orientovaného programování (OOP).[3]

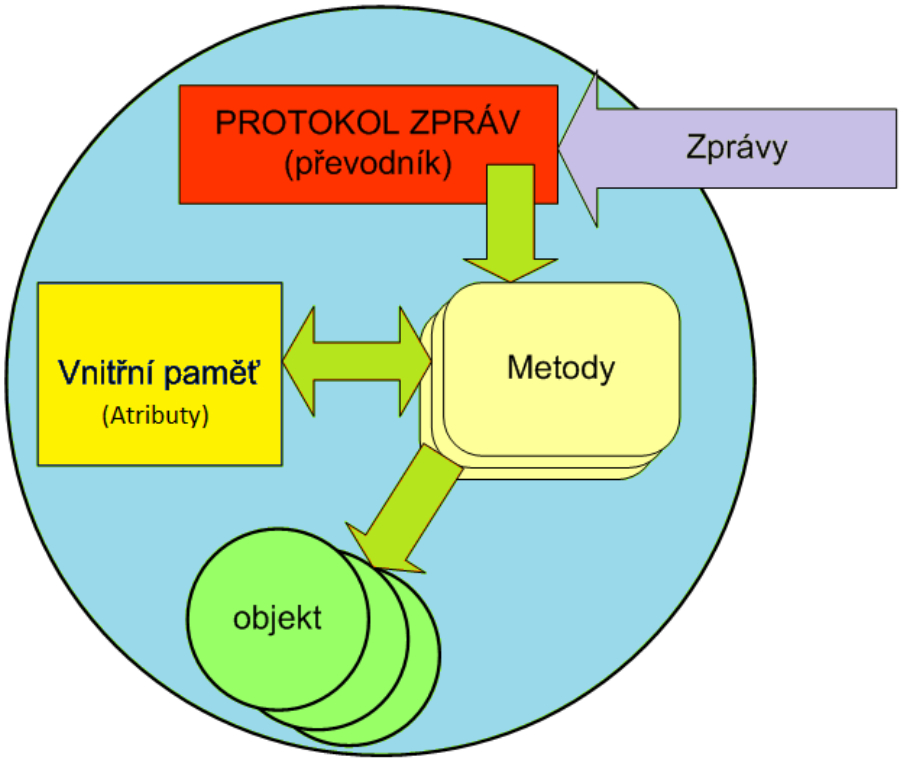
[1] Zdroj: Krubová, Katerina. Historie programovacích jazyku. [online]. 2000 [cit. 2010-04-21] Kolokviální práce. Masarykova Univerzita, Fakulta informatiky.
[2] Zdroj: C_(programovací_jazyk) In Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, 2006, 2010 [cit. 2010-04-20].
[3] Zdroj: NĚMEC, Jan. C/C++ (31) - Jazyk C++, historie, charakteristika, vztah k C. Linuxsoft.cz [online]. 9.1.2006 , č. 31, [cit. 2010-04-20].

1.2 Objektově orientované programování (OOP)

Zde nastává velký rozdíl oproti strukturovanému programování a tím je větší důraz na data. Základním pojmem je zde objekt.

1.2.1 Objekt

Objekt si můžeme představit jako entitu, která v sobě zapouzdřuje svůj stav a chování. Aby se tohoto docílilo, musí objekt v sobě mít vnitřní paměť k uchovávání dat, pak metody vykonávající práci nad těmito daty. Dále v sobě může obsahovat i jiné objekty a nakonec by určitě neměl chybět protokol zpráv. Ten slouží k předání zprávy, která má úkol zavolat určitou metodu objektu viz obrázek 1.



Obrázek 1 - Ukázka objektu v OOP[4]

1.2.2 Základní vlastnosti

- Zapouzdření – což je zabalení atributů a metod do jednoho objektu viz obrázek 1.
- Dědičnost – je vztah mezi třídami, které mají něco společného. Tím se docílí k zamezení duplicitě při definici tříd.
- Polymorfismus – představuje situaci, kdy více objektů má v sobě stejný název metody, přičemž každý objekt ji může mít jinak definovanou. To nastává především ve vzniklých potomcích, při použití dědění.

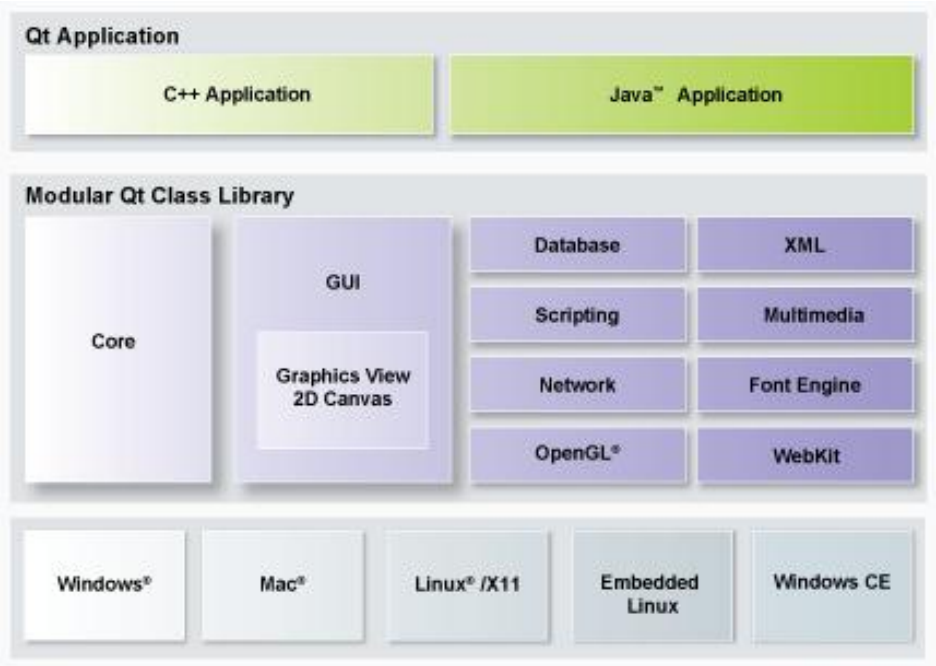
[4] Zdroj:Vlastní

1.3 Moderní programovací jazyky

Zde je příklad nejpoužívanějších programů pro tvorbu aplikace s podporou OOP.

1.3.1 Jazyk C++ s knihovnou Qt

Určitě bych zmínil hodně populární knihovnu Qt pro vytváření programu s graficky uživatelským rozhraní. Její největší výhodou je funkčnost na jakémkoliv operačním systému, je tedy multiplatformní. Tato knihovna využívá standardní třídy a funkce C++, ale také přidává velké množství vlastních tříd takzvaných Q objektů. Ke komunikaci mezi objekty Qt knihovny zde slouží signály, které zpracovávají sloty. Slot můžeme chápat jako funkci s více možnostmi. Dá se tedy říci, že díky signálům a slotům je Qt multiplatformní. Na obrázku 2 vidíte schéma Qt knihovny.



Obrázek 2 - Ukázka Qt knihovny[5]

1.3.2 Java

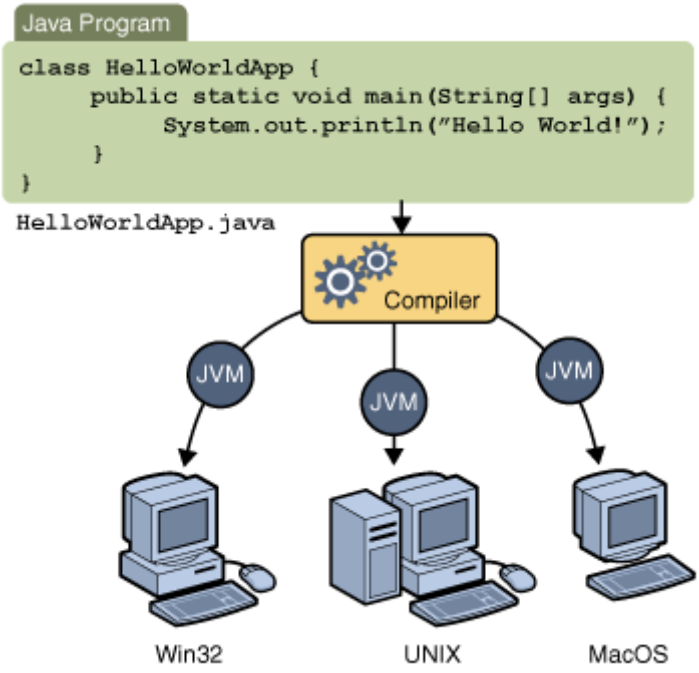
V roce 1995 přišla firma Sun s jazykem Java. Mezi jeho hlavní přednosti patří nezávislost na konkrétním hardwaru či operačním systému viz obrázek 4. To zajišťuje jeho interpretace prostřednictvím virtuálního počítače. Java k práci potřebuje soubor základních nástrojů k vývoji aplikací, který se nazývá Java Development Kit (JDK), později označován jako Software Development Kit (SDK). Java již obsahuje silnou typovou kontrolu, kde se už nepoužívají ukazatelé a také byla odstraněna vícenásobná dědičnost oproti C++.

Průběh spuštění programu v Javě pak probíhá, že nejdříve musí být napsán zdrojový kód do souboru s koncovkou .java, Poté se zdrojový kód zkompiluje do byte-

[5] Zdroj: Dostupný na WWW http://docs.huihoo.com/qt/

kódu s koncovkou .class. Tento byte-kód pak interpretuje v počítači Java virval machine (JVM), to vše je vidět na obrázku 3.

Obrázek 3 - Ukázka vývoje softwaru v Javě [6]



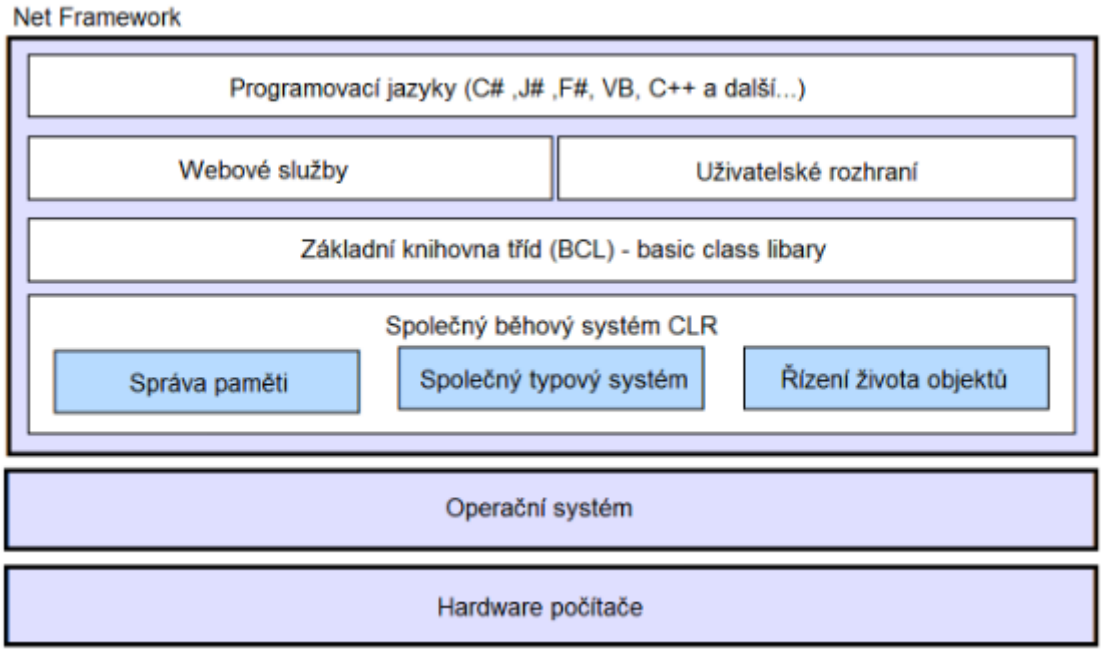
Obrázek 4 - Ukázka multiplatformosti Javy[7]

1.3.3 C#

Později přišla firma Microsoft s jazykem C# s platformou .NET Framework. Jazyk C v sobě obsahuje takzvaný *garbage collector*, který nám zajišťuje automatickou správu paměti. Zpracování chyb se zde může řešit pomocí výjimek. Tento jazyk je jako Java *case sensitive*, což znamená, že rozlišuje velká a malá písmena při programování a také nepodporuje vícenásobnou dědičnost. Protože je tento jazyk čistě objektově orientovaný, tak k atributům se dostáváme přes metody *get* a *set*, které jsou součásti *properties* atributy. Jako velké usnadnění pro programátory je, že neobsahuje a ani nepotřebuje dopředí deklaraci a díky tomu není důležité pořadí deklarace metod.[8]

[6] Zdroj: Dostupný na WWW http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html
[7] Zdroj: Dostupný na WWW http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html
[8] Zdroj: BĚHÁLEK, Marek. Programovací jazyk C#. [online]. 2007 [cit. 2010-04-26].

.NET Framework je rozhraní, které zajišťuje komunikaci mezi komponentou a klientem v bezpečném prostředí. Obsahuje v sobě dvě základní části. První je společná knihovna tříd Basic Class Library (BCL), která je společná pro všechny jazyky v systému .NET. Druhá část je běhový systém Commin Language Runtime (CLR), což je virtuální stroj, ve kterém běží všechny aplikace systému .NET. Vše můžete vidět na obrázku 5 níže.[9]



Obrázek 5 - Ukázka schématu systému .NET Framework[10]

2 Uchovávání dat

Data se v aplikaci dají uchovat mnoha způsoby. Ale pro každou aplikaci se musí předem promyslet, který způsob je pro konkrétní aplikaci nejefektivnější. Z počátku této práce bylo myšleno, že data se budou uchovávat v relační databázi pomocí tabulek. Při podrobnější analýze se však tento způsob uložení dat ukázal jako neefektivní.. Proto jsem zvolil uchovávání dat v souborovém formátu CSV.

2.1 CSV soubor

Zkratka CSV je z anglického comma separated values, což v českém překladu znamená hodnoty oddělené čárkami. Tento si můžeme představit jako textový soubor, kde jsou hodnoty uchovávány v řádcích a mezi těmito hodnotami jsou takzvané oddělovače. Oddělovač může být, jak bylo řečeno, čárka i středník. To jsou dva nejpoužívanější oddělovače. Při použití čárky by hodnoty měli být ještě v uvozovkách, protože může nastat případ, kdy v hodnotě se objeví také čárka, viz desetinné místo, nebo čárka ve větě. Tento příklad vidíte níže.

[9] Zdroj: HORVÁTH, Tomáš. .NET Framework. *Programujte* [online]. 2008 , 1, [cit. 2010-04-25]
[10] Zdroj: Dostupný na WWW http://programujte.com/?akce=clanek&cl=2008120700-net-framework

Tabulka 1 - Tabulka před uložením do CSV souboru

Kód	Název	Cena bez DPH	Cena s DPH	čas
2947	demontáž krytu rozvaděče do š.0,7m	100	119	0,5
611	oceloplechový zákryt roštu š.30cm	200	238	0,237

Tuto tabulku lze pak zapsat do csv souboru takto:

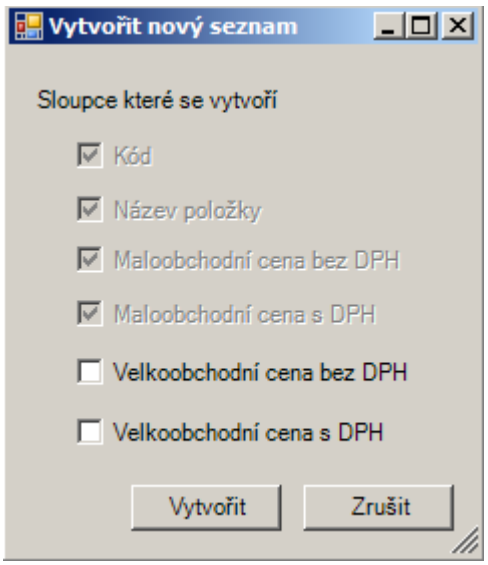
Kód,Název,Cena bez DPH,Cena s DPH
2947,"demontáž krytu rozvaděče do š.0,7m, ",100,119,"0,5"
611, oceloplechový zákryt roštu š.30cm,200,238,"0,237"

3 Uživatelská část aplikace

Tato aplikace je určená pro návrh rozpočtových prací se zaměřením na elektrikářské práce. Je tedy určená pro osoby, které potřebují snadno vytvořit návrh rozpočtu neboli takzvaného. montážního listu.

3.1 Vytvoření vlastního seznamu zboží

K vytvoření vlastního seznamu musíte v horní nabídce kliknout na záložku *Soubor* a dále na položku *Vytvořit nový*. Zobrazí se vám okno, kde si zvolíte, jestli k sloupcům Kód, Název, Maloobchodní cena bez DPH a Maloobchodní cena s DPH budou patřit sloupce Velkoobchodní cena bez DPH a Velkoobchodní cena s DPH viz obrázek 6. Poté vám vyskočí okno pro zadání první položky do seznamu zboží.



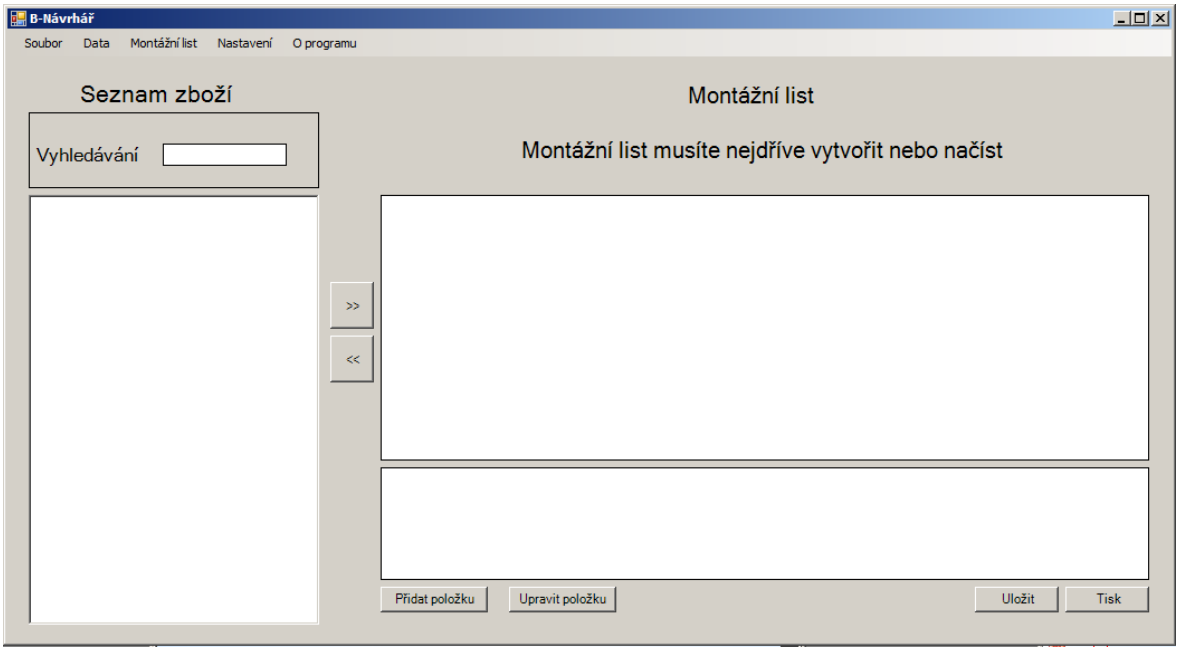
Obrázek 6 - Ukázka vytvoření nového seznamu[11]

[11] Zdroj: Vlastní

3.2 Importování dat do aplikace

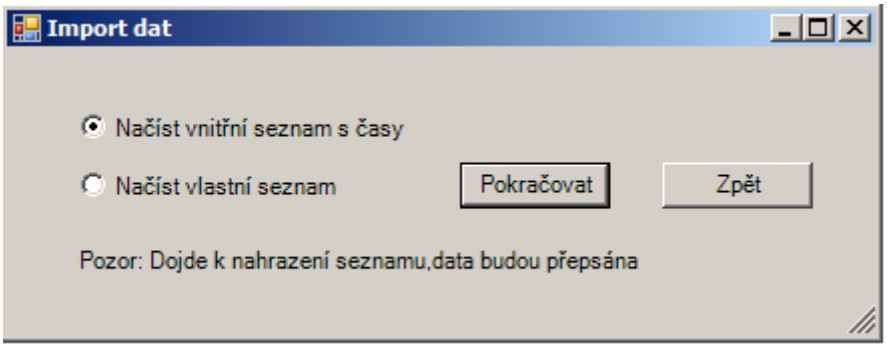
Nejprve je důležité mít pečlivě zhotovený vlastní seznam zboží. Tím je myšleno, že ve sloupci název by měli být položky pojmenovány celým názvem, namísto používání zkratek. Tím pak vzniká výrazně menší úspěšnost spárování s položkami a normami času.

Při spuštění aplikace se uživateli načte hlavní formulář, který vidíte na obrázku 7. Pokud jste již provedli import dat, tak se vám v levé části načte seznam zboží. Jestliže ne, tak nejdříve musíte v horní nabídce kliknout na *Data* zvolit položku *Import dat.*



Obrázek 7 - Ukázka hlavního formuláře[12]

Po kliknutí na položku Import dat se vám zobrazí okno viz obrázek 8, kde si zvolíte, jaký seznam chcete naimportovat.



Obrázek 8 - Ukázka importu Data[13]

[12] Zdroj: Vlastní
[13] Zdroj: Vlastní

Při zvolení první možnosti *Načíst vnitřní seznam s časy* se vám opět zobrazí hlavní formulář, ale s tím rozdílem, že v levé části bude načtený seznam zboží. Jinak při zvolení druhé možnosti *Načíst vlastní seznam* se nám otevře otevírací dialogové okno, kde si z výběru zvolíte formát, v kterém máte svá data a otevřete svůj soubor. Tento soubor musí však dodržet pořadí a počet sloupců viz tabulka 1 a tabulka 2.

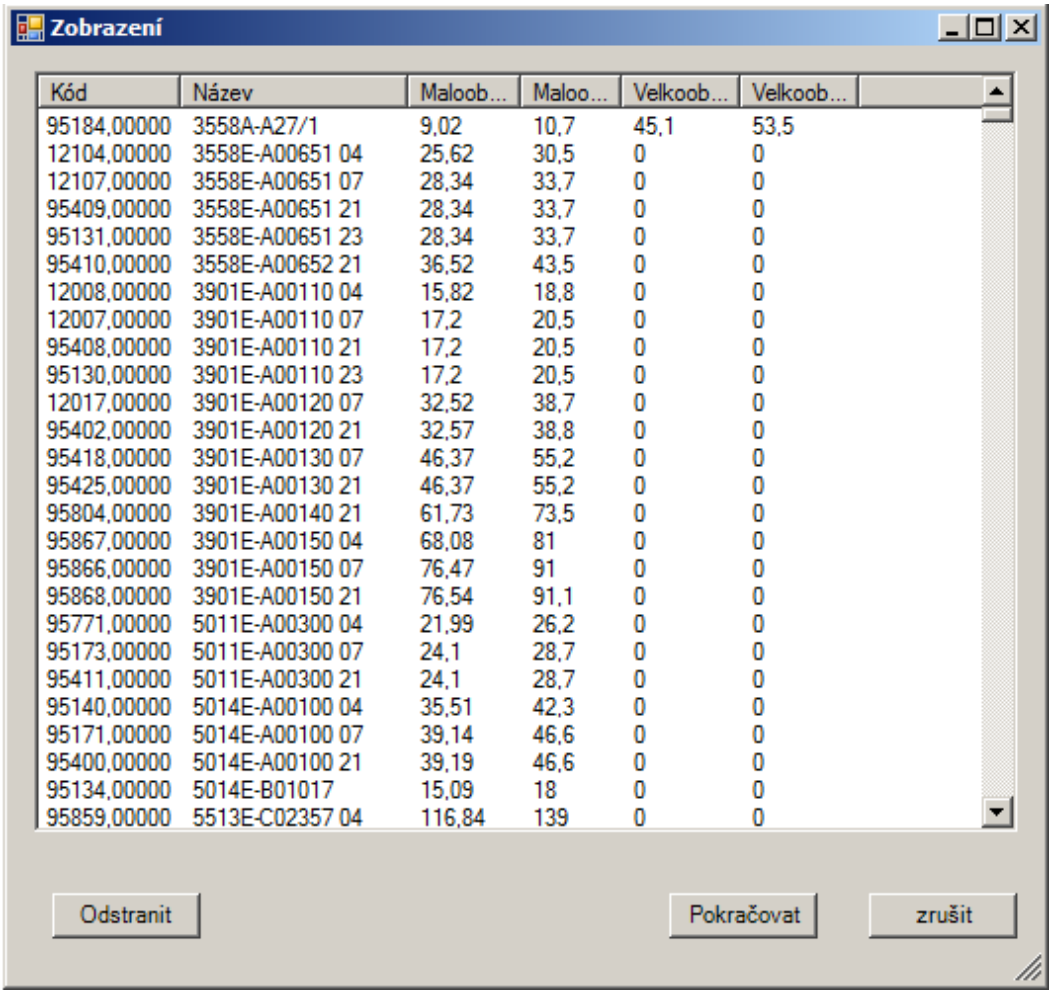
Tabulka 2- První vzor sloupců

Kód	Název	Maloobchodní cena bez DPH	Maloobchodní cena s DPH

Tabulka 3 - Druhý vzor sloupců

Kód	Název	Maloobchodní cena bez DPH	Maloobchodní cena s DPH	Velkoobchodní cena bez DPH	Velkoobchodní cena s DPH

Po otevření se zobrazí formulář, kde se zobrazí data ze souboru, který jste načetli. Ten slouží pouze pro kontrolu, jako na obrázku 9.



Obrázek 9 - Ukázka zobrazení dat, před importem[14]

Po kliknutí na tlačítko *pokračovat* se spustí párovací algoritmus. Po skončení párování se zobrazí okno s informací kolik položek bylo spárováno a kolik jich zbývá. Po odkliknutí informačního okna se vám budou postupně zobrazovat okna s jednotlivými položkami, které se nedokázali spárovat s normami času ve vnitřním seznamu, dokud nekliknete na tlačítko *Dokončit*.

3.3 Přidání nové položky do seznamu zboží

Přidat novou položku do seznamu si můžete po kliknutí v horní nabídce na *Data* a položku *Přidat položku*. Zobrazí se vám okno, kde vyplníte její údaje a uložíte.

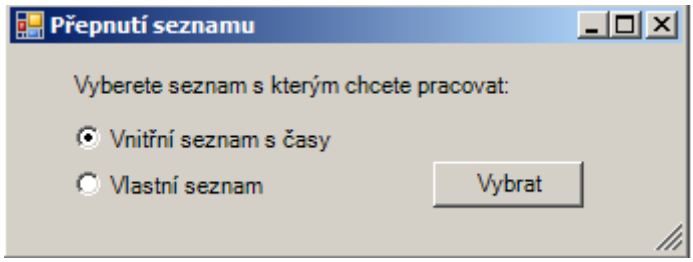
3.4 Odebrání položky ze seznamu zboží

Odebrat položku ze seznamu můžete kliknutím kliknutí v horní nabídce na *Data* a položku *Odebrat položku*. Zobrazí se vám okno, kde budou všechny položky seznamu, kde označíte položku, kterou chcete smazat a kliknete vlevo dole na tlačítko *Odstranit*. Důležité je, pokud chcete tyto změny uložit, kliknout na tlačítko *Pokračovat*.

[14] Zdroj: Vlastní

3.5 Přepnutí pracovního seznamu

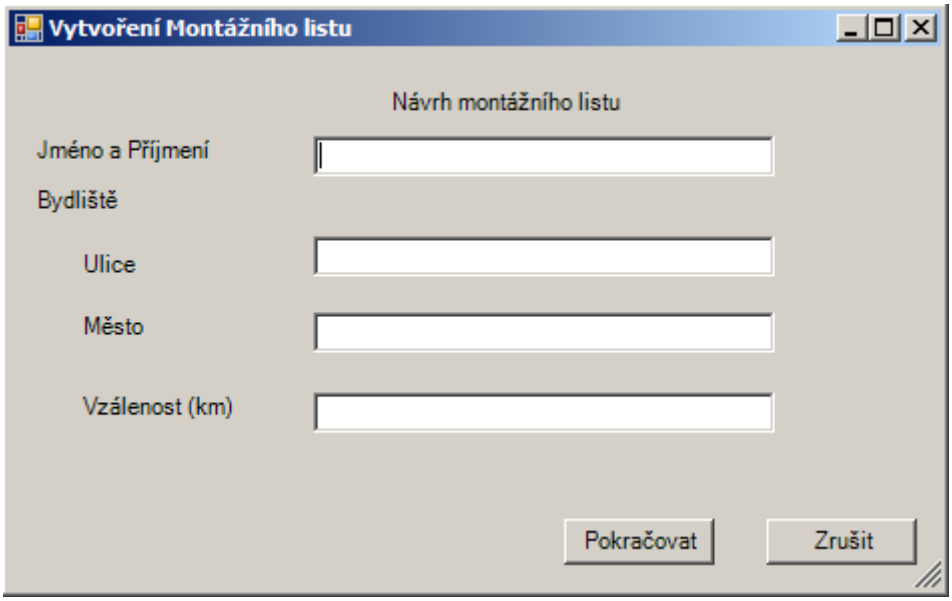
Někteří uživatelé mají svůj vlastní seznam zboží, který si načtou do aplikace, ale také by chtěli občas využít vnitřní seznam, který má u všech položek normu času. To se uskuteční v horní nabídce po kliknutí na *Data* na položku *Přepnout*. Zobrazí se vám formulář jako na obrázku 10.



Obrázek 10 - Ukázka přepnutí pracovního seznamu[15]

3.6 Vytvoření montážního listu

K vytvoření návrhu rozpočtu je potřeba vytvořit montážní list, který vytvoříme po kliknutí na *Montážní list* v hlavní nabídce a na záložku *Vytvořit nový*. Zobrazí se vám formulář, kde zadáte údaje o osobě, pro kterou tento návrh vytváříte, viz obrázek 11.



Obrázek 11 - Ukázka vytvoření nového montážního listu[16]

3.7 Uložení Montážního listu

Pokud máte montážní list, na kterém budete chtít později pokračovat nebo si ho jen někam zálohovat je potřeba ho uložit. To docílíte v horním menu kliknutím na *Montážní list* a položku *Uložit*.

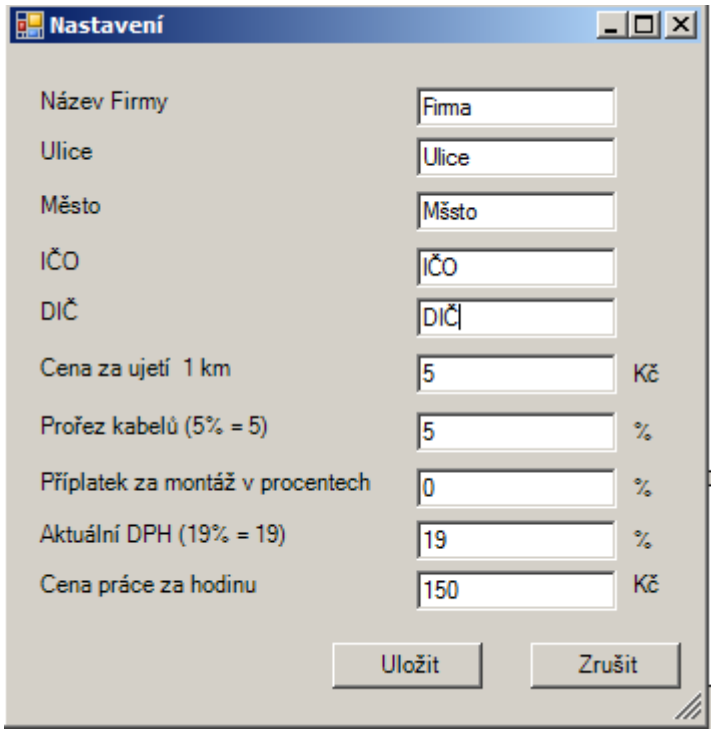
[15] Zdroj: Vlastní
[16] Zdroj: Vlastní

3.8 Načtení montážního listu

K načtení dřív uloženého montážního listu stačí kliknout v horní nabídce na *Montážní list* a na položku *Načíst*.

3.9 Nastavení údajů

Velice důležitá část je nastavení údajů o vás jako uživateli aplikace. To zajistíte v horní nabídce kliknutím na *Nastavení*. Zde si vyplníte informace o vaší firmě a případně vaše ceny, viz obrázek 7.



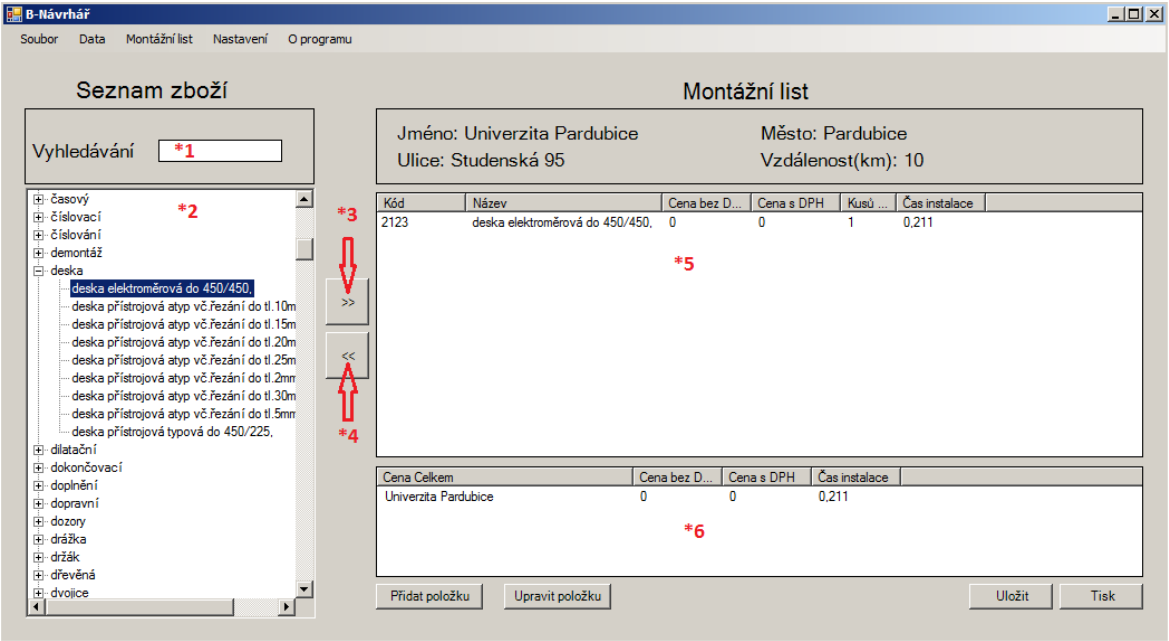
Obrázek 12 - Ukázka nastavení údajů o uživateli[17]

3.10 Práce s montážním listem

3.10.1 Souhrnný popis

K práci s montážním listem je potřeba pár informací, které popíšu na obrázku 8.

[17] Zdroj: Vlastní



Obrázek 13 - Ukázka popisu montážního listu[18]

- 1 - Textové pole pro *Vyhledávání*, slouží k vyhledání kategorií v seznamu zboží, pokud je kategorie nalezena, tak se rozvine a ukáže všechny její položky.
- 2 - Seznam zboží roztříděný na kategorie podle prvního slova.
- 3 - Tlačítko pro přidání označené položky ze seznamu zboží do montážního listu.
- 4 - Tlačítko pro odebrání označené položky z montážního listu.
- 5 - Montážní list.
- 6 - Souhrn ceny a času montážního listu.

3.10.2 Přidání položky do montážního listu

Položku do montážního listu lze přidat buď tlačítkem >> (*3 viz obrázek 8), nebo dvojklikem na položku v seznamu zboží. Pokud chcete přidat položku, která není v seznamu zboží, jako například ostatní náklady, stačí kliknout na tlačítko, které se nachází dole *Přidat položku*.

3.10.3 Smazání položky z montážního listu

Položku lze z montážního listu smazat pouze jedním způsobem a to tlačítkem << (*4 viz obrázek 8).

3.10.4 Úprava položky v montážním listu

Položku v montážním listu je možné upravit. To můžeme dvojím způsobem a to stisknutím tlačítka *Upravit položku* nebo dvojklikem na položku v montážním listu. Upravit jde pak nejen počet kusů či metrů té položky, ale i název, cenu a normu času položky. To se po stisknutí tlačítka *Uložit* uloží i do seznamu zboží.

[18] Zdroj: Vlastní

3.10.5 Tisk montážního listu

Pokud máte vytvořený montážní list je skoro samozřejmé, že ho budete chtít vytisknout pro osobu, které tento návrh děláte. K tomu stačí stisknou tlačítko *Tisk*, které se nachází vpravo dole.

4 Programátorská část

Tato aplikace je naprogramovaná v jazyce C# pomocí programátorského nástroje Microsoft Visual Studio 2008 s prostředím .NET Framework 3.5 SP1.

4.1 Postup práce na aplikaci

Ze začátku jsem se rozmýšlel jaký jazyk si zvolit k programovaní. Rozmýšlel jsem se především mezi jazyky Java a C#. S Javou jsem se již setkal a zaujala mě svojí jednoduchostí a účinností. Ale jak už bylo výše zmíněno tak C# vzniknul z Javy a tak přechod na něj by nebyl zas tak složitý, díky podobné syntaxi. Dalš výhodou jazyka C# bylo vývojové prostředí Visual studio, ve kterém se s C# může pracovat a s kterým jsem se dříve již setkal. Toto prostředí s Frameworkem .NET mi bylo natolik sympatické, že jsem jej zvolil pro svou práci.

Po výběru programovacího jazyka a vývojového prostředí následovalo bližší seznámení s jazykem C#. Dále bylo potřeba získat data od středně velké firmy a normy času které se vztahují na elektrikářské úkony. Nad získanými daty jsem následně musel provést jejich analýzu. V analýze se muselo rozhodnout jaká data zde budou důležitá a jak a kde se tyto data budu uchovávat. To vedlo ke vzniku prvních dvou tříd a to `ItemInputDatabase` a `ItemMyDatabase` do kterých se tyto data nahrála.

Získaná data bylo potřeba načíst ze souboru do vývojového prostředí. Zde jsem zvolil cestu pomocí `interfacu`, což by se dalo chápat jako takový rodič, který ví, že bude načítat list typu `ItemInputDatabase`, ale neví jak. Od této třídy `Iload` jsem pak vytvořil dva potomky, a to třídu `loadCSV` pro načítání formátu CSV a `loadDBF` pro načítání databázového formátu DBF.

Dále bylo potřeba získaná data spárovat podle jejich názvů, aby se k uživatelovým položkám přidala jejich norma času. Zde jsem narazil na problém s názvy položek, kde bylo potřeba získat klíčová slova, podle kterých by se párovalo. Toto řeší třída `WorkWithTables`. Ta v sobě obsahuje metody pomocí kterých jsem z názvu odstranil předložky a ponechal slova s minimálně čtyřmi znaky. Zbylé klíčové řetězce z názvu se ukládali do Listu s řetězci.

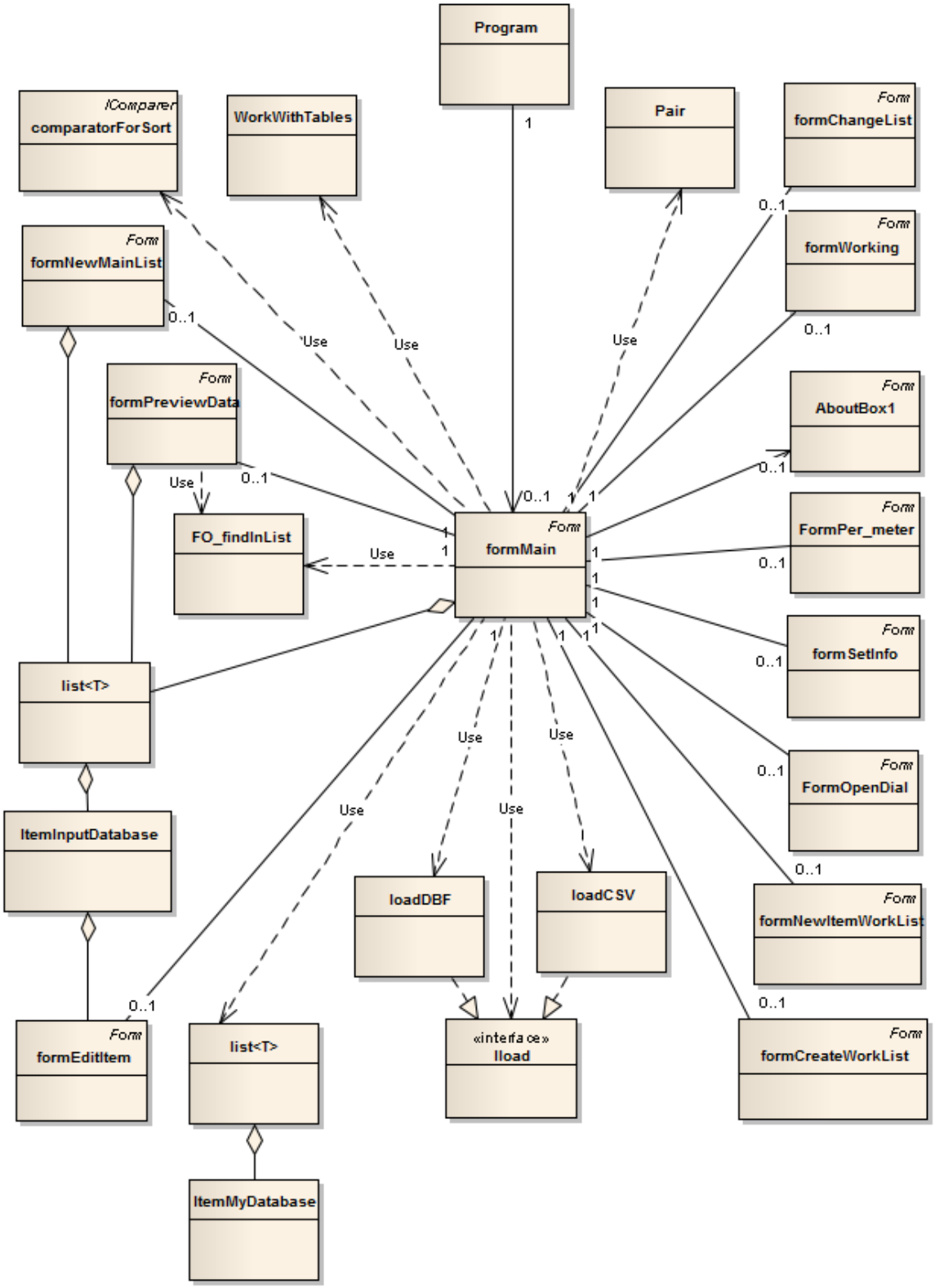
Když už jsme měli data připravená na párování, bylo potřeba vytvořit třídu, která by to zajistila. Třídu jsem pojmenoval `Pair`. Z počátku jsem měl myšlenku porovnávat dva listy řetězců. Jeden z názvu uživatelovy položky a druhý z položky s normou času. Výsledkem by pak bylo počet shod vstupních listů. Po serii testování jsem narazil na

problémy s diakritikou a zkratek. Proto jsem vytvořil další funkci, která nahradila písmena s diakritikou jejich protějšky. Při porovnávání jsem použil funkci `indexOf` kvůli odhalení zkratek (svítilna / svit.).

Po vytvoření spárovaného seznamu položek jsem vytvořil hlavní formulář `formMain` a k němu postupně přidával ostatní podle potřeby. Zde nevznikaly žádné větší problémy, až ke konci, kdy jsem vytvářel funkci pro tisk výsledného návrhu. Zde nastalo chybné zobrazování textu při tisknutí více stran. Po třítýdenním hledáním jsem objevil chybející kód. Poté jsem již dolaďoval menší chyby, které se postupem objevovaly.

Diagram Tříd

Na obrázku 14 vidíte diagram tříd celé aplikace.



Obrázek 14 Ukázka Diagramu tříd[19]

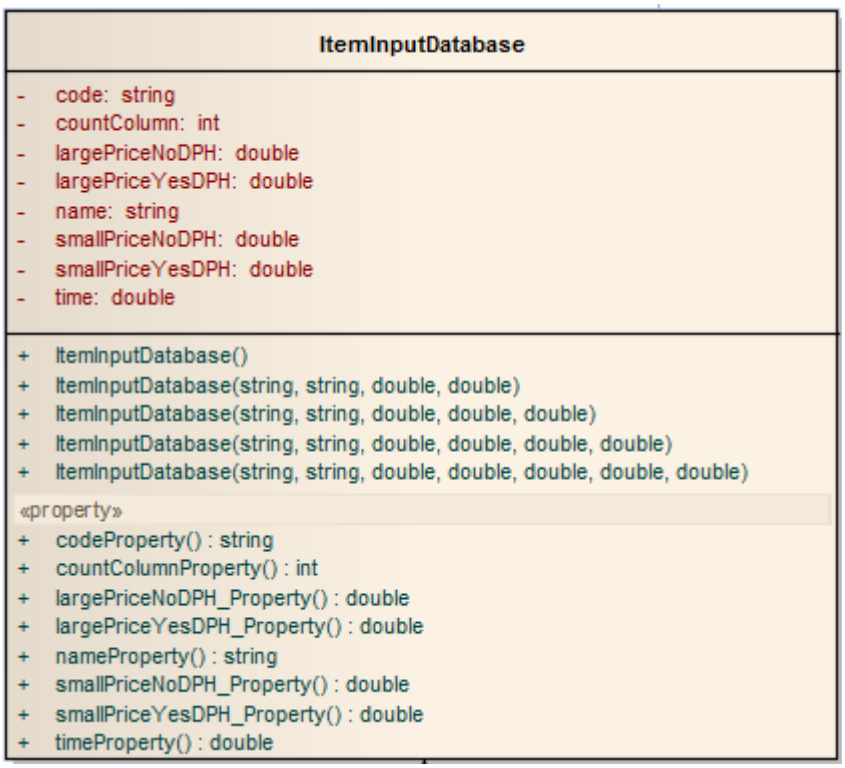
[19] Zdroj: Vlastní

4.2 Popis objektů

Zde bych chtěl popsat základní vlastnosti tříd a formulářů.

4.2.1 Třída ItemInputDatabase

Je důležitým prvkem aplikace, protože v této třídě se uchovává jedna položka seznamu, která obsahuje kód, název, cenu a čas, za který bude položka zamontována. Dále obsahuje takzvané `Properti`, které v sobě mají metody `Set` a `Get` k získání nebo nastavení atributů a konstruktory této třídy viz obrázek 15.

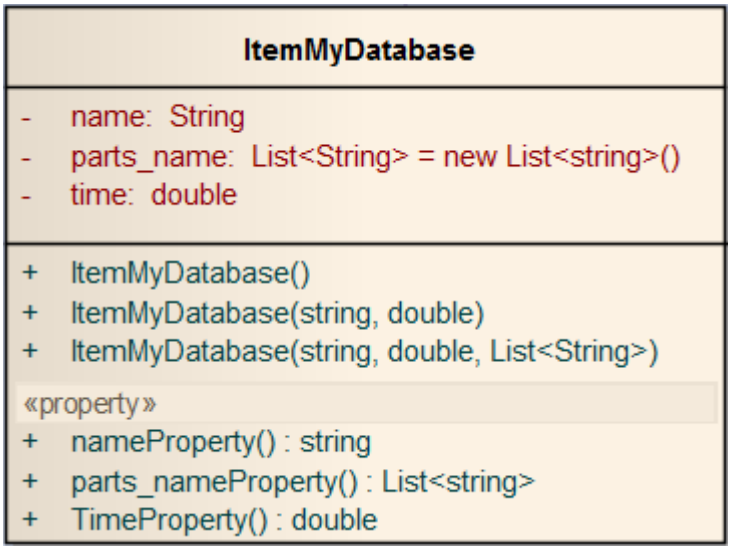


Obrázek 15 - Ukázka diagramu třídy ItemInputDatabase[20]

4.2.2 Třída ItemMyDatabase

Do této třídy se nahraje jedna položka ze seznamu normy času, kterou má v sobě aplikace uloženou v binární podobě. Potom tato třída slouží ke spárování položek podle schody v názvu nebo vytvoření samotného listu s normami času. Obsahuje název, čas jedné položky a list jednotlivých klíčových slov v jeho názvu, viz obrázek 16.

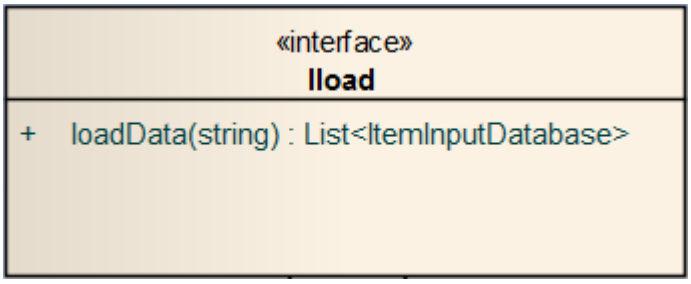
[20] Zdroj: Vlastní



Obrázek 16 - Ukázka diagramu třídy ItemMyDatabase[21]

4.2.3 Interface Iload

Tento interface na obrázku 17 slouží k načtení dat ze souborů různých formátů, které díky interfacu můžeme jednoduše rozšiřovat. Stačí pouze vytvořit potomka tohoto interfacu a v něm nadefinovat způsob načítání nového formátu. Je zde nadeklarovaná metoda `loadData`, která má návratovou hodnotu `List<ItemInputDatabase>`, ale dále není definováno, jak tuto hodnotu získá.

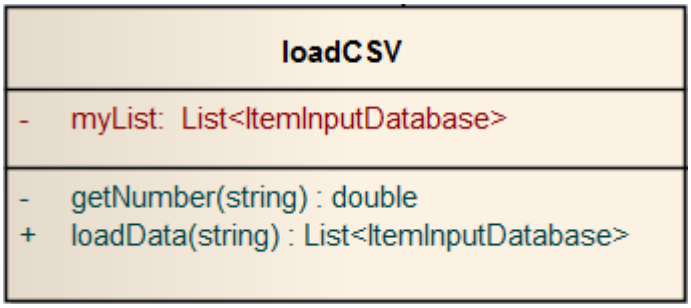


Obrázek 17 - Ukázka diagramu třídy Iload[22]

4.2.4 Třída loadCSV

Tato třída je potomkem interfacu `Iload` a slouží k načtení dat z textového formátu CSV pomocí zděděné funkce `loadData` viz obrázek 18. V této funkci je již nadefinovaný přesný postup, jak tyto data získá ze souboru typu CSV. Dále také obsahuje metodu `getNumber`, která získá z řetězce pouze číselnou část a vrátí ji v datovém typu double.

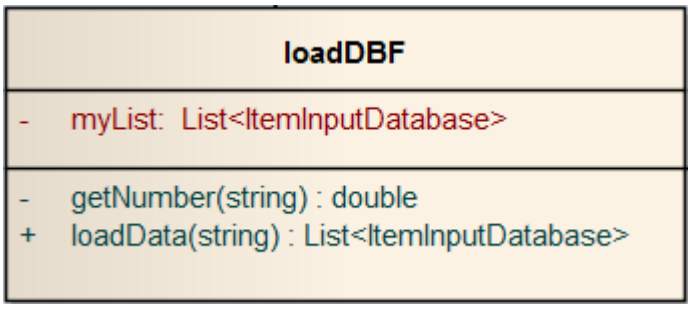
[21] Zdroj: Vlastní
[22] Zdroj: Vlastní



Obrázek 18 - Ukázka diagramu třídy loadCSV[23]

4.2.5 Třída loadDBF

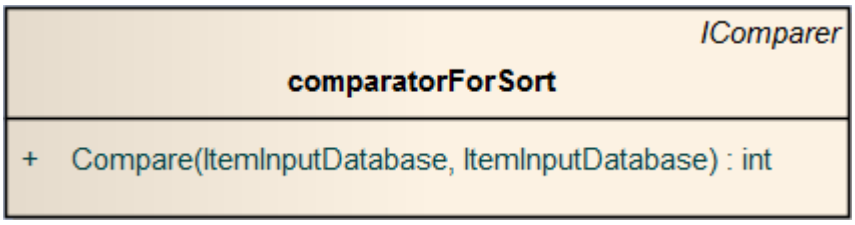
Tato třída je potomek interfacu `Iload` a slouží k načtení dat z databázového formátu DBF pomocí zděděné funkce `loadData` viz obrázek 19. V této funkci je již nadefinovaný přesný postup jak tyto data získá z databázového souboru DBF. Dále také obsahuje metodu `getNumber`, která získá z řetězce poze číselnou část a vrátí ji v datovém typu double.



Obrázek 19 - Ukázka diagramu třídy loadDBF[24]

4.2.6 Třída comparatorForSort

Tato třída je potomkem rozhraní `IComparer` a obsahuje pouze jednu metodu `Compare` viz obrázek 20. Ta slouží k setřídění položek v seznamu, které jsou typu `ItemInputDatabase`. Třídí se vzestupně podle názvu položky.

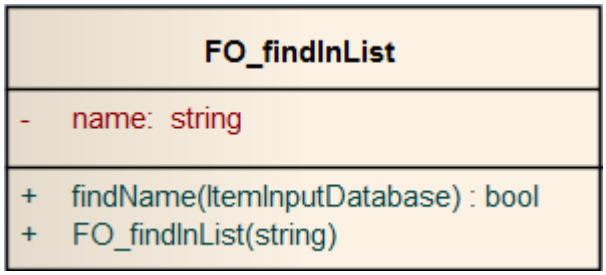


Obrázek 20 - Ukázka diagramu třídy comparatorForSort[25]

[23] Zdroj: Vlastní
[24] Zdroj: Vlastní
[25] Zdroj: Vlastní

4.2.7 Třída FO_findInList

Slouží k vyhledávání jedné položky typu `ItemInputDatabase` v seznamu podle jejího atributu `name`. To zaručuje funkce `findName`, která má jako vstupní parametr typ `ItemInputDatabase` a vrací typ `bool` podle toho jestli se názvy shodují viz obrázek 21. Předtím však musí být pomocí konstruktoru nastaven název hledané položky. Tato třída se používá jako parametr v objektu `Predicate`, který je pak sám parametrem ve funkci `Find` v objektu `List<T>`.



Obrázek 21 - Ukázka diagramu třídy FO_findInList[26]

4.2.8 Třída Pair

Tato třída na obrázku 22 slouží k párování seznamu uživatele se seznamem s normy času.

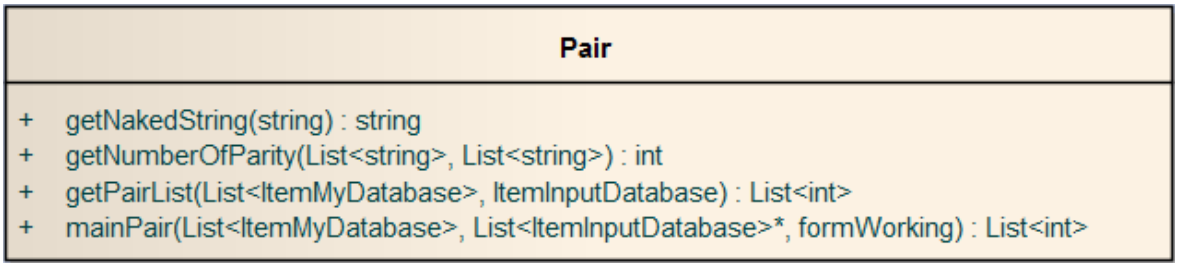
Funkce `getNakedString` upraví vstupní řetězec tak, že nahradí všechny písmena s diakritikou za jejich protějšky bez diakritiky. Tento řetězec pak vrátí zpět

Další funkcí je `getNumberOfParity`, do které vstupují dva listy s řetězcem. Tyto řetězce se pak porovnávají funkcí `indexOf` pro vyhledání schody buď celého řetězce nebo aspoň jeho části, v případě zkratky. K tomu se také využije funkce `getNakedString`. Jako návratová hodnota je počet řetězců, ve kterých se shodují.

Předposlední funkce je `getPairList`, která má za úkol projet všechny položky v seznamu s normami času a pomocí funkce `getNumberOfParity` zjistí indexy všech položek se stejnou shodou v názvu jako hledaná položka.

Funkce `mainPair` „vezme“ každou položku seznamu uživatele a porovná ji se všemi položkami seznamu s normami času. Porovnávání probíhá pomocí funkce `getPairList`, která vrátí list indexů položek se stejným počtem shod. Pokud se v Listu nenachází žádná položka, musí pak čas zadat uživatel. Když je v Listu pouze jedna položka, tak se její čas přiřadí k položce ze seznamu uživatele. Problém, je pokud se v Listu nachází více položek. Toto jsem vyřešil tím, že se z Listu zjistí maximální a minimální čas položek a položce uživatele se pak přiřadí maximální čas za předpokladu, že maximální čas je menší jak 6 minut nebo pokud platí vzorec `(max * 0.3) >= (max - min)`. Jestliže nastane situace, kdy ani jedno neplatí, musí pak uživatel čas zadat ručně.

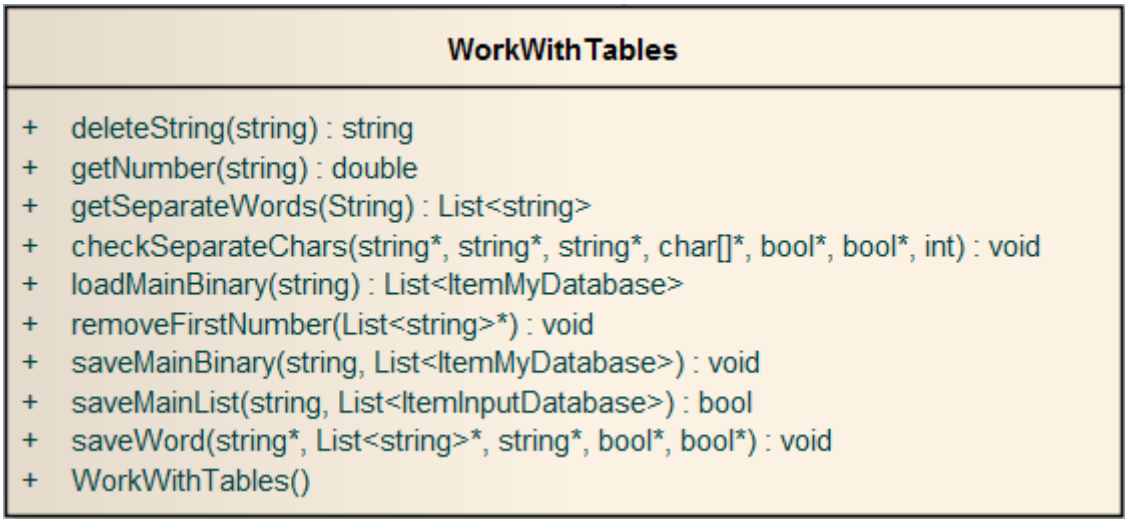
[26] Zdroj: Vlastní



Obrázek 22 - Ukázka diagramu třídy Pair[27]

4.2.9 Třída WorkWithTables

V této třídě se nacházejí pouze pomocné metody, jako je třeba binární načítání a ukládání seznamu s normami času. Tato binární podoba však nejdříve musela být načtena, a to v podobě textového souboru. V tomto souboru se na každém řádku nachází název elektrikářské práce a její norma času, za který bude provedena. Tyto dvě položky se nahrají do atributu `name` a `time` třídy `ItemMyDatabas`. V této třídě se nachází ještě jeden atribut a to `parts_name` do kterého se nahraje `List` typu `string`. Tento List obsahuje klíčová slova z názvu, která jsou velice důležitá při párování se seznamem uživatele.



Obrázek 23 - Ukázka diagramu třídy WorkWithTables[28]

4.2.10 Formulář formNewMainList

Tento formulář slouží k vytvoření nového pracovního seznamu. Ve formuláři si uživatel zvolí, jaké sloupce chce v seznamu mít, přičemž volí pouze, jestli k sloupcům Kód, Název, Maloobchodní cena bez DPH a Maloobchodní cena s DPH budou patřit sloupce Velkoobchodní cena bez DPH a Velkoobchodní cena s DPH.

[27] Zdroj: Vlastní
[28] Zdroj: Vlastní

4.2.11 Formulář FormOpenDial

K načtení seznamu, neboli importu dat, nám slouží tento formulář. Zde si uživatel zvolí, jestli chce nahrát implicitní seznam s normami času, ale bohužel u žádné položky nebude zadána její cena. Nebo si může uživatel zvolit načtení vlastního seznamu.

4.2.12 Formulář formPreviewData

Je to formulář, který má dvojí využití. První je na zobrazení položek seznamu uživatele a druhý na odstranění některých položek. Stačí jen označit položku a kliknout na tlačítko *Odstranit*.

4.2.13 Formulář formWorking

Je to jednoduchý formulář, který obsahuje pouze `Progress Bar`. Což je zobrazovací prvek pro uživatele, aby věděl, že program zpracovává jeho požadavek.

4.2.14 Formulář formEditItem

Tento formulář slouží buď k editaci a přidání jedné položky do seznamu, přičemž se zobrazí pouze atributy jedné položky. Nebo pro editaci přidané položky v montážním listu, kde je navíc položka pro změnu počtu kusů, popřípadě metrů u kabeláže.

4.2.15 Formulář formChangeList

Je to jednoduchý formulář, který obsahuje pouze 2 přepínače a tlačítko Vybrat. Slouží k přepnutí z implicitního seznamu s normami času na uživatelský seznam, nebo naopak.

4.2.16 Formulář formCreateWorkList

Je to formulář pro vytvoření montážního listu pro návrh rozpočtu, kde se zadají čtyři důležité informace. A to jméno a příjmení, ulice, města a vzdálenost klienta od našeho místa podnikání.

4.2.17 Formulář formSetInfo

Zde ve formuláři se nastavují základní informace o uživatelovi a jeho firmě. Nachází se zde například ceny účtování za ujetí jednoho kilometru, procentuální výše prořezu kabelu, příplatku za montáž, Aktuální výše DPH a cena práce účtovaná za hodinu

4.2.18 Formulář AboutBox1

Pouze informativní formulář o programu.

4.2.19 Formulář formMain

Hlavní formulář celé aplikace, která se nazývá `formMain`. Tento formulář je inicializován v metodě `Main`, která se nachází ve statické třídě `program.cs`. Tento formulář se skládá z horního menu, dále se na levé straně formuláře nachází seznam zboží stromovité struktury, kde se po načtení nachází roztříděné položky seznamu podle prvního slova v názvu položky. Do tohoto seznamu položek se přidávají položky do takzvaného Montážního Listu, který se nachází na pravé části formuláře. Tento list je vlastně výchozí část formuláře, který si uživatel může uložit do počítače a vytisknout.

5 Závěr

Cílem této práce nebylo pouze zhotovení softwaru pro návrh rozpočtu pro elektrikářské práce, ale i seznámit čtenáře s nástroji k jeho zhotovení. To je popsáno v teoretické části, kde je zaznamenán vývoj programovacích jazyků i s vývojem způsobu programování až po dnes nejpoužívanější objektově orientované.

Dále bych chtěl zmínit menší změnu oproti zadání, kde je psáno využití relační databáze. Tato volba však byla zavržena po podrobné analýze a přešlo se k uchování dat pomocí souborového formátu CSV, který jeví snadnější práci s daty.

V praktické části je pak popsán postup při programování a to v jazyce C# s Frameworkem 3.5 ve vývojovém prostředí Visual studio 2008. Při vývoji se v této práci kladl důraz na efektivitu, funkčnost a hlavně přehlednost aplikace pro uživatele.

Cíle této práce byly splněny a dále by se tato aplikace mohla rozšířit mnoha směry. Například, po získání dalších norem času s jiným zaměřením než elektrikářské, by se dala použít ve více odvětvích práce. Dále v této práci by bylo snadné rozšířit načítací formáty uživatelova seznamu. To je možné díky použití interfacu při načítání.

Ke konci bych chtěl jen zmínit, že tato práce pro mne, jako autora, byla velkým přínosem. Na této práci jsem se naučil základní práci v jazyce C#. Tento jazyk se mi jeví jako zatím nejlepší programátorský jazyk, s kterým jsem se dosud setkal.

6 Literatura

[1]**KRUBOVÁ, Kateřina. Historie programovacích jazyků**. [online]. 2000 [cit. 2010-04-21] Kolokviální práce. Masarykova Univerzita, Fakulta informatiky. Elektronická kopie dostupná z WWW: <http://www.fi.muni.cz/usr/jkucera/pv109/2000/xkrubova.htm>.

[2]**C_(programovací_ jazyk) In *Wikipedia: the free encyclopedia*** [online]. St. Petersburg (Florida): Wikipedia Foundation, 2006, 2010 [cit. 2010-04-20]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/C_%28programovac%C3%AD_jazyk%29>.

[3]**NĚMEC, Jan. C/C++ (31) - Jazyk C++, historie, charakteristika, vztah k C.** ***Linuxsoft.cz*** [online]. 9.1.2006 , č. 31, [cit. 2010-04-20]. Dostupný z WWW: <http://www.linuxsoft.cz/article.php?id_article=1058>.

[4]**BĚHÁLEK, Marek. Programovací jazyk C#.** [online]. 2007 [cit. 2010-04-26]. Dostupné z WWW: <http://www.cs.vsb.cz/behalek/vyuka/pcsharp/text/index.html>.

[5]**HORVÁTH, Tomáš. .NET Framework.** ***Programujte*** [online]. 2008 , 1, [cit. 2010-04-25]. Dostupný z WWW: <http://programujte.com/?akce=clanek&cl=2008120700-net-framework>.

7 Seznam příloh

7.1 CD obsahující

- zdrojové kódy aplikace
- instalační soubor aplikace
- Obrázek vývoje programovacích jazyku