UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Systém evidence a řízení práce ve firmě

Petr Zelený

Bakalářská práce
2011

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2010/2011**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Petr ZELENÝ**
Osobní číslo:	**I08203**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Systém evidence a řízení práce ve firmě**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

V úvodní části je nutno provést rešerši stávajících SW řešení pro komunikaci a administrativní řízení firmy. Rešerši je nutné doplnit o porovnání s nově navrhovaným systémem, který bude předmětem této práce. Úvodní část musí obsahovat analýzu navrhovaného řešení, která bude obsahovat popis použitých technologií, návrh databáze a aplikačního řešení.
Cílem této práce je vytvoření funkční aplikace, která bude obsahovat nástroje sloužící pro evidenci a řízení práce v malé a střední firmě.
Aplikace bude umožňovat:
- registraci uživatelů
- přístup uživatelů do systému podle jejich práv
- evidenci pracovních úkolů, jejich termínů a pracovníků zodpovědných za úkol
- evidenci zaměstnanců
- evidenci vnitropodnikových dokumentů, příkazů a směrnic
- vyhledávání úkolů a generování sestav dle zadaných kritérií.

Pro vytvoření aplikace bude využit skriptovací programovací jazyk PHP nebo JAVA a databáze Oracle nebo MySQL.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

***LACKO, Luboslav. Oracle. Správa, programování a použití databázového systému. Brno: Computer press, 2007. 573 s. ISBN 978-80-251-1490-2.**

***JAMES R. GROFF, PAUL N. WEINBERG. SQL kompletní průvodce. Brno: Computer press, 2005. 936 s. ISBN 80-251-0369-2.**

***PHP 6, MySQL, Apache:Vytváříme webové aplikace. Brno: Computer press, 2009. 816 s. ISBN 978-80-251-2767-4.**

Vedoucí bakalářské práce: **Ing. Miloslav Macháček, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **17. prosince 2010**

Termín odevzdání bakalářské práce: **13. května 2011**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2011

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 11. 05. 2011 Petr Zelený

Poděkování

Rád bych poděkoval panu Ing. Miloslavu Macháčkovi, Ph.D. za cenné rady, připomínky a samotné vedení mé práce. Dále bych chtěl poděkovat všem, kteří mě během studia podporovali a byli mi oporou.

Anotace

Tato práce se zabývá problematikou využití softwarových nástrojů pro komunikaci, řízení a evidenci práce ve firmě. Obsahem teoretické části práce je přehled a porovnání softwarových nástrojů řešících tuto problematiku. Praktická část je věnována implementaci vlastního manažerského systému, jehož návrh vznikl na základě analýz provedených v teoretické části této práce.

Klíčová slova

Manažerský informační systém, Java, MySQL, řízení práce, ekonomický systém, MIS.

Title

Work management and work registration system.

Annotation

This project deals with problems of using software tools for communication, management and registration work in the company. Content of theoretical part is an overview and comparison of software solutions for solving problems. Practical part is devoted to implementation management system, which was designed based on the analysis carried out in the theoretical part.

Keywords

Management system, Java, MySQL, work management, economic system, MIS.

Obsah

Seznam zkratek

EIS	Executive Information Systems
ER	Entity Relationship
GTD	Get Things Done
JDBC	Java Database Connectivity
JVM	Java Virtual Machine
OLTP	On Line Transaction Processing
SQL	Structured Query Language
UML	Unified Modelling Language

Seznam obrázků

1 Použití informačních a řídicích systémů v podnicích

Softwarová řešení, která směřovala k podpoře manažerských a analytických úkolů ve vedení společností, se začala objevovat již koncem sedmdesátých let minulého století. Tento rozvoj byl provázán s rozvojem on-line technologií. První komerční produkty se objevily zhruba v polovině let osmdesátých v USA. Tyto nástroje byly založeny na technologii multidimenzionálního uložení a zpracovávání dat. Systémy pracující na tomto principu jsou označovány jako EIS. Následně došlo k dynamickému rozvoji těchto systémů. Na začátku devadesátých let se začali dostávat i na český trh s informačními systémy. [1, kap. 1]

Díky stále většímu objemu dat, která byla uchovávána velkými společnostmi, došlo v USA k rozšíření dalších trendů v uchovávání dat. Jednalo se o takzvané datové sklady (Data Warehouse) a datová tržiště (Data Marts). Díky velkému množství dat umístěných v datových skladech byla otevřena cesta pro nástroje dolování dat. S růstem trhu a konkurence docházelo k neustálému vývoji aplikací a nástrojů tohoto zaměření. Všechny tyto nástroje, informační systémy i pracovní postupy můžeme zahrnout do pojmu business inteligence. Pod tímto pojmem můžeme rozumět celou řadu procesů a aplikací, které podporují manažerské rozhodování a plánovací činnosti podniků. [1, kap. 1]

Vzhledem k velkému rozsahu této problematiky je firemní informační systémy možno dělit podle různých kritérií. Systémy můžeme rozdělit například takto:

- analytické systémy,
- transakční systémy.

Transakční systémy jsou obvykle postaveny na technologii relačních databází a zobrazují aktuální stav transakcí v podniku. Jedná se například o stav účetnictví, evidenci zakázek, evidenci úkolů a podobně. Vzhledem k fungování podniku se tato data poměrně rychle mění. Tyto systémy jsou určené k on-line zpracování dat a jsou označovány jako OLTP systémy. Data získaná prostřednictvím OLTP systému mohou být dále poskytována analytickému systému (jedná se o poměrně časté využití OLTP systémů). Hlavní slabinou těchto systémů je především nízká schopnost zpracovat data do podoby ceněných zdrojů pro rozhodovací procesy. [2]

Analytické systémy pracují nad daty získanými pomocí OLTP systémů (takzvaná primární data). Tyto systémy jsou postaveny nad multidimenzionálními databázemi, které obsahují již agregovaná data. Díky použití multidimenzionálních databází umožňují tyto systémy zpracovávat statistiky z velkého množství dat (řádově stovky tisíc až miliony záznamů) při zachování přijatelné časové odezvy pro uživatele. Tyto databáze rovněž umožňují sledovat souvislosti dat z různých pohledů. [3]

2 Problematika systémů pro řízení a evidenci práce

Problematika informačních systémů ve firmách je velice rozsáhlá a jednotlivé nástroje spolu bývají velmi úzce propojeny. Z tohoto důvodu je poměrně obtížné zařadit danou aplikaci do jediné kategorie. Systémy řízení a evidence práce můžeme zařadit do kategorie manažerských aplikací EIS (Executive Information Systems), nebo do kategorie manažerských informačních systémů MIS (Management Information Systems). Systémy pro řízení a evidenci práce by měli obsahovat dvě základní části, které jsou níže popsány.

- Část, která řeší samotné přidělování úkolů na úrovni pracovníků výroby a s ní spojené vykazování práce.
- Manažerskou část, která umožňuje vyhodnocovat odvedenou práci a zprostředkovává podklady pro manažerské řízení.

2.1 Řízení úkolů

Přehledné a efektivní řízení úkolů je základním stavebním kamenem řízení výroby ve společnosti. Každý pracovník by měl jasně vědět, jaké úkoly má řešit, jejich datum vyhotovení, prioritu a časovou náročnost. V závislosti na těchto informacích může být následně práce koordinována tak, aby byly jednotlivé úkoly včas dokončeny a výroba tak nemusela čekat na dokončení některých dílčích úkolů, které jsou součástí úkolu hlavního.

Úkoly, které jsou výrobou řešeny, mohou být mnohdy velmi náročné a rozsáhlé. Z tohoto důvody by měl systém umožňovat rozdělení hlavního úkolu na několik dílčích úkolů a vytvoření relací mezi těmito úkoly.

2.2 Manažerská část

Dosahování dobrých výsledků a růst společnosti není možný bez manažerského řízení. Manažeři jsou poměrně vytížené osoby, které činí důležitá rozhodnutí s následným vlivem na celý chod firmy. Manažer tedy musí mít k dispozici přehledně uspořádané údaje, které vyjadřují efektivitu, výkonnost a aktuální stav výroby.

Systém musí rovněž umožňovat dohledání informací o práci, která byla odvedena na zakázce, aby bylo možné zjistit slabé články výroby a eliminovat je.

3 Porovnání dostupných produktů na trhu

Současná nabídka nástrojů, které se více či méně dotýkají problematiky řízení a evidence práce ve firmě, je velice početná. Velmi častým jevem je, že konkrétní systém řeší specifickou část problematiky řízení společnosti. Z tohoto důvodu je následně nutné použít těchto nástrojů několik.

Použití několika různých nástrojů má velké nevýhody. Jedná se především o:

1. nejednotné ovládání systémů,
2. nutnost komunikace s různými výrobci,
3. složité propojení nesourodých systémů,
4. různé požadavky na software a hardware.

Jednotlivé kategorie softwaru pro řízení firmy jsou uvedeny v dalších podkapitolách.

3.1 Systémy řízení úkolů

Základním prvkem tohoto systému je úkol (assigment, issue), který obsahuje popis činnosti, která má být vykonána. Každý úkol je přiřazen konkrétnímu uživateli, od kterého je požadováno vyřešení úkolu.

Pro zlepšení orientace obsahují úkoly informace o prioritě, datu vyhotovení, kategorii úkolu, stavu úkolu a podobně. Moderní systémy rovněž umožňují správu verzí úkolu a uchovávání jednotlivých jeho změn. Samozřejmostí bývá i možnost připojit poznámku.

Tyto systémy jsou velice často využívány pro vyvíjení softwarových aplikací, jako takzvané bug tracking systémy (systémy pro odstraňování chyb programu). Několik zástupců této kategorie je uvedeno níže.

- MantisBT[1].
- Bugzilla[2].
- The Bug Genie[3].

Systémy jsou v praxi velmi používané a spolehlivé. Vzhledem k tomu, že jsou určeny pouze pro řízení a evidenci úkolů, neumožňují další operace, jako je vedení výkazů práce, evidence zakázek a podobně.

3.2 Ekonomické systémy

Vedení účetnictví, nebo daňové evidence je povinností každé firmy. Na trhu je opět velké množství softwarových nástrojů, které plně řeší tuto problematiku. V drtivé většině

[1] http://www.mantisbt.org
[2] http://www.bugzilla.org
[3] http://www.thebuggenie.com

se jedná o produkty komerční, což v souvislosti s firemním účetnictvím není nic překvapivého. Vzhledem k tomu, že účetnictví je specifickou částí, tyto programy neumožňují na zavedené zakázky navazovat úkoly či vykazovat práci. Několik zástupců této kategorie je uvedeno níže.

- Pohoda[4].
- Many S3[5].
- Altus Vario[6].

Jak již bylo dříve zmíněno, všechny tři systémy jsou placené a jejich cena se pohybuje v řádech desítek tisíc korun v závislosti na používaných modulech.

3.3 Manažerské nástroje

Vzhledem k tomu, že je v poslední době kladen velký důraz na efektivní řízení, spadá do této kategorie opravu velké množství nástrojů. Do této kategorie můžeme zařadit programy od správců elektronické pošty, přes programy pro plánování projektů, získávání informací z výroby, až po programy určené pro prezentaci. Níže je opět vyjmenováno několik zástupců této kategorie.

- Microsoft Outlook[7].
- Microsoft Project[8].
- Things Mack[9].

3.4 Komplexní nástroje pro řízení a evidenci práce

Na trhu jsou dostupné i nástroje, které problematiku řízení a evidence práce řeší komplexně (většinou bez účetnictví). Tyto systémy bývají často řešeny jako webové aplikace, které umožňují přístup odkudkoli.

Vzhledem k tomu, že potřeby každé společnosti mohou být velmi individuální, jsou tyto systémy často stavěny „na míru“ konkrétní firmě. Cena těchto systémů je poměrně vysoká. V závislosti na zvolené verzi aplikace se pohybuje v řádech desítek až stovek tisíc korun. Několik zástupců této kategorie je představeno a porovnáno v následující kapitole.

[4] www.stormware.cz
[5] www.money.cz/money-s3
[6] www.vario.cz
[7] http://office.microsoft.com/cs-cz/outlook
[8] www.microsoft.com/cze/project2010
[9] http://culturedcode.com/things

4 Porovnání dostupných systémů pro řízení a evidenci práce

Navrhovaný koncept, tedy desktop aplikace, je méně obvyklým v oblasti informačních manažerských systémů. Této oblasti dominují webové aplikace. Po vyzkoušení několika volně dostupných desktop aplikací (žádná neumožňovala přístup ke vzdálené databázi), byly pro porovnání zvoleny tři webové aplikace. Zvolené aplikace disponují obdobnými vlastnostmi, které jsou požadovány v zadání této práce. Níže jsou uvedeny názvy aplikací spolu s jejich výrobci.

- **MarkTime** – software české společnosti CleverApp, s.r.o.
- **WorkWatch** – software české společnosti VERTIGO.CZ, a.s.
- **Easy Project** – software české společnosti Easy Software, s.r.o.

4.1 MarkTime PORTAL

Jedná se o software českého výrobce. Cena aplikace je 25 500 Kč + 1 150 Kč za každého uživatele. Výrobce poskytuje přístup do demoverze aplikace[10] zdarma.

Online aplikace umožňuje následující:

- řízení lidských zdrojů,
- vykazování práce,
- sledování příjmů a výdajů,
- generování faktur a klientských vyúčtování,
- sdílení dokumentů.

Výhody:

- kalendář manažera,
- timeline – manažer má k dispozici časovou osu svých podřízených,
- kapacitní plánování – tvorba plánů pro podřízené,
- klientská vyúčtování a fakturace,
- možnost uživatelské tvorby formulářů,
- velké množství evidovaných údajů,
- velmi komplexní a profesionální aplikace.

Nevýhody:

- nepodporuje správu verzí úkolů a automatické zasílání informačních emailů,
- méně grafických statistik,
- nepodporuje uživatelské nastavení vzhledu,
- nepodporuje uživatelské řízení priorit.

[10] Přístup je možné získat na stránkách výrobce www.vykaz.cz.

Celkové zhodnocení:

Jedná se o profesionální aplikaci, jejíž kvalita dle mého názoru odpovídá ceně. Aplikace umožňuje uchovávání a sledování velkého množství údajů. Velkou výhodou je rovněž možnost tvorby vlastních formulářů a kapacitní plánování pro podřízené.

Menší výhradu mám ke grafickému vzhledu aplikace, který je na dnešní dobu poměrně zastaralý. Rovněž absence podpory prohlížeče Google Chrome je v dnešní době, kdy je poměrně rozšířen, citelnou nevýhodou.

4.2 Work Watch

Aplikace českého výrobce, která je v základní verzi dostupná zdarma. Výrobce poskytuje zdarma přístup do demoverze[11].

Online aplikace umožňuje následující:

- evidence zakázek,
- přehled zakázek a nákladů,
- výkazy práce,
- hlídání termínů zakázek,
- export do ekonomického systému money S3.

Výhody:

- export do ekonomického systému,
- pokročilá uživatelská práva,
- uživatelské nastavení systému,
- aplikace je zdarma,
- možnost fakturace,
- nezávislost na platformě.

Nevýhody:

- nepodporuje tvorbu úkolů,
- nemá grafické statistiky,
- nepodporuje evidenci dokumentů,
- neumožňuje generování výplatních listin,
- vzhled neodpovídá aktuálním trendům v oblasti internetových aplikací.

Celkové zhodnocení:

Jedná se o plnohodnotnou aplikaci, která může být použita v menší firmě. Její hlavní výhodou je možnost exportu do ekonomického systému, což je u neplacené aplikace poměrně neobvyklá věc. Aplikace naopak obsahuje menší množství modulů (absence

[11] Demoverze aplikace je dostupná na stránkách výrobce http://demo.workwatch.cz.

evidence dokumentů, absence modulu úkolů) a dále neobsahuje přehledné grafické statistiky.

4.3 Easy Project

Aplikace českého výrobce. Cena této aplikace je závislá na verzi systému. Aplikace Easy Project Mini je za cenu 390 Kč/měsíc za každého uživatele. Výrobce poskytuje účet[12] pro vyzkoušení aplikace na 14 dní zdarma.

Online aplikace umožňuje následující:

- evidenci projektů,
- evidenci výkazů práce,
- řízení rozpočtů,
- evidenci dokumentů,
- možnost zasílání zpráv.

Výhody:

- kalendář úkolů,
- podrobné nastavení a řízení projektů,
- individuální uživatelské nastavení,
- možnost exportu do pdf a Excelu,
- velmi uživatelsky přívětivá a moderní aplikace.

Nevýhody:

- nepodporuje uživatelské řízení priorit
- méně grafických statistik
- nepodporuje správu verzí úkolů
- neumožňuje připomínání úkolů a událostí.

Celkové zhodnocení:

Dle mého názoru se jedná o nejzdařilejší aplikaci ze tří porovnávaných. Její nástroje umožňují plnohodnotné projektové řízení firmy při zachování maximální přehlednosti a intuitivnosti ovládání. Vzhled a možnosti ovládání odpovídají dnešním trendům webových aplikací. Menší rezervy vidím v absenci nástrojů, které usnadňují manažerské řízení a absenci správy verzí jednotlivých úkolů.

[12] Přístup do aplikace je možné získat na stránkách výrobce www.easyproject.cz.

5 Návrh vlastní aplikace vzhledem k analýze dostupných nástrojů

Po provedení analýzy dostupných nástrojů bylo vytvořeno zadání tak, aby nově navrhovaná aplikace byla konkurenceschopná a přinášela řešení v oblastech, kde dostupné aplikace nebyly příliš silné.

Požadavky na funkčnost aplikace:

- přístup z jakéhokoli místa pomocí internetu,
- aplikace nezávislá na platformě,
- moduly, umožňující řízení firmy (zakázky, úkoly, výkazy práce, firemní dokumenty, evidence pracovníků, výplatní listiny),
- přehledné grafické statistiky,
- jednoduché a intuitivní ovládání,
- individuální nastavení aplikace,
- podpora manažerských nástrojů (individuální řízení priorit, připomínání úkolů, připomínání událostí),
- podpora zasílání informací o změně úkolu,
- automatické informace o změnách úkolu,
- možnost spravování několika nezávislých databází pomocí jedné aplikace.

Model aplikace

V závislosti na požadavcích bylo nutné zajistit uchovávání velkého množství vzájemně propojených dat a flexibilní práci s nimi. Z tohoto důvodu byla jako datové úložiště využita relační databáze MySQL.

Přístup k databázi mohl být řešen dvěma základními způsoby, a to:

- pomocí webové aplikace,
- pomocí desktop aplikace.

V současné době je velkým trendem aplikace tohoto typu vyvíjet jako webové. Argumenty pro koncept webové aplikace jsou především:

- snadný přístup odkudkoli a nezávisle na platformě,
- není nutná instalace,
- malá zátěž pro klientskou stanici,
- snadná údržba – data na jednom místě.

Zamýšlená aplikace měla za účel využít co nejvíce výhod aplikací webových a navíc připojit i jisté výhody desktop aplikací.

Snadný přístup odkudkoli a nezávisle na platformě

Díky použití multiplatformího programovacího jazyka Java bude možné aplikaci provozovat na jakékoli platformě s nainstalovaným JVM. Databáze dat bude umístěna na vzdáleném serveru. O zajištění přístupu k serveru s databází se postará klientská aplikace pomocí síťových protokolů rozhraní JDBC.

Instalace

Aplikaci nebude nutné instalovat v pravém slova smyslu. Instalace proběhne pouze jednou po prvním spuštění administrátorem. Touto operací dojde ke spuštění instalačního skriptu, který vytvoří novou databázi a administrátorský účet. Ostatní uživatelé pouze nastaví správné údaje o umístění databáze a jejich přístupových účtech.

Zátěž systému

Vzhledem k tomu, že od aplikace nejsou požadovány rozsáhlé výpočty, ani náročná práce s grafikou, nebude se provoz aplikace významně projevovat na zátěži operačního systému dnešních počítačů. Maximum statistických výpočtů obstará databázový server a následně je pouze předá k zobrazení.

Snadná údržba

Data, využívaná pro práci aplikace, budou umístěna v jedné databázi obdobně jako u webové aplikace.

Komfort desktop aplikace

I přes současný trend v oblasti webových aplikací je zřejmé, že větší komfort pro uživatele nabízí desktop aplikace. Pro ilustraci můžeme například porovnat sadu Microsoft Office a nástroj Google dokumenty. Díky tomu, že aplikace běží na klientském počítači, je možné do budoucna implementovat i složité funkce, které bude systém v přijatelném čase schopen spočítat.

Možnost správy více databázových účtů

Aplikace bude fungovat pouze jako univerzální klient (zobrazovač a manažer dat), díky tomu vznikne možnost spravovat pomocí jedné aplikace několik nezávislých databází. Vrcholový manažer tak může řídit dvě firmy pomocí stejné aplikace.

6 Funkčnosti systému

Systém byl navržen tak, aby poskytoval nástroje ke snadnému řízení práce v malé a střední firmě od pracovníků výroby po top management. Velký důraz byl kladen na intuitivnost použití, moderní vzhled a využití nástrojů manažerské metody GTD v praxi, do které je možné zařadit nástroj uživatelského řízení priorit jednotlivých úkolů.

Systém nemá ambici nahradit současné ekonomické softwary (účetnictvím se vůbec nezabývá). Smyslem tohoto systému je sloužit jako doplňkový software pro efektivní řízení a přesnou evidenci práce či nákladů výroby.

6.1 Implementované funkce systému

Základní funkčnost

- Evidence uživatelů (pracovníků).
- Evidence firem (zákazníků).
- Evidence zakázek.
- Evidence úkolů.
- Evidence výkazů práce.
- Generování a tisk výplatních listin (sestava odpracovaných úkolů).
- Evidence firemních dokumentů.
- Přístup uživatelů na základě přihlášení.

Rozšířená funkčnost

- Nastavení grafického vzhledu aplikace.
- Používání nástrojů metody GTD.
    - Připomínání úkolů.
    - Připomínání událostí.
    - Individuální nastavení priorit řešených úkolů.
- Automatická kontrola nově přiřazených úkolů.
- Odesílání informačních emailů o změně úkolu.
- Možnost připojení k více účtům (databázím).
- Možnost uživatelského nastavení minimálních přístupových práv pro jednotlivé moduly.

6.2 Cílová skupina uživatelů systému

Systém je určen pro řízení práce v jakékoli malé a střední firmě, ve které je práce prováděna především pomocí osobních počítačů. Podmínkou úspěšného a efektivního použití aplikace je trvalý přístup pracovníků k internetu. Vzhledem k tomu, že systém funguje na platformě Java, její využití je možné nezávisle na platformě. Díky neustále rostoucí oblibě zařízení typu smartphone je předpoklad, že obory využití se budou dále

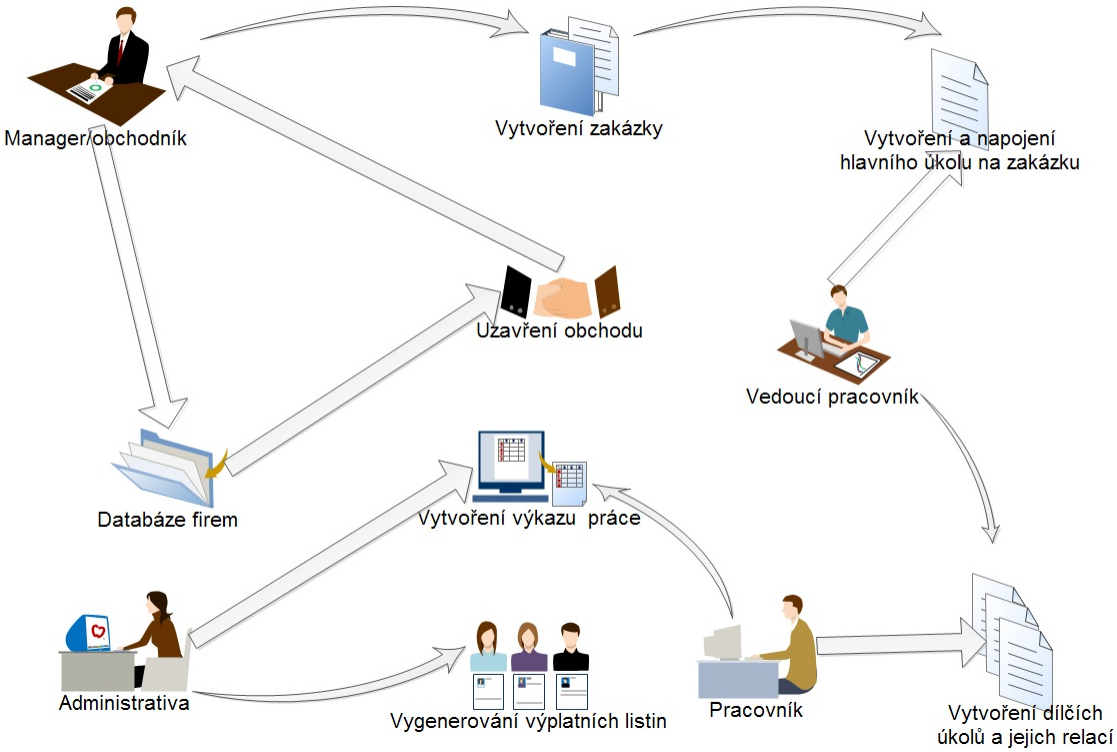
rozšiřovat i mimo kanceláře. Aktuálně je hlavní předpokládané využití pro následující obory:

- vývoj software,
- telemarketingová centra,
- projekční kanceláře,
- obchodní společnosti,
- reklamní a marketingové agentury.

6.3 Rich picture diagram základního použití systému v praxi

Na obrázku 1 je uvedeno předpokládané využití základní funkce systému, která zahrnuje práci s moduly *firmy*, *zakázky*, *úkoly* a *výkazy práce*. Kromě uvedených funkcí systém samozřejmě obsahuje i nadstavbové služby, které slouží pro efektivní dosahování manažerských cílů. Kompletní výčet dostupných funkcí je uveden v následující podkapitole.

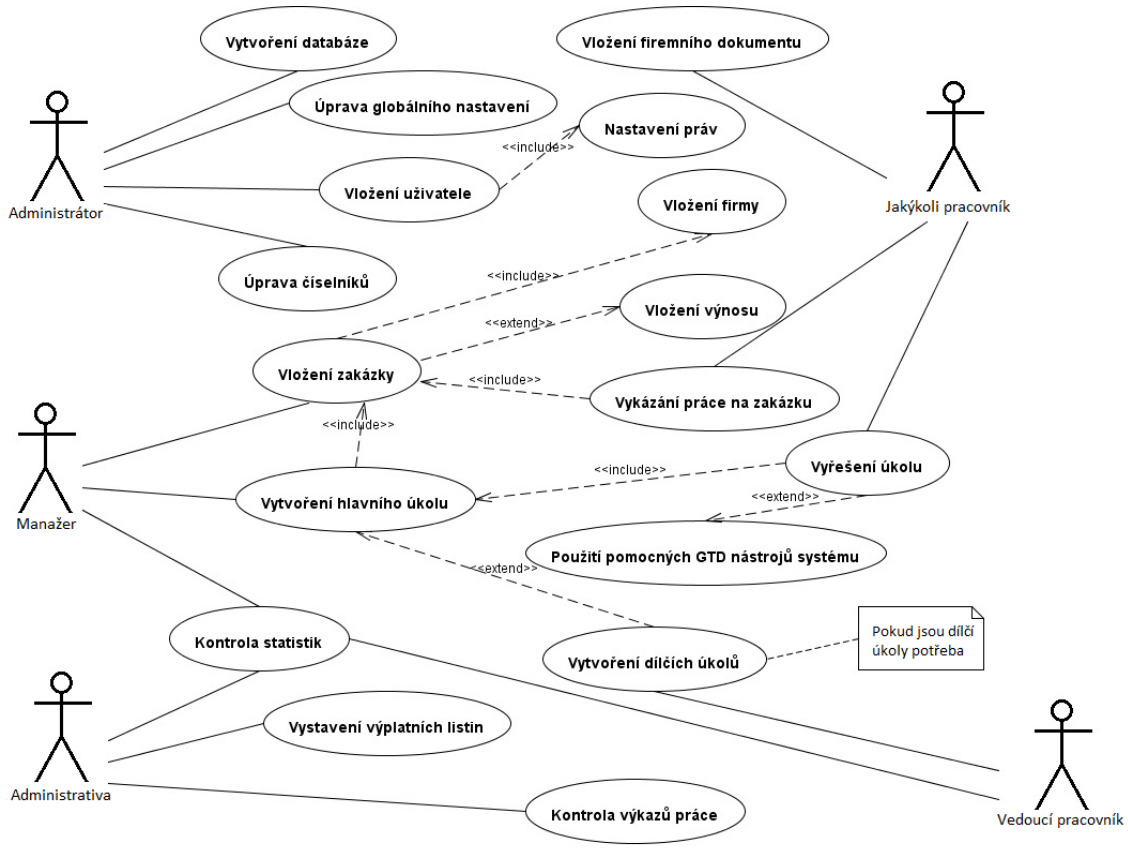
V diagramu jsou znázorněny všechny čtyři uživatelské role (*manažer*, *administrativa*, *vedoucí pracovník*, *pracovník*). Poslední rolí, která je v systému standardně dostupná, je role administrátor. Ta však nebyla navržena pro běžné používání systému, ale pouze pro jeho instalaci a nastavení.



Obrázek 1 - Rich picture diagram základní funkčnosti systému

6.4 UML use case diagram kompletních funkcí systému

UML use case diagram na obrázku 2 zobrazuje kompletní výčet případů užití pro jednotlivé role uživatelů v systému. Tento diagram zobrazuje použití systému tak, jak byl navržen. Vzhledem k tomu, že je u jednotlivých modulů možné měnit minimální oprávnění pro přístup, může být použití aplikace v konkrétním případě mírně odlišné.



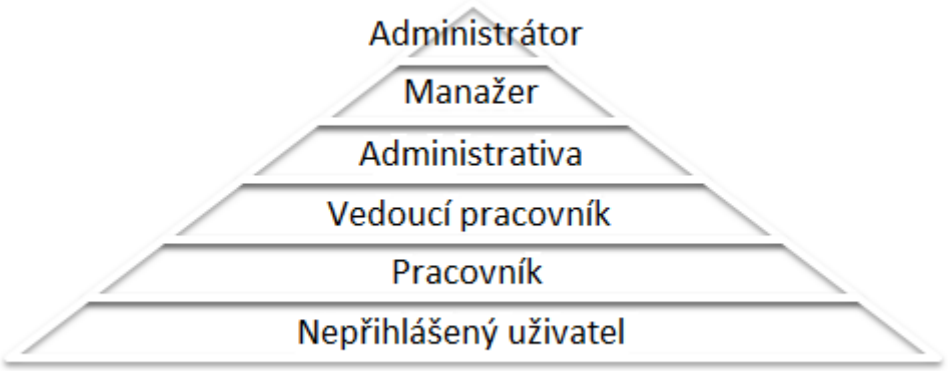
Obrázek 2 - UML use case diagram kompletních funkcí systému

6.5 Oprávnění a řízení přístupu k jednotlivým modulům

Každá aplikace pro řízení práce ve firmě musí nabízet i nástroje pro řízení přístupu k jednotlivým modulům. Vzhledem k tomu, že potřeba nastavení je v každé společnosti individuální, nejsou kompetence systému nastaveny pevně, ale je možné, aby administrátor (případně uživatel, který má potřebné oprávnění) globálně nastavil minimální kompetence pro každý modul.

Pro každý modul je možné v globálním nastavení systému nastavit minimální oprávnění potřebné pro čtení, editaci a vkládání informací. Hierarchie oprávnění, uvedená na následujícím obrázku 3, je pevně daná. Pro rozhodování systému, zda má uživatel dostatečnou kompetenci, je důležitá hodnota proměnné *kompetence_value* uvedená v databázi u každého oprávnění. Vzhledem k tomu, že hodnoty *kompetence_value* od sebe vždy odděluje 10 jednotek, je teoreticky možné doplnit do systému dalších 60 oprávnění. Tento zásah je však možný pouze přímým zásahem přímo databáze. Oprávnění, která jsou

standardně v systému dostupná, by měla bez problému poskytovat dostatečnou variabilitu při správě uživatelů ve firmě.



Obrázek 3 - Hierarchie dostupných oprávnění systému

6.6 UML activity diagram pro uživatelskou roli manažer

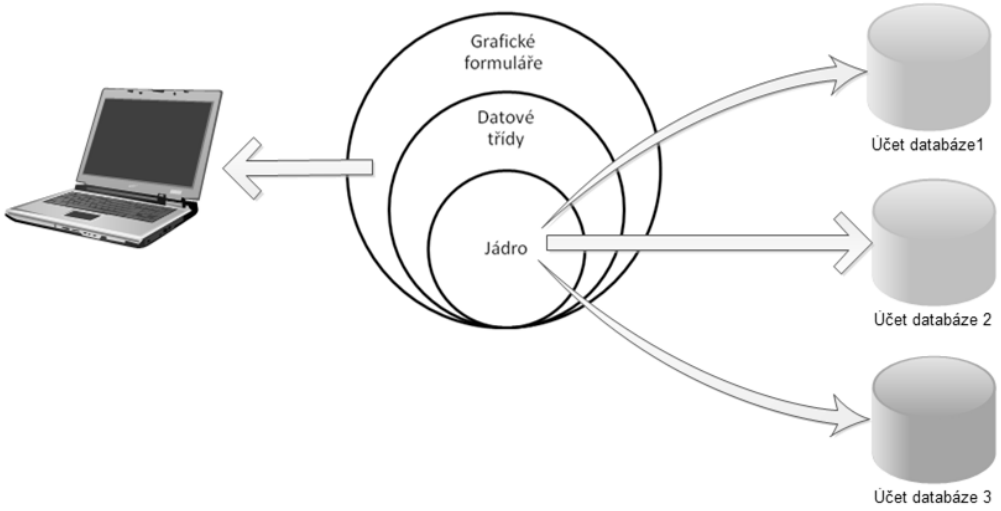
Uživatelská role *manažer* je rolí s nejvyšším oprávněním[13], která je určena pro běžnou (uživatelskou) práci se systémem. Na následujícím obrázku 4 je uveden UML activity diagram pro práci tohoto uživatele v systému.

Obrázek 4 - UML activity diagram role Manažer

[13] Uživatelská role Administrátor by neměla být používána pro běžnou práci se systémem.

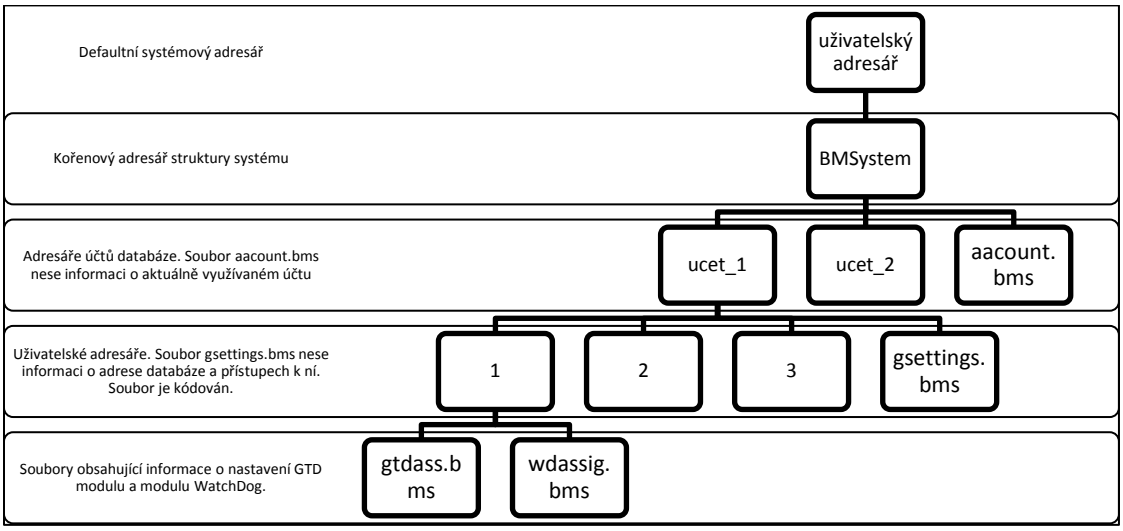
7 Architektura systému

Systém byl vyvíjen dle zásad objektově orientovaného programování a od začátku vývoje zde byla snaha oddělit jednotlivé části systému tak, aby mohly být používány nezávisle na sobě. Jedná se o tři základní části: jádro systému, datové třídy a grafickou nadstavbu v podobě formulářů. Architektura systému je znázorněna na obrázku 5.



Obrázek 5 - Architektura systému

Z předcházejícího obrázku 5 je patrné, že systém dokáže spravovat několik různých databází (účtů). Člen top managementu tedy může pomocí jedné aplikace řídit několik firem jednoduchým přepínáním mezi účty. Pro zajištění této funkčnosti bylo nutné implementovat oddělenou adresářovou strukturu, která je uložena na pevném disku počítače. Struktura adresářů s popisem je uvedena na obrázku 6.



Obrázek 6 - Adresářová struktura systému

8 Použité technologie

Vzhledem k záměru vyvinout multiplatformní desktop aplikaci, která spravuje vzdálenou MySQL databázi, byl pro implementaci zvolen programovací jazyk Java. Pro vývoj byla použita databáze MySQL 5.5.8. V dalších podkapitolách jsou uvedeny podrobnosti k jednotlivým technologiím a použitým knihovnám.

8.1 Programovací jazyk Java

Programovací jazyk Java je objektově orientovaným programovacím jazykem, který byl vyvinut firmou Sun Microsystems. V současné době je Java vlastněna společností Oracle. Jazyk Java se postupem času stal velmi oblíbeným mezi vývojáři. Důvody velké obliby jsou uvedeny níže.

- Přenositelnost – univerzální použití programů psaných v Javě vyjadřuje nejlépe známé heslo „Write once, run anywhere“.
- Bezpečnost a typová kontrola.
- Automatická správa paměti – jedná se o jednu z největších výhod tohoto jazyka, kdy přidělování, respektive uvolňování paměti je řízeno vnitřními nástroji Javy. [4]

Vzhledem k neustálému vývoji tohoto programovacího jazyka se vyčlenilo několik edicí.

- Java2SE (Standard Edition) – standardní edice Javy určená na programování aplikací pro desktop počítače.
- Java2EE (Enterprise Edition) – rozšířená edice určená pro programování servletů a internetových aplikací.
- JSP (Java Server Pages) – platforma umožňuje kombinaci HTML kódu a příkazů javy, které jsou zpracovávány na straně serveru.
- J2ME (Micro Edition) – platforma, která umožňuje vývoj aplikací pro mobilní zařízení. [5]

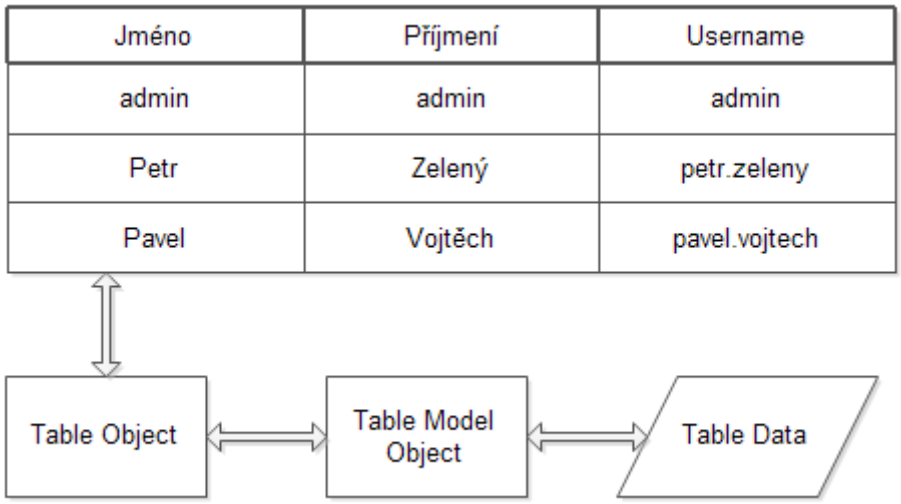
8.2 Swing

Swing je knihovna pro ovládání počítače pomocí grafického rozhraní. Tato knihovna nahradila původní grafickou knihovnu AWT. Vzhledem k tolik zmiňované platformní nezávislosti Javy, využívá Swing takzvaný *Look and Feel manager*, který zajišťuje správné vykreslování vzhledu v závislosti na používané platformě.

Knihovna Swing podporuje takzvané událostmi řízené programování. Veškeré akce jsou tedy řízeny událostmi, které vznikají na základě uživatelského zásahu, nebo jsou vygenerovány jinou součástí aplikace. Swing komponenty zachycují tyto akce pomocí listenerů a následně na ně reagují.

Komponenty ze třídy Swing standardně využívají takzvanou MVC architekturu (model – view – controller). Každá komponenta má tedy svůj datový model, nad kterým je implementováno uživatelské rozhraní. Z tohoto důvodu je modifikace jednotlivých částí

na sobě téměř nezávislá. Na následujícím obrázku 7 je názorně uvedeno použití tohoto modelu u komponenty JTable [6].



Obrázek 7 - Schematické znázornění JComponent [6]

8.3 Java JDBC

Technologie JDBC (Java Database Conectivity) je základem platformy J2EE a slouží pro usnadnění práce s databázemi. Aplikační rozhraní JDBC poskytuje nástroj pro jednotný přístup k databázím. Základem tohoto modelu je využití služeb konkrétního JDBC ovladače. Tento ovladač následně konvertuje požadavky programátora přímo na konkrétní nativní volání dané databáze, se kterou komunikuje [7].

Pomocí API rozhraní JDBC a JDBC ovladače je tedy programátor oddělen od specifického rozhraní databáze a programuje pouze proti jednotnému rozhraní JDBC. Technologie JDBC tedy umožňuje jakýsi univerzální přístup ke všem podporovaným databázím, kterých je v dnešní době již velké množství.

Znázornění JDBC z hlediska programátora je uvedeno na následujícím diagramu (obr. 8). Toto znázornění je velmi zjednodušené, funkce JDBC je poměrně složitá a liší se v závislosti na použití konkrétního ovladače [7].

Obrázek 8 - Schéma funkce JDBC [7]

Samotná práce s rozhraním JDBC je poměrně snadná a intuitivní. Skládá se ze zavolání ovladače JDBC, vytvoření spojení a následného vytvoření objektu

statement. Následně je již možné nad tímto objektem vykonávat konkrétní SQL příkazy. Níže je uveden příklad připojení k databázi a vykonání SQL příkazu. [8]

```
Class.forName("com.mysql.jdbc.Driver");
connection = DriverManager.getConnection(dbURL, account, password);
statement = connection.createStatement();
statement.executeUpdate(sqlQuery);
```

8.3.1 Ochrana proti SQL injection

Obdobně jako u každé jiné databázové aplikace je třeba věnovat pozornost ochraně proti SQL injection. SQL injection je technika, kdy jsou do jednotlivých databázových dotazů, vzniklých na základě odeslání dat z formuláře, úmyslně vkládány další části dotazů za účelem poškození databáze či získání kontroly nad aplikací nebo databází.

Jako ochrana proti SQL injection na straně JDBC slouží takzvaný PreparedStatement. Jedná se o objekt, který v sobě nese předkompilovaný SQL statement. Jednotlivé metody, které umožňují nastavení parametrů PraparedStatement mají silnou typovou kontrolu a neumožňují nastavení proměnné na jinou hodnotu, než je povolena. Níže je uvedeno použití objektu PreparedStatement

```
String sqlQuery = "insert into my table values (?, ?)";
PreparedStatement ps = connection.prepareStatement(sqlQuery);
ps.setInt(1, id);
ps.setString(2,value);
ps.executeUpdate();
```

8.4 Java JFreeChart

JFreeChart je externí knihovna, která se používá ke generování grafů a diagramů. Knihovna je opět napsána v jazyce Java, tudíž není žádný problém s její implementací. Další výhodou je licence LGPL, díky které je knihovna JFreeChart distribuována zdarma.

Knihovna velice usnadňuje programátorovi tvorbu grafů. Zjednodušeně řečeno stačí pouze vytvořit datový model, který bude následně graficky zobrazen pomocí mechanismů JFreeChart. Knihovna umožňuje výběr z velkého množství typů grafů (XY, koláčový, bodový, sloupcový atd.). Dále je možné využívat zvětšování a zmenšování vybraných oblastí grafu, odečítání hodnot z grafu a velké množství dalších funkcí. [9]

8.5 MySQL

Technologie uvedené v předchozích kapitolách byly použity pro tvorbu uživatelské aplikace. Nyní se dostáváme k databázové části, která uchovává většinu dat potřebných pro správnou funkci systému. Tato data jsou následně databází poskytnuta aplikaci, pomocí které jsou uživateli přehledně zobrazena.

MySQL je databázový systém, který aktuálně vlastní společnost Oracle Corporation. Systém je distribuován za určitých okolností pod licencí GPL, tudíž může být zdarma. Databázový systém MySQL je mezi vývojáři a správci velice rozšířen, z tohoto důvodu odpadají problémy se zajištěním vhodného serveru a podobně. Ovládání

databázového systému probíhá prostřednictvím standardizovaného jazyka SQL. Tento typ databáze (konkrétně MySQL 5) byl pro implementaci zvolen z níže uvedených důvodů. [10, str. 14 - 17]

- Databáze MySQL je velice rozšířená a díky své jednoduchosti i oblíbená.
- Předností tohoto systému je vysoká rychlost. MySQL je obecně považováno za jednu z nejrychlejších databází.
- Databáze je v některých případech zdarma.
- Síťová podpora databáze.
- K MySQL je bezproblémový přístup pomocí Javy a JDBC.

8.6 Vývojářské nástroje

Pro vývoj aplikace bylo použito několik vývojových nástrojů. Všechny uvedené nástroje jsou distribuovány zdarma. Popis použitých vývojových nástrojů je uveden v následujících odstavcích.

NetBeans 7.0 Beta

Jedná se o velice podařené prostředí pro tvorbu aplikací v mnoha programovacích jazycích, které je vyvíjeno pod GPL licencí a je sponzorováno společností Oracle Corporation. Přestože byla použita první beta verze této verze, podařilo se v ní aplikaci bez problémů vyvinout.

Oracle Data Modeler 3.0

Tento software je určený pro návrh databázového modelu. Umožňuje grafické znázornění jednotlivých tabulek, vytvoření relací mezi nimi a DDL export vytvořeného modelu.

XAMPP

XAMPP je balík aplikací potřebných pro spuštění webového serveru. Balík obsahuje webový server Apache, který je již konfigurován tak, aby jeho použití pro vývoj bylo co nejsnadnější.

MySQL 5.5.8

Tato verze databáze byla při vývoji nainstalována na lokální počítač a byla spravována prostřednictvím aplikace phpMyAdmin v balíku XAMPP.

9 Datový model

Návrhu datového modulu byla věnována velká pozornost. Vzhledem k tomu, že jednotlivé procesy ve firmě jsou silně provázány s ekonomickou stránkou společnosti, bylo třeba zajistit správné promítání mezi zakázkami, výkazy práce a jinými položkami, které tvoří příjmy, nebo náklady společnosti. ER diagram navrhnuté databáze je z důvodu velikosti umístěn v přílohách (příloha A). V další podkapitole jsou stručně popsány jednotlivé tabulky, které tvoří databázi.

9.1 Popis tabulek

9.1.1 Modul uživatelé

USERS – uchovává informace o uživatelích systému např. jméno, příjmení, datum narození, oprávnění atd. Heslo uživatele je ukládáno hashovaně.

USER_ADDRESS – tabulka adres příslušných uživatelů.

COMPETENCE – číselník jednotlivých uživatelských rolí. Každá uživatelská role má i hodnotu *competence_value*. Na základě této hodnoty je řízen přístup uživatelů k jednotlivým modulům aplikace.

SYSTEM_TABLE_MIN_COMPETENCES – tabulka, která obsahuje nastavení minimálních kompetencí pro čtení, zápis a vložení záznamu do příslušné tabulky.

9.1.2 Modul společnosti

COMPANY – uchovává informace o společnosti např. název, IČO, DIČ, kontaktní osoba apod.

ADDRESS – uchovává adresy společností. Atribut *addres_type_id* určuje, zda se jedná o doručovací adresu, nebo adresu sídla firmy.

ADDRESS_TYPE – číselník typů adres u společností.

9.1.3 Modul zakázky

ORDERS – uchovává informace o založených zakázkách např. název, datum zahájení, datum ukončení, id společnosti, které byla zakázka vystavena apod.

ORDER_CATEGORY – číselník kategorií zakázek.

COST – tabulka všech nákladových položek. Uchovává informace o jednotkové ceně, množství, DPH apod.

COST_TYPE – číselník typ nákladů.

PROFIT – tabulka všech příjmových položek. Uchovává informace o jednotkové ceně, množství, DPH apod.

PROFIT_TYPE – číselník typů příjmů.

UNITS – číselník jednotek používaných u nákladů a příjmů.

9.1.4 Modul úkoly

ASSIGMENT – uchovává údaje o založených úkolech např. název, časovou náročnost, číslo zakázky, datum vyhotovení apod.

NOTES – uchovává všechny poznámky připojené k úkolům.

RELATIONS – uchovává jednotlivé relace mezi úkoly.

RELATIONS_TYPES – číselník typů relací.

PRIORITY – číselník priorit úkolů.

ASSIGMENTS_CATEGORIES – číselník kategorií úkolů.

9.1.5 Modul výkazy práce a výplaty

WORKSHEET – uchovává informace o výkazech práce např. id uživatele, čas zahájení práce, čas ukončení práce, číslo zakázky, číslo úkolu apod.

WORK_TYPE – číselník typů výkazu práce.

WAGES – uchovává informace o vystavených výplatních listinách např. id pracovníka, celkový počet odpracovaných hodin, celkovou hodnotu výplaty apod.

9.1.6 Modul firemní dokumenty

COMPANY_DOCUMENTS – uchovává informace o firemních dokumentech např. název dokumentu, datum vložení, datum platnosti atd.

DOCUMENT_TYPE – číselník typů firemních dokumentů.

10 Architektura aplikace

Aplikace byla od začátku vyvíjena v souladu s pravidly objektově orientovaného programování. Hlavním cílem bylo co nejvíce oddělit data v databázi, datové třídy aplikace a grafické rozhraní aplikace.

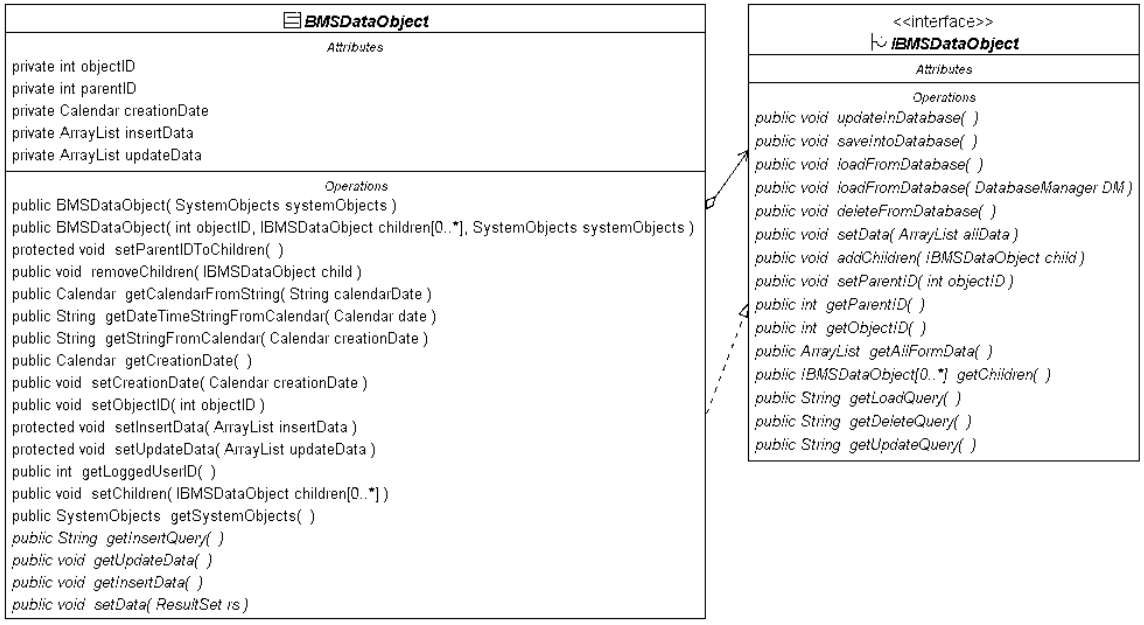
Zjednodušeně řečeno, aplikaci tvoří následující tři části: jádro, datové třídy a grafické formuláře. V dalších podkapitolách jsou uvedeny podrobnosti k těmto jednotlivým částem.

10.1 Jádro aplikace

Jádro aplikace tvoří třídy, které zajišťují základní funkci systému. Jedná se především o abstraktní rodičovské třídy, od kterých jsou zděděny další součásti systému. Dále jde o třídy zajišťující přihlašování a dodržování uživatelských práv. Jsou zde obsaženy třídy, které fungují na pozadí systému a zajišťují služby připomínání úkolů či rozesílání informačních emailů. UML diagram tříd jádra zde z důvodu velikosti není uveden. Diagram je umístěn na přiloženém CD. Níže jsou uvedeny podrobné informace ke třídám, které tvoří jádro systému.

10.1.1 BMSDataObject a rozhraní IBMSDataObject

Tato třída je společným předkem všech datových tříd, které komunikují s databází. Tato třída a všechny její potomci implementují rozhraní *IBMSDataObject*. UML diagram této třídy a rozhraní je uveden na obrázku 9.



Obrázek 9 - UML diagram třídy BMSDataObject

Privátní atributy každé datové třídy je možné rozdělit na několik částí, které jsou níže vyjmenovány.

- Atributy třídy, které jsou ukládány do databáze (jedná se o konkrétní informace o objektu).
- Kontejner potomků datové třídy.
- ID objektu, ID rodiče.

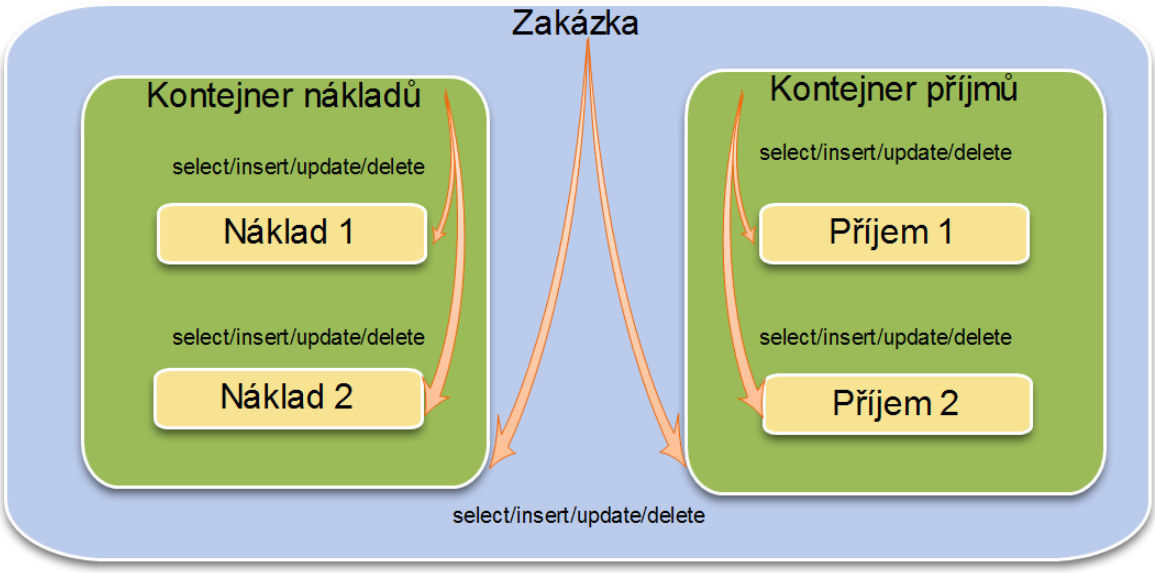
Každá datová třída může mít své potomky, kteří jsou ukládáni do generického kontejneru *ArrayList<IBMSDataObject>*.

Každá datová třída musí umožňovat operace, které zajišťují její komunikaci s databází. Jednotlivé metody jsou popsány v následujících odstavcích.

Metody pro práci s databází

```
public void updateInDatabase() throws SQLException,
ClassNotFoundException;

public void saveIntoDatabase() throws SQLException,
ClassNotFoundException;

public void loadFromDatabase() throws SQLException,
ClassNotFoundException;

public void deleteFromDatabase() throws SQLException,
ClassNotFoundException;
```

Jak již vyplývá z názvu metod, slouží pro načítání, ukládání, úpravu a mazání atributů objektů. Každá datová třída musí zajistit zavolání těchto metod pro všechny své potomky. Metody jsou tedy implementovány pomocí rekurzivního algoritmu. Schematický diagram datových tříd je uveden na obrázku 10.



Obrázek 10 - Rekurzivní úprava tříd v databázi

Konkrétní implementace metody zajišťující update datové třídy a update jejich potomků v databázi je pro ilustraci uvedena níže. Obdobně jsou implementovány i ostatní metody pro *select*, *insert* a *delete*. V metodách se mění pouze konkrétní SQL query databázového dotazu.

```
public void updateInDatabase() throws SQLException,
ClassNotFoundException {
        DatabaseManager DM = new DatabaseManager();
        DM.connect();
        getUpdateData();
        String updateQuery = getUpdateQuery();

        if (updateQuery != null) {
            if (objectID != 0) {
            int i = DM.executeUpdateQuery(updateQuery + objectID,
      updateData);
            } else {
                int i = DM.executeUpdateQuery(updateQuery, updateData);
            }
        }
        if (children != null) {
            if (children.isEmpty()) {
                DM.disconnect();
                return;
            }

            Iterator<IBMSDataObject> it = children.iterator();
            while (it.hasNext()) {
                it.next().updateInDatabase();
            }
        }
        DM.disconnect();
    }
```

Konkrétní SQL query, které bude použito pro operaci s daným objektem, poskytují následující čtyři abstraktní metody, které musí být u všech potomků třídy *BMSDataObject* překryty konkrétní implementací.

```
abstract public String getInsertQuery();

abstract public String getDeleteQuery();

abstract public void getUpdateData();

abstract public void getInsertData();
```

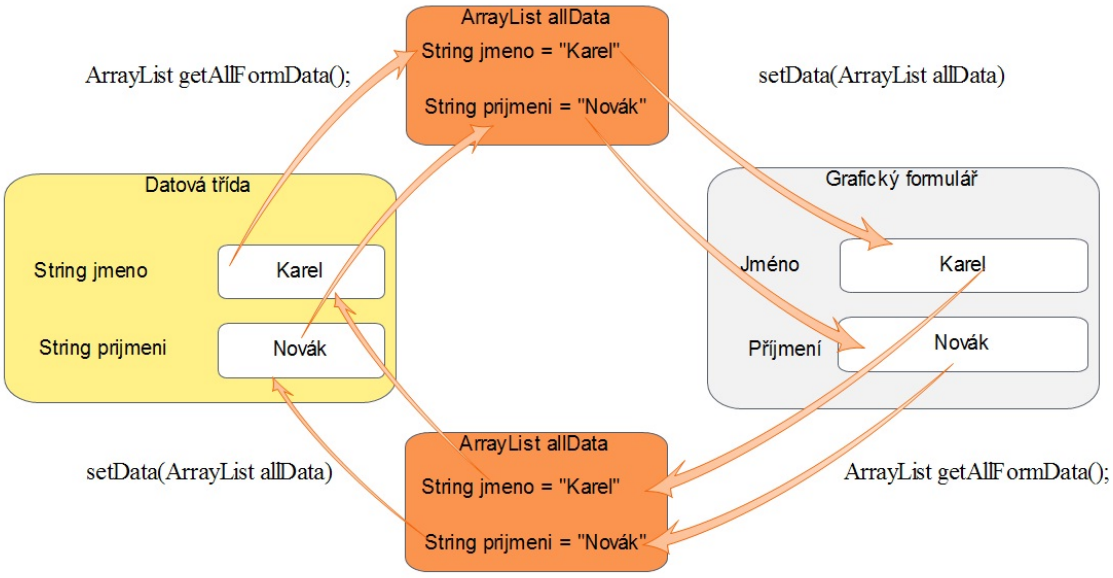
Metody pro komunikaci s grafickými formuláři

Pro předávání dat mezi grafickým formulářem a datovou třídou slouží následující dvě metody.

```
public void setData(ArrayList allData);

public ArrayList getAllFormData();
```

Jedna metoda zpracovává kontejner dat, druhá ho poskytuje. Konkrétní implementace metody daného objektu musí znát pořadí, v jakém jsou data v kontejneru

uložena, a dle tohoto pořadí je nastavuje jednotlivým atributům třídy. Schematické znázornění využití těchto metod je uvedeno na obrázku 11.



Obrázek 11 - Schéma komunikace datových tříd a grafických formulářů

10.1.2 BMSComponent

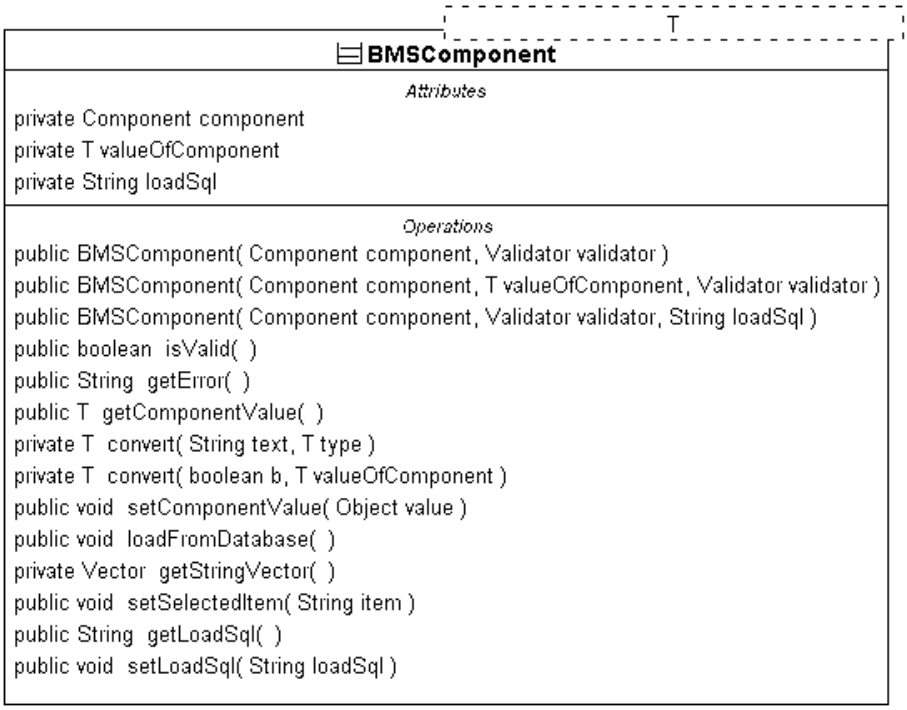
Základním stavebním prvkem použitým ve všech formulářích, které jsou propojeny s datovými třídami, je objekt *BMSComponent*. Všechny tyto komponenty jsou následně zapouzdřeny grafickou třídou *BMSGUIForm*, která je potomkem třídy JPanel.

UML diagram generické třídy *BMSComponent* je uveden na obrázku 12. Třída *BMSComponent* zobecňuje použití Swing komponent dostupných v knihovně. Především zajišťuje konverzi mezi typem String, který je získán z pole formuláře pomocí metody getText(), na konkrétní požadovaný datový typ a opačně. Dále samozřejmě zajišťuje sjednocení nastavení hodnoty pro jakoukoli komponentu (JRadioButton, JTextField, JComboBox). Třída rovněž dokáže načíst model komponenty *JComboBox* podle předem určeného SQL query. Tato funkce je hojně využívána u číselníků, které nabízejí pouze hodnoty obsažené v databázi.

Další služba třídy *BMSComponent* je validování údajů zadaných do jednotlivých polí. Pokud je u pole formuláře vyžadována validace zadaného údaje, stačí dané komponentě nastavit příslušný objekt validátoru, který se automaticky postará o kontrolu a v případě, že se zadaná data neshodují, vygeneruje chybové hlášení. K této funkci je používáno služeb třídy *Validator*. *Validátor* umožňuje níže uvedené nastavení.

- Povinný/nepovinný údaj.
- Nutná shoda s regulárním výrazem.
- Kontrola, zda vyplněný údaj, který se stane cizím klíčem tabulky v databázi, je obsažen jako primární klíč v jiné tabulce.

Shoda hodnoty pole s regulárním výrazem je řešena pomocí enum třídy, která nabízí předdefinované regulární výrazy pro *e-mail*, *číslo*, *text* a *datum*.



Obrázek 12 - UML diagram třídy BMSComponent

10.1.3 BMSGUIForm a rozhraní IBMSGUIForm

Třída *BMSGUIForm* je předkem všech grafických formulářů, které následně poskytují, nebo zobrazují informace datových tříd. Díky použití jednotného předka všech grafických formulářů došlo k usnadnění celé implementace, jelikož velká část operací je pro všechny grafické formuláře shodná. Každý grafický formulář musí implementovat rozhraní *IBMSGUIForm*, které sjednocuje veřejné metody dostupné pro tento typ objektů. UML diagram této třídy a jejího rozhraní je uveden v přílohách (příloha B).

Privátní atributy každé třídy grafického formuláře je možné rozdělit na několik částí uvedených níže.

- Atributy jednotlivých grafických komponent formuláře, které tvoří základní komponenty knihovny Swing.
- Kontejner potomků třídy grafického formuláře, který obsahuje potomky třídy BMSGUIForm implementující rozhraní IBMSGUIForm.
- Privátní řetězec obsahující chybové hlášení, který je generován třídami validátoru v závislosti na nastavení konkrétních BMSComponent.
- Objekt zajišťující přístup k akcím v závislosti na nastavených uživatelských právech.

Každá třída grafického formuláře může mít své potomky (obdobně jako u třídy datové), kteří jsou ukládáni do generického kontejneru `ArrayList<IBMSGUIForm>`.

Každá třída grafického formuláře musí umožňovat následující operace:

```
public IBMSDataObject makeSourceObject();

public void updateSourceObject();

public void reload();

public void saveOrUpdate();
```

Metoda *makeSourceObject()* je zřejmě nejdůležitější metodou. Tato metoda zajistí rekurzivní vytvoření konkrétní datové třídy a všech jejich potomků. Metoda tedy musí být u všech potomků třídy *BMSGUIForm* překryta vlastní implementací. Po stisknutí tlačítka *Uložit* je tato metoda rekurzivně zavolána. Výsledkem práce metody je kompletní datový objekt třídy, který je následně uložen do databáze. Uložení opět proběhne rekurzivně pro všechny potomky. Níže je uveden zdrojový kód této metody.

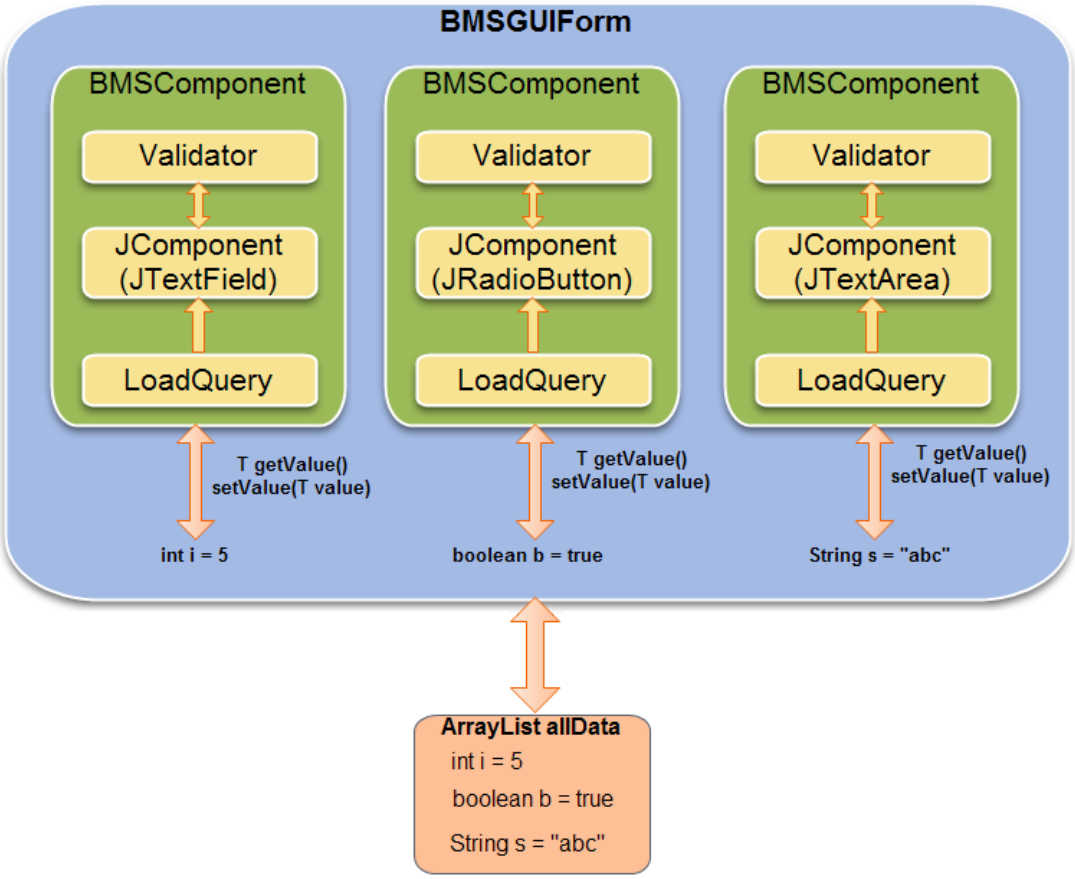
```
public IBMSDataObject makeSourceObject() {
        sourceObject = makeSourceObject(getAllFormData());

        if (children == null) {
            return sourceObject;
        } else if (children.isEmpty()) {
            return sourceObject;
        }

        Iterator<IBMSGUIForm> it = children.iterator();
        while (it.hasNext()) {
            IBMSGUIForm child = it.next();
            IBMSDataObject dataObject = child.makeSourceObject();
            if (dataObject != null) {
                sourceObject.addChildren(dataObject);
            }
        }

        return sourceObject;
    }
```

Metoda *updateSourceObject()* je volána v případě, že je upravován již existující formulář. Po stisknutí tlačítka *Upravit* nastaví tato metoda správné hodnoty privátních atributů zdrojového objektu. Následně je nad zdrojovým objektem zavolána metoda *updateInDatabase()*.

Metoda *reload()* zajišťuje opětovné načtení údajů formuláře z databáze. Je použita například po uložení objektu k získání data vložení formuláře. Metoda opět musí zajistit rekurzivní načtení všech svých potomků.

Metoda *saveOrUpdate()* volá v závislosti na stisknutém tlačítku příslušnou metodu napojené datové třídy (*saveInDatabase()*, *updateInDatabase()*). Tato metoda je implementována proto, že může nastat situace, kdy rodičovská třída obsahuje několik potomků, z nichž je část již uložena a část teprve uložena bude. Metoda se tedy u každého potomka sama rozhodne, zda je nutné ho uložit, nebo upravit.

Na obrázku 13 je zjednodušeně znázorněna komunikace mezi grafickým formulářem a jeho komponentami. V praxi jsou jednotlivé komponenty uloženy v kontejneru *ArrayList<BMSComponent> allComponents*. Nad tímto kontejnerem jsou následně prováděny akce typu save, update a reload.



Obrázek 13 - Znázornění funkce grafického formuláře

10.1.4 BMSTableModel, BMSTablePanel a rozhraní IBMSTablePanel

Třída *BMSTableModel* je potomkem abstraktní třídy *AbstractTableModel*, kterou rozšiřuje o metody popsané v následujících odstavcích.

`public BMSTableModel(ResultSet tableData, int countOfRows)` – tento konstruktor zajistí vytvoření tabulky (dvojrozměrného pole dat) na základě výsledku databázového dotazu předaného ve formě objektu ResultSet.

`public void addColumn(String columnName, int position)` – metoda zajišťuje přidání nového sloupce tabulky na zadanou pozici.

`public void addColumnData(int column, Object value)` – metoda umožňuje naplnit nově přidaný sloupce libovolnou hodnotou dat.

Všechny další třídy tabulek jsou vytvořeny nad datovým modelem *BMSTableModel* a využívají jeho služeb. Níže je pro ukázku uveden konstruktor metody a metoda *addColumnData()*.

Konstruktor třídy BMSTableModel

```
public BMSTableModel(ResultSet tableData, int countOfRows) throws
SQLException, ClassNotFoundException {
        editableCells = new ArrayList<Integer>();
        int rowIndex = 0;
        ResultSetMetaData rsmd;
        rsmd = (ResultSetMetaData) tableData.getMetaData();
        this.columnNames = new String[rsmd.getColumnCount()];
        this.data = new Object[countOfRows][rsmd.getColumnCount()];
        while (tableData.next()) {
            for (int i = 0; i < rsmd.getColumnCount(); i++) {
                this.data[rowIndex][i] = tableData.getObject(i + 1);
            }
            rowIndex++;
        }
    }
```

Metoda addColumnData

```
public void addColumnData(int column, Object value) {
        Object[][] newData = new Object[getRowCount()][getColumnCount()];
        boolean paste = false;
        for (int i = 0; i < getRowCount(); i++) {
            paste = false;
            for (int j = 0; j < getColumnCount() - 1 + 1; j++) {
                if (j == column) {
                    newData[i][j] = value;
                    paste = true;
                } else if (paste) {
                    newData[i][j] = this.data[i][j - 1];
                } else {
                    newData[i][j] = this.data[i][j];
                }
            }
        }
        this.setData(newData);
    }
```

Třídy *BMSTableModel* a *BMSTablePanel* slouží pro univerzální zobrazování tabulkových dat. UML diagram těchto tříd a jejich rozhraní je uveden v přílohách (příloha C). Všechny panely tabulek musí implementovat rozhraní *IBMSTablePanel* a následující metody.

```
public void setTableModel();

public void loadTable()

protected String getLoadQuery()

public void reloadTableWithConstraint(String filterConstraint,
ArrayList constraintContainer, boolean changeLimits);

public void setColumnNames(String columnNames[]);
```

Metoda *public void setTableModel()* umožňuje napojení nového datového modelu.

Metoda *public void loadTable()* provede SQL query, které bude vráceno překrytou metodou *getLoadQuery()* potomka třídy *BMSTablePanel* a výsledný *ResultSet* zobrazí jako tabulku.

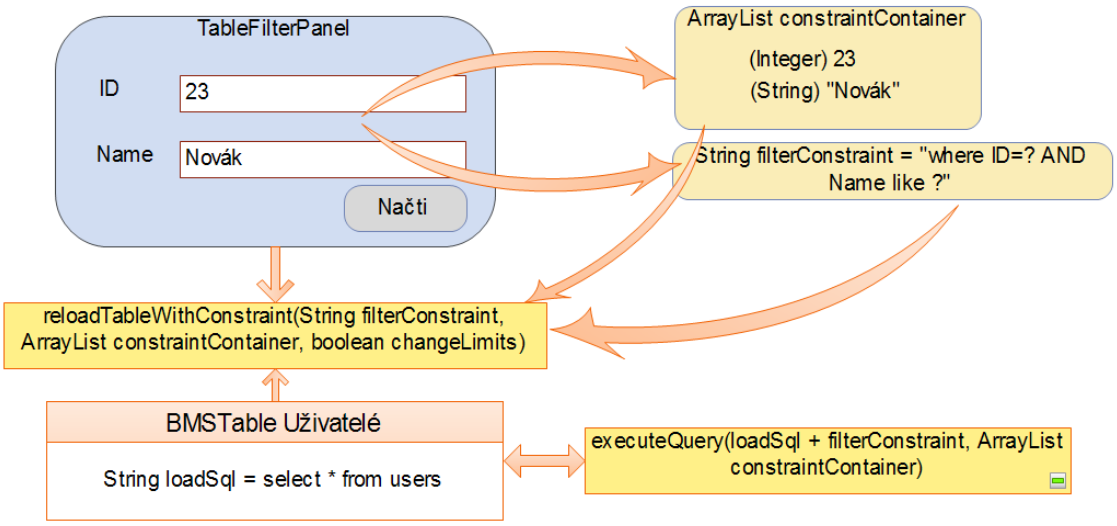
Konkrétní metoda *getLoadQuery()* s jedním ze složitějších databázových dotazů je uvedena níže. Jedná se o databázový dotaz, který zajistí načtení tabulky uživatelů.

```
@Override
protected String getLoadQuery() {
String sql = " select * from (select user_id,
first_name,surname,user_name, email, activ, competence.competence_name
from users left join competence on users.competence_id =
competence.competence_id where users.competence_id in (select
competence_id from competence where competence_value <= (select
competence_value from users left join competence on users.competence_id =
competence.competence_id where user_id="+
super.getSystemObjects().getLoginManager().getLoggedUserID() + "))"
+ " union
(select user_id, first_name,surname, user_name, email, activ,
competence.competence_name from users left join competence on
users.competence_id = competence.competence_id where users.user_id=" +
super.getSystemObjects().getLoginManager().getLoggedUserID() +
")) as myTab";

return sql;
    }
```

public void setColumnNames(String columnNames[]) – tato metoda nastaví názvy sloupců v hlavičce tabulky dle řetězců uvedených v poli, které je předáno jako argument funkce.

public void reloadTableWithConstraint(String filterConstraint, ArrayList constraintContainer, boolean changeLimits) – tato metoda je volána panelem filtru tabulky. V případě, že do filtru byla zadána některá omezení, bude k SQL query připojena klauzule *where* a výsledek nového dotazu bude opět zobrazen. Pokud ve filtru nejsou žádná omezení, bude zavolána metoda *loadTable()* a tímto způsobem znovu načtena aktuální data tabulky.

Schéma funkčnosti filtru tabulky a metody *reloadTableWithConstraint(String filterConstraint, ArrayList constraintContainer, boolean changeLimits)* je uvedeno na obrázku 14.



Obrázek 14 - Znázornění funkce tabulek a jejich filtrů

public void setTableFilter(IBMSTableFilter tableFilter) – metoda umožňuje tabulce připojit panel s konkrétním filtrem, který implementuje jednotné rozhraní *IBMSTAbleFilter*.

public void setColumnNames (String columnNames[ ]) – metoda nastaví nová jména sloupců do hlavičky tabulky.

public void makeTableFromArray (IBMSWatchObject[ ] [ ]) – metoda vytvoří tabulku na základě dat předaných ve dvojrozměrném poli.

public void reloadTableFromArray (IBMSWatchObject[ ] [ ]) – metoda provede aktualizaci dat tabulky na základě pole nových hodnot.

Vzhledem k tomu, že se v tabulkách objevují tlačítka, check boxy a combo boxy, které standardně komponenta JTable nepodporuje, bylo třeba vytvořit následující třídy, které zajišťují správné vykreslování těchto komponent do buňky tabulky.

ButtonEditor.java – třída umožňuje editovat buňku pomocí kliknutí tlačítka.

ButtonRenderer.java – třída zajišťuje vykreslení buňky jako tlačítka.

CheckBoxEditor.java – třída vykreslí buňku tabulky jako JComboBox.

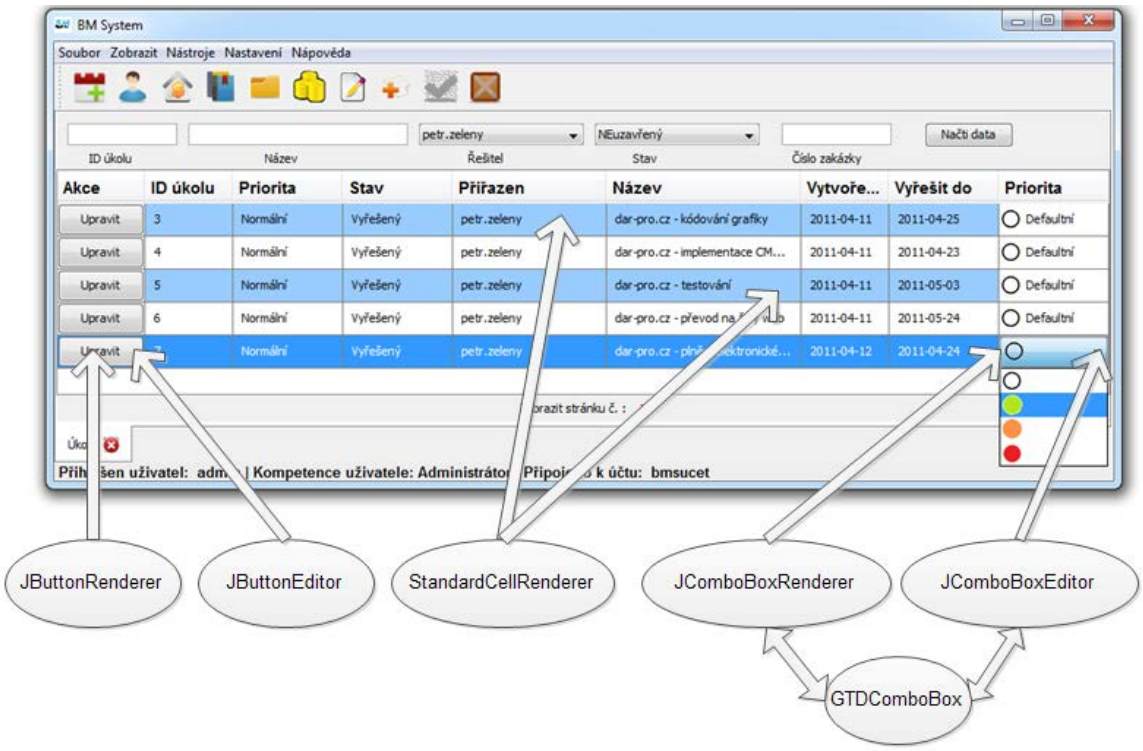
CheckBoxRenderer.java – třída vykreslí buňku tabulky jako check box.

ComboBoxEditor.java – třída umožní editaci buňky pomocí JComboBoxu.

GTDComboBox.java – speciálně upravená komponenta JComboBox, která umožňuje zobrazení a výběr obrázků. Komponenta je použita pro uživatelské nastavení priorit v modulu *úkoly* a *úkoly ke zpracování*.

StandardCellRenderer.java – tato třída zajišťuje základní formát tabulek. Na každou buňku tabulky je při vytvoření aplikováno pravidlo popsané ve třídě *StandardCellRenderer*. Jedná se například o podbarvení řádku, zarovnání textu, odsazení a podobně.

Použití výše uvedených tříd je znázorněno na obrázku 15.



Obrázek 15 - Využití tříd typu "Renderer"

10.1.5 DatabaseManager

Třída *DatabaseManager* zajišťuje navazování spojení a veškerou komunikaci s databází. Pro samotnou komunikaci s databází je využíváno rozhraní JDBC, které s MySQL databází komunikuje pomocí ovladače `mysql-connector-java-5.1.15-bin`. Třída umožňuje vytvoření *PreparedStatement* ze zadaných parametrů, vykonání konkrétních SQL query a vykonání SQL skriptů.

Na obrázku 16 je uveden UML diagram třídy DatabaseManager a dále jsou zde stručně popsány důležité metody této třídy.

Obrázek 16 - UML diagram třídy DatabaseManager

`public int executeUpdateQuery(String query, ArrayList parameters)` – metoda slouží pro provádění SQL příkazů, které nevrací žádný výsledek (např. insert a update). Návratovou hodnotou je počet řádků tabulky, kterých se příkaz týkal.

`public String executeSqlScript(InputStream resourceAsStream)` – metoda provede SQL skript, který je předán ve formě vstupního proudu ze souboru. Návratovou hodnotou je seznam chyb, které nastaly při zpracování skriptu.

`public ResultSet executeQuery(String query, ArrayList parameters)` – metoda slouží pro provádění SQL příkazů, které vrací nějaký výsledek (příkazy typu select). Návratovou hodnotou metody je objekt ResultSet, který obsahuje vybrané řádky tabulky.

Pro ilustraci je níže uvedena privátní metoda *`private PreparedStatement prepareStatmentFromQuery`*, která zajistí zpracování řetězce databázového dotazu, do kterého doplní parametry dotazu předané v kontejneru *parameters*. Objekt *`PreparedStatment`* je využíván jako ochrana proti SQL injection.

```
private PreparedStatement prepareStatmentFromQuery(String query,
ArrayList parameters) throws SQLException {
        PreparedStatement ps;
        Object parameter;
        int index = 0;
        ps = connection.prepareStatement(query);
        Iterator it;
        if (parameters == null) {
            return ps;
        }
        it = parameters.iterator();
        while (it.hasNext()) {
            index++;
            parameter = it.next();
            if (parameter == null) {
                ps.setNull(index, 1);
            } else if (parameter instanceof Integer) {
                ps.setInt(index, (Integer) parameter);
            } else if (parameter instanceof Double) {
                ps.setDouble(index, (Double) parameter);
            } else if (parameter instanceof String) {
                ps.setString(index, (String) parameter);
            } else if (parameter instanceof InputStream) {
                ps.setBinaryStream(index, (InputStream) parameter);
            } else if (parameter instanceof Object) {
                ps.setObject(index, parameter);
            }

        }
        return ps;
    }
```

10.2 Ostatní balíčky a třídy aplikace

V předchozí kapitole bylo stručně popsáno jádro aplikace, které je z velké části tvořeno univerzálními (často abstraktními) metodami předků, od kterých jsou následně tvořeny již konkrétní implementace potomků.

Pro každý modul jsou ve zdrojových kódech vytvořeny tři samostatné balíčky například assigments, *gui.assigments* a *gui.assigments.resources*. V balíčku *assigments* jsou umístěny datové třídy modulu týkajícího se úkolů, v balíčku *gui.assigments* jsou umístěny třídy, které se starají o konkrétní vykreslování grafické podoby dat. Poslední balíček *gui.assigments.resources* obsahuje zdrojové informace ke komponentám, které tvoří grafiku. Kompletní seznam balíčků je z důvodu velikosti uveden v přílohách (příloha D).

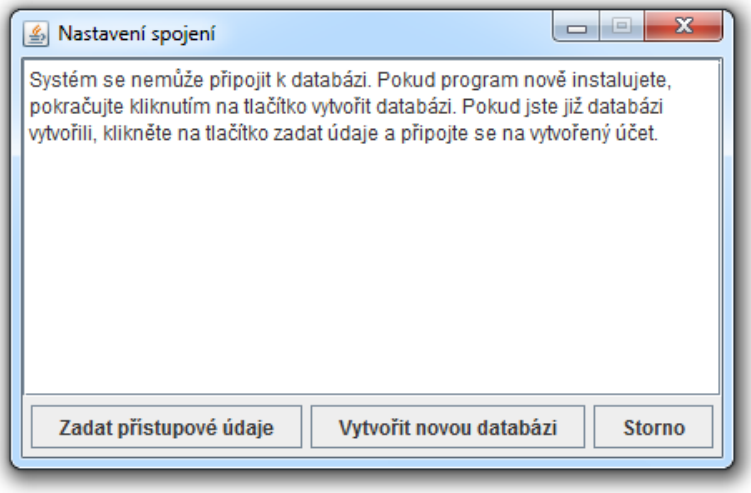
Vzhledem k tomu, že celý program je tvořen přibližně 150 třídami, nebudou zde všechny popisovány. Popis jednotlivých tříd je možné získat z dokumentace přiložené

na CD. Většina tříd, které zde nebyly popsány, jsou potomky jedné z rodičovských tříd uvedených výše. Jejich implementace se tedy liší pouze v konkrétních parametrech pro daný modul.

11 Instalace aplikace

Vzhledem k tomu, že aplikace potřebuje pro svoji funkci správně vytvořenou MySQL databázi a rovněž konkrétní adresářovou strukturu na pevném disku uživatele, je třeba před prvním použitím provést krátkou instalaci. Instalaci provádí osoba, která bude působit jako administrátor aplikace (během instalace bude vytvořen i administrátorský účet).

Po spuštění program prohledá uživatelské adresáře a pokusí se nalézt soubor obsahující informace o umístění databáze. Pokud není tento soubor nalezen, nepodaří se připojit databázi a uživateli je zobrazeno následující okno s popisem této situace. Informační okno je uvedeno na obrázku 17.



Obrázek 17 - Informace o ztrátě připojení

V této chvíli má uživatel dvě možnosti. Pokud ví, že je databáze již vytvořená a systém pouze z nějakého důvodu databázi nedokázal připojit (možná změna hesla účtu databáze, nebo porušení adresářové struktury na disku), stačí zadat adresu databáze a přístupové údaje k jejímu připojení (tlačítko *Zadat přístupové údaje*). Pokud je program instalován poprvé, je třeba databázi celou vytvořit (tlačítko *Vytvořit novou databázi*).

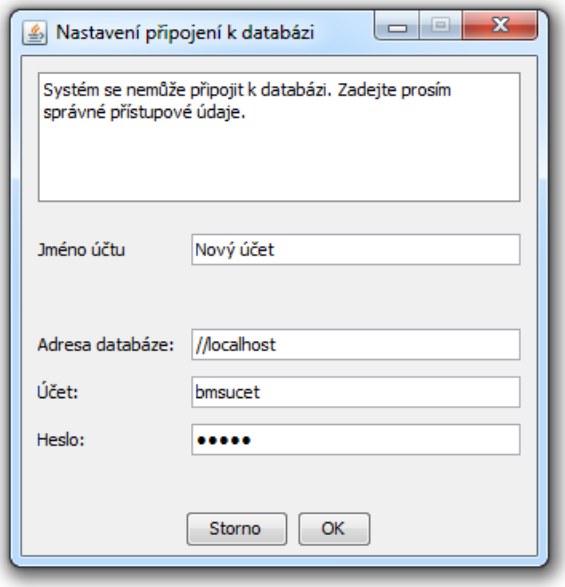
11.1 Zadání přístupových údajů

Po kliknutí na tlačítko *Zadat přístupové údaje* se zobrazí okno uvedené na dalším obrázku (obr. 18). Vzhledem k tomu, že databáze i přístupový účet jsou již vytvořeny, stačí systému tyto informace nastavit.

Jméno účtu – identifikující název účtu, ke kterému se bude uživatel přihlašovat. V přihlašovacím okně se tento název objeví ve výběru účtů dostupných pro přihlášení.

Adresa databáze – konkrétní adresa, na které je databáze dostupná.

Účet – jméno účtu, který byl vytvořen instalátorem pro přístup programu do databáze s patřičným oprávněním. Nedoporučuje se zadávat administrátorský účet se všemi právy.



Obrázek 18 - Nastavení spojení s databází

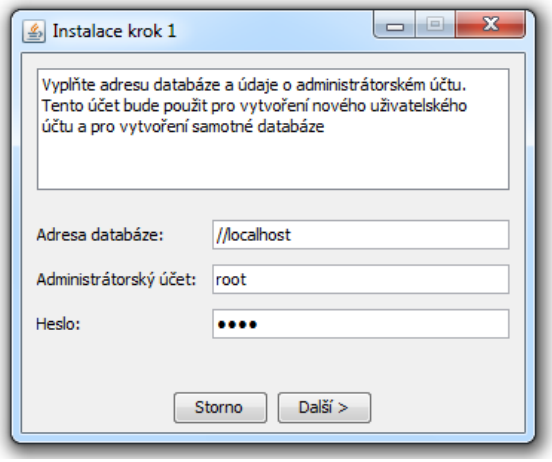
Po kliknutí na tlačítko *OK* systém otestuje, zda se připojí k databázi a v případě, že ano, bude vytvořena adresářová struktura na disku. Následně je program automaticky spuštěn. V případě, že se k databázi nepodaří připojit, je opět uživatel vyzván k zadání správných údajů (obr. 17).

11.2 Nová instalace

Pokud je aplikace instalována poprvé, je nutné vytvořit pomocí instalátoru novou databázi. Po kliknutí na tlačítko *Vytvořit novou databázi* bude spuštěna instalace databáze. Uživatel je vyzván k zadání administrátorského přístupu k databázi. Okno s tímto nastavením je uvedeno na obrázku 19.

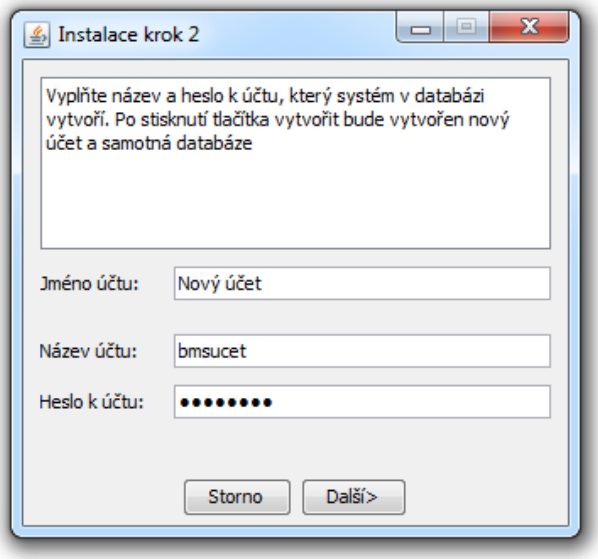
Zadaný administrátorský účet musí mít přidělená práva pro vytváření tabulek a uživatelských účtů. Prostřednictvím tohoto administrátorského účtu dojde k vytvoření nové databáze s názvem *bms*. Instalační skript dále vytvoří všechny potřebné tabulky a přístupový účet.

Popisovaná instalace se týká pouze prvního použití administrátorem. Ostatní uživatelé již nemusí aplikaci instalovat. V jejich případě stačí pouze nastavit správné přístupové údaje k databázi (kapitola 11.1).



Obrázek 19 - Instalace krok 1

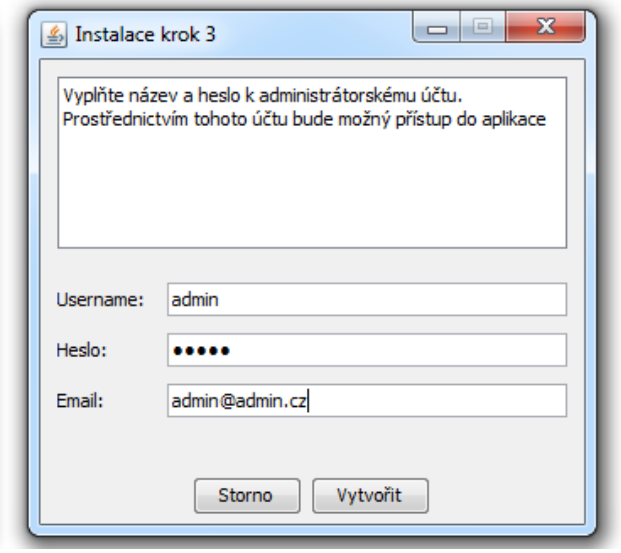
Po stisku tlačítka *Další* systém otestuje, zda se dokáže k databázi připojit. Pokud je připojení v pořádku, následuje další krok instalace, kde je třeba vyplnit údaje o uživatelském účtu (doporučuje se nepoužívat diakritiku), který bude nově vytvořen pro přístup programu k databázi. Tento účet bude na rozdíl od účtu administrátorského využíván pro veškerou práci s databází. Uvedený název účtu a heslo je třeba si zapamatovat. Druhý krok instalace je uveden níže na obrázku 20.



Obrázek 20 - Instalace krok 2

V posledním kroku (obr. 21) je třeba vyplnit přihlašovací údaje administrátorského účtu aplikace (slouží pro přihlášení administrátora do systému). Prostřednictvím tohoto účtu se bude možno přihlásit do aplikace a založit další uživatele. Administrátorský účet by

neměl být používán pro běžnou práci se systémem, ale pouze pro vytváření nových uživatelů, úpravu globálního nastavení a modifikaci číselníků.



Obrázek 21 - Instalace krok 3

Po kliknutí na tlačítko *Vytvořit* bude spuštěn instalátor, který zpracuje SQL skript a vytvoří novou databázi s názvem *bms*. Dále v databázi vytvoří nový přístupový účet s příslušnými právy pro tuto databázi.

Pokud instalace proběhne v pořádku, je instalátor ukončen a spuštěna aplikace vyžadující přihlášení pomocí zadaného účtu z kroku 3.

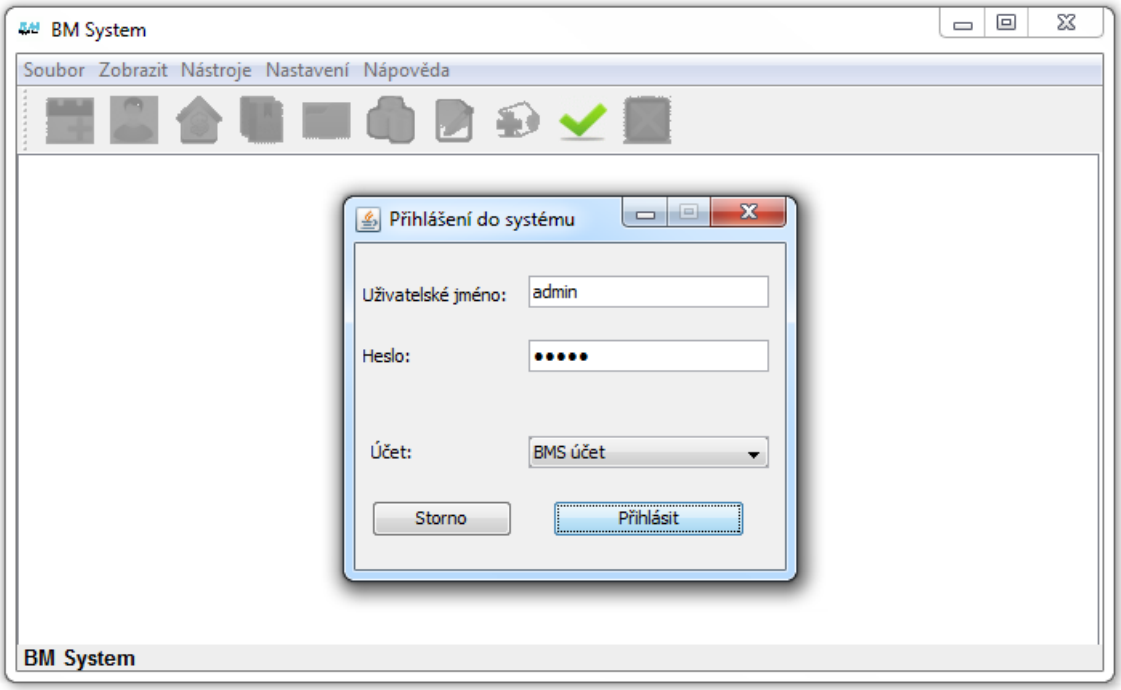
Pokud se instalace z nějakého důvodu nezdaří, budou změny odrolovány a databáze tudíž nebude vytvořena.

12 Uživatelská příručka

12.1 Přihlášení

Pro veškeré operace s programem je vyžadováno přihlášení do systému. Po spuštění aplikace je tedy uživatel automaticky vyzván k zadání přihlašovacích údajů. Přihlašovací okno je rovněž možné vyvolat kliknutím na ikonu zelené „fajfky“, nebo pomocí klávesové zkratky Ctrl + L.

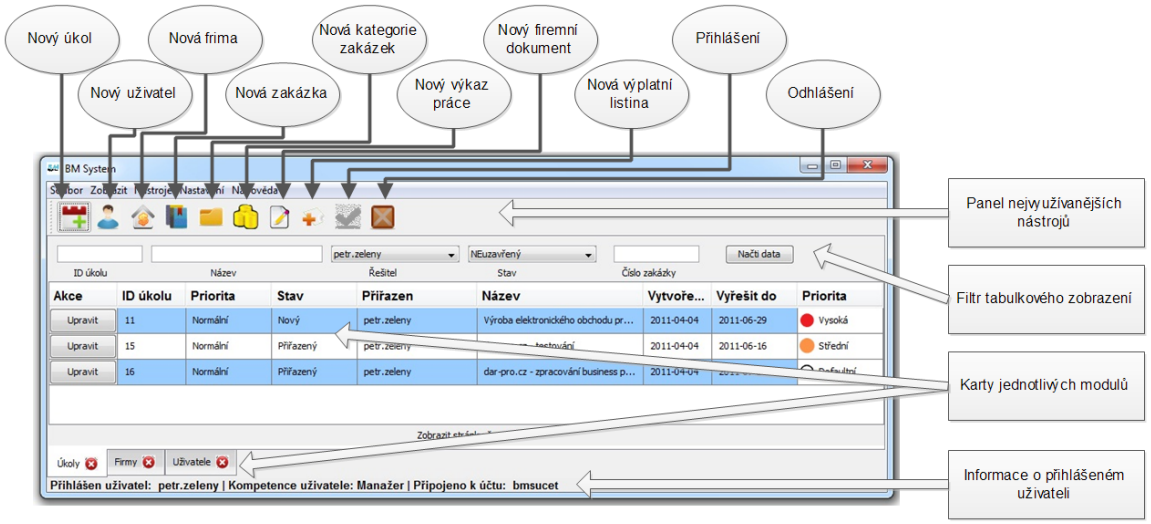
Pomocí roletkového výběru účtů, si uživatel může zvolit konkrétní účet databáze, ke kterému se chce přihlásit. Po stisknutí tlačítka *Přihlásit* porovná systém hodnoty hashe hesla (hesla jsou v databázi uložena hashovaně) a v případě shody uživatele přihlásí. Přihlašovací obrazovka je uvedena na obrázku 22.



Obrázek 22 - Úvodní okno aplikace s přihlášením

Po přihlášení se ve status baru okna (řádek v zápatí) zobrazí informace o přihlášeném uživateli, jeho oprávnění a účtu databáze, ke kterému se uživatel připojil. V závislosti na oprávnění daného uživatele a na nastavení uživatelských práv v systému dojde ke zviditelnění jednotlivých ikon ovládacího panelu a položek menu.

Pokud se uživatel úspěšně přihlásil, je již možné se systémem plnohodnotně pracovat. Popis jednotlivých částí programu je uveden níže na obrázku 23. Pokud se přihlášení z nějakého důvodu nezdařilo (chybné heslo, nebo uživatelské jméno), je uživateli tato informace zobrazena přímo v přihlašovacím okně.



Obrázek 23 - Základní popis aplikace

12.2 Nastavení programu

Program umožňuje velké množství nastavení. Nastavení je možné rozdělit na globální a lokální. *Globální nastavení* zahrnuje soubor pravidel, která jsou společná pro všechny uživatele programu. *Globální nastavení* administruje administrátor a ostatní uživatelé si toto nastavení stahují z databáze.

Lokální nastavení umožňuje každému uživateli individuální nastavení vzhledu a *watch dogu*. Pokud uživatel nevyplní své lokální nastavení, je nastavení přebráno z globálního nastavení programu.

12.2.1 Globální nastavení

Okno globálního nastavení je dostupné z menu *Nastavení* → *Vlastnosti* →*Globální nastavení* (obr. 24). Níže je uveden popis jednotlivých karet nastavení.

Připojení – tato karta umožňuje upravit informace týkající se připojení programu k databázi. Nastavení je možné využít v případě, že administrátor bude chtít ručně (prostřednictvím vlastní administrace databáze) změnit uživatelský účet pro přístup k databázi. Po připojení uživatelů dojde ke stažení aktuálního nastavení, a to bude dále používáno.

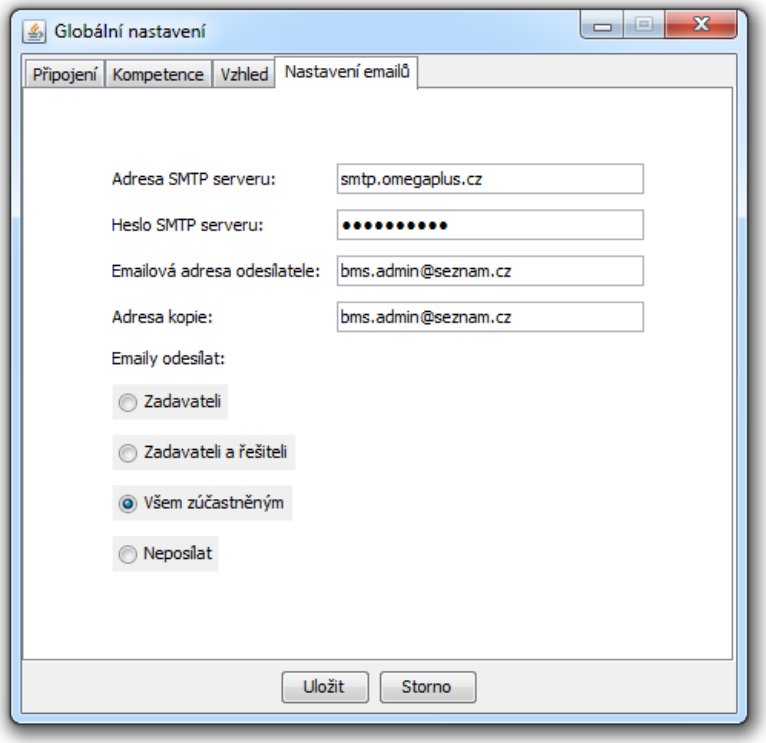
Kompetence – na této kartě je možné pro každý modul nastavit minimální hodnotu oprávnění (čtení, zápis, změna). Pokud tedy administrátor nastaví u modulu *Úkoly* minimální oprávnění pro zápis na *Vedoucí pracovník*, budou moci nové úkoly vytvářet osoby, které mají oprávnění *Vedoucí pracovník*, nebo vyšší.

Vzhled – tato karta umožňuje měnit visuální podobu aplikace. Je zde možné nastavit počet řádků, které se budou zobrazovat na jedné stránce tabulky, barevnost tabulek, barevnost formulářů a způsob zobrazování formulářů po kliknutí levým tlačítkem. Oba způsoby zobrazování jsou níže popsány.

- *Zobrazování Nové okno* – po kliknutí levým tlačítkem bude formulář zobrazen jako samostatné okno. Po kliknutí pravým tlačítkem bude formulář zobrazen jako další záložka hlavního okna.

- *Zobrazování Nová záložka* – funkce zobrazování bude přesně opačná, než u nastavení *Nové okno*.

Nastavení emailů – tato karta (obr. 24) slouží pro nastavení emailové služby, která zajišťuje zasílání informačních emailů o změnách jednotlivých úkolů. Pro korektní funkci této služby je nutné správně vyplnit připojení k SMTP serveru, který daná společnost používá.



Obrázek 24 - Globální nastavení systému

- Pole *adresa* a *heslo SMTP serveru* jsou údaje potřebné k připojení na SMTP server. Pokud server nepožaduje heslo, zůstane pole prázdné.

- *Emailová adresa odesílatele* – jedná se o emailovou adresu, která bude uvedena jako adresa odesílatele u odesílaných emailů.

- *Emailová adresa kopie* – pokud je tato adresa vyplněna, budou všechny emaily odesílány do kopie i na tuto adresu.

Další část panelu umožňuje nastavit pravidla samotného odesílání. Emaily zachycující stav úkolu před změnou a po změně jsou odesílány na emaily jednotlivých uživatelů dle následujícího nastavení.

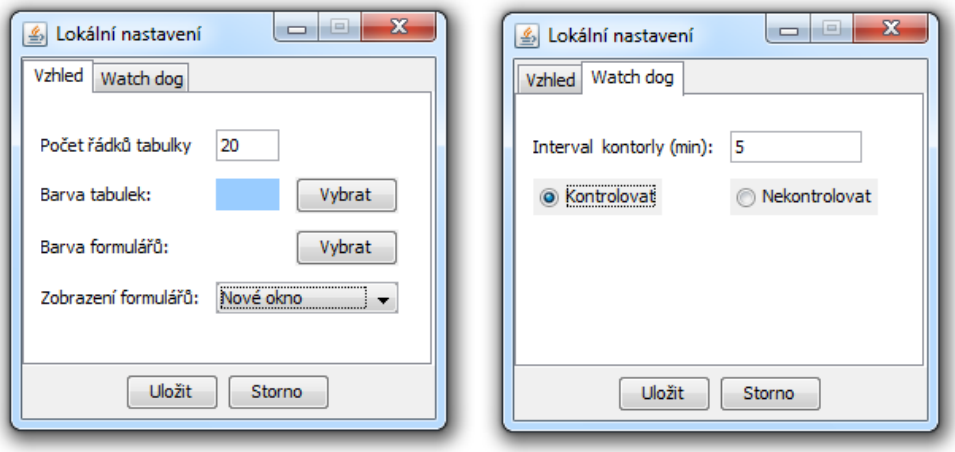
- *Zadavateli* – emaily jsou zasílány pouze uživateli, který úkol vytvořil.
- *Zadavateli a řešiteli* – informační emaily jsou zasílány uživateli, který úkol vytvořil a uživateli, kterému byl úkol aktuálně přiřazen.
- *Všem zúčastněným* – informační emaily jsou zasílány všem uživatelům, kteří na úkol vykázali práci, nebo přidali poznámku.
- Neposílat – informační emaily nebudou vůbec odesílány.

12.2.2 Lokální nastavení

Okno *lokálního nastavení* je dostupné z menu *Nastavení* → *Vlastnosti* → *Lokální nastavení* (obr. 25). Pomocí tohoto okna je na kartě *Vzhled* možné nastavit visuální podobu programu. Význam jednotlivých položek nastavení je totožný s globálním nastavením aplikace.

Na další kartě *Watch dog* je možné nastavit hlídání nově přiřazených úkolů.

- *Interval kontroly* – pokud je vyplněná hodnota intervalu kontroly v minutách a zaškrtnutý radio button *Kontrolovat*, bude systém periodicky kontrolovat, zda uživateli nebyly přiřazeny nové úkoly k vyřešení. Pokud systém zjistí nově přiřazené úkoly, zobrazí je automaticky v přehledné tabulce.



Obrázek 25 - Lokální nastavení systému

12.3 Moduly programu

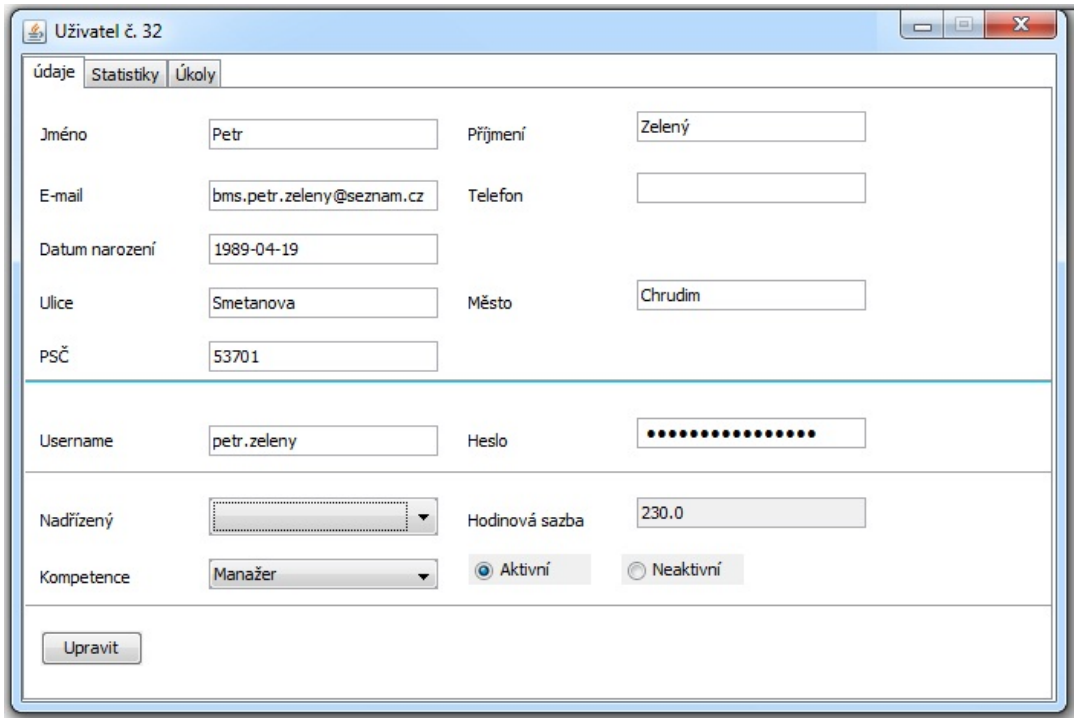
Všechny moduly programu jsou dostupné z nabídky menu *Zobrazit*. Po kliknutí na daný modul, bude zobrazena tabulka s výpisem informací zvoleného modulu. Moduly je také možné vyvolat pomocí klávesových zkratek, které budou uvedeny v dalším popisu.

12.3.1 Modul uživatelé

Tento modul obsahuje výpis všech uživatelů zavedených do systému. Modul je možné vyvolat z menu *Zobrazit → Uživatele*, nebo pomocí klávesové zkratky *Alt + P*. Každý záznam v tabulce je opět možné pomocí tlačítka *Upravit* libovolně editovat.

Formulář editace uživatele (obr. 26) obsahuje kartu s podrobnými údaji o uživateli, kartu s aktuálně přiřazenými úkoly a kartu se statistikou práce.

Přidání nového uživatele do systému je možné pomocí ikony „osoby“ v ovládacím panelu aplikace, nebo pomocí klávesové zkratky *Ctrl + P*. Aby bylo uživatele možné vytvořit, je nutné vyplnit pole *E-mail*, *Username* a *Heslo*. Pro správnou funkci systému musí mít uživatel vyplněno pole *Hodinová sazba*. Pokud by byla sazba nulová, bude tento uživatel vytvářet nulové náklady (toto nastavení může být ve specifických situacích použito). Pomocí roletky *Nadřízený* je možné nastavit nadřízeného vkládané osoby. Nadřízený vidí ve výpisu uživatelů sebe a své podřízené. Svým podřízeným může rovněž upravit hodinovou sazbu.



Obrázek 26 - Formulář uživatele systému

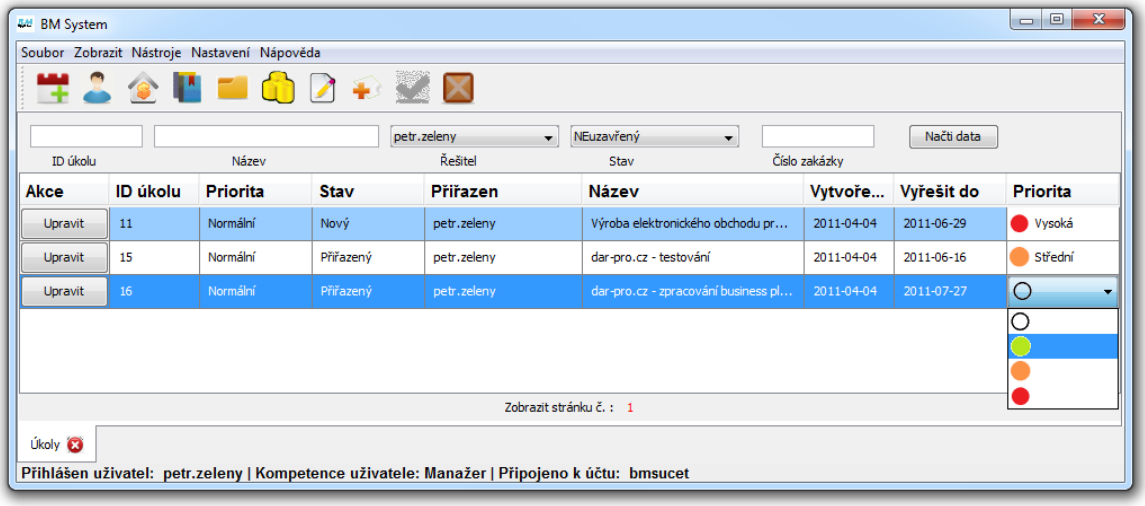
12.3.2 Modul úkoly

Na obrázku 27 je uvedeno zobrazení výpisu modulu úkolů. Tento modul je jedním z nejdůležitějších a zároveň nejsložitějších. Modul *úkoly* je možné zobrazit z menu

Zobrazit → Úkoly, nebo pomocí klávesové zkratky *Alt + U*. Samotný výpis úkolů je velmi intuitivní, proto zde bude popsán pouze sloupec *Akce* a *Priorita*.

Tlačítko *Upravit* ve sloupci *Akce* otevře konkrétní úkol, který je následně možné upravit. Vzhledem k velikosti je vzhled formuláře uveden v přílohách (příloha E).

Sloupec *Priorita* umožňuje používat metodu GTD pro řízení manažerského času. Pokud je u úkolu nastavena jiná priorita, než defaultní (prázdné kolečko), bude úkol automaticky zařazen do zobrazení *Úkoly pro zpracování* (dostupné z menu *Zobrazit → Úkoly ke zpracování*). Pokud bude u úkolu nastavena priorita *Defaultní*, bude úkol opět ze sekce *Úkoly pro zpracování* odebrán. Tato funkce je obdobná jako funkce nastavování priorit v programu Microsoft Outlook.



Obrázek 27 - Uživatelské řízení priorit

Nový úkol je možné přidat kliknutím na ikonu „tabulky s plusem“, nebo pomocí klávesové zkratky *Ctrl + U*. Formulář úkolu se skládá ze dvou karet.

Zadání – na této kartě je uveden konkrétní text úkolu, je zde možné nastavit prioritu, řešitele a další parametry úkolu. Aby bylo možné úkol vytvořit, je třeba, aby měl vyplněné pole *Název úkolu*.

Statistiky – tato karta obsahuje dvě grafické statistiky. První statistika zobrazuje využití časového fondu úkolu. Sloupcový graf porovnává předpokládanou náročnost s aktuálně odpracovaným časem úkolu a zobrazuje jejich rozdíl. Druhá statistika ukazuje v koláčovém grafu rozdělení práce na úkolu seskupené dle jednotlivých pracovníků. Rodičovský úkol agreguje data všech svých potomků.

12.3.3 Modul firmy

Modul *firmy* obsahuje výpis všech firem (zákazníků). Aby bylo možné vytvořit novou *zakázku* a *úkol*, je třeba napojit zakázku na existující firmu. Zobrazení tohoto modulu je možné z menu *Zobrazit → Firmy*, nebo pomocí klávesové zkratky *Alt + F*.

Novou firmu je možné přidat pomocí ikony *„domu“*, nebo použitím klávesové zkratky *Ctrl + F*. Pro přidání firmy je nutné ve formuláři vyplnit její název.

12.3.4 Modul kategorie zakázek

Aby bylo možné lépe organizovat jednotlivé zakázky společností, je pomocí modulu *kategorie zakázek* možné vytvářet ke každé společnosti libovolný počet kategorií, do kterých jsou následně ukládány konkrétní zakázky. Výpis založených kategorií je možné zobrazit z menu *Zobrazit → Kategorie zakázek,* nebo pomocí klávesové zkratky *Alt + K.*

Novou kategorii je možné přidat pomocí kliknutí na ikonu *„složky“*, nebo použitím klávesové kombinace *Ctrl + K*. Ve formuláři následně stačí vybrat v roletkovém menu příslušnou firmu a do pole *Název kategorie* doplnit požadovaný název. Bez založení kategorie zakázek není možné dále na firmu napojit jednotlivé zakázky.

12.3.5 Modul zakázky

Modul z*akázky* obsahuje seznam všech zakázek napojených na konkrétní společnost (kategorii zakázek vytvořenou pro tuto společnost). Zobrazení modulu je možné z menu *Zobrazit → Zakázky,* nebo pomocí klávesové zkratky *Alt + Z.*

Novou *zakázku* je možné přidat kliknutím na ikonu „desek“, nebo pomocí klávesové zkratky *Ctrl + Z*. Každá zakázka se skládá ze tří karet.

Zakázka – tato karta obsahuje základní informace o zakázce (datum vytvoření, společnost, kategorii zakázek apod.) a také panely příjmů a nákladů.

Příjmy jsou tvořeny manuálně manažerem pomocí tlačítka *Přidat profit*. Příjem je typicky přidáván po uzavření obchodu se zákazníkem a je ve výši fakturované částky za zboží či služby.

Náklady mohou být rovněž vytvořeny manažerem pomocí tlačítka *Přidat náklad*. V tomto případě se nejčastěji jedná o takzvaný externí náklad, který vzniká společnosti v důsledku nákupu služeb od externích dodavatelů.

Náklady jsou dále tvořeny automaticky po uložení výkazu práce. Tímto způsobem se promítá práce pracovníků do nákladů společnosti. Tyto náklady mají v poli *Typ* uvedeno *Výkaz práce*. Tento typ nákladu rovněž zobrazuje informaci o čísle výkazu a začlenění výkazu do výplatní listiny. Kliknutím na číslo výkazu se automaticky zobrazí formulář s daným pracovním výkazem. Pokud byl výkaz práce začleněn do výplatní listiny, bude ve sloupci *Zaplaceno* uvedeno slovo *Ano*. Po kliknutí na toto slovo se zobrazí formulář příslušné výplatní listiny. Tyto moduly jsou tedy spolu úzce provázány. Formulář zakázky s výpisem příjmů a nákladů je z důvodu velikosti uveden v příloze (příloha F).

Úkoly – na této kartě je zobrazen výpis nevyřešených úkolů, které jsou navázány na zakázku. Nastavení filtru úkolů je samozřejmě možné změnit, tudíž si uživatel může zobrazit například veškeré úkoly, které již byly na zakázce odvedeny.

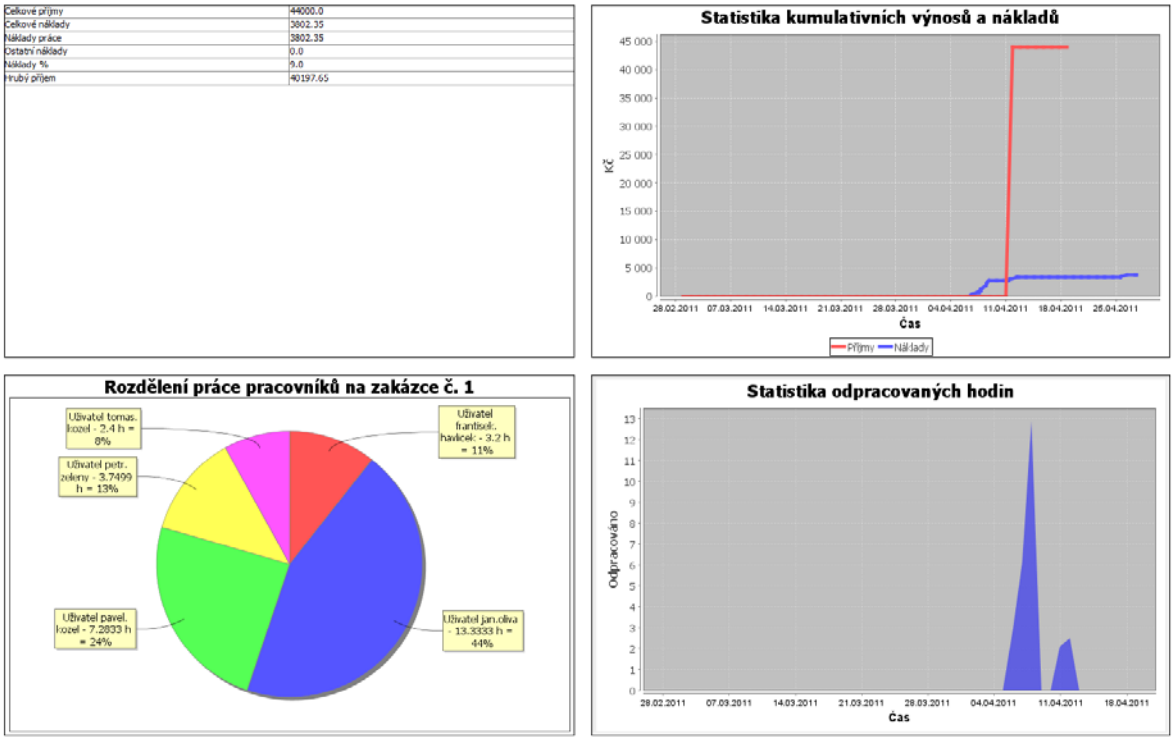
Statistiky – tato karta obsahuje podrobné statistiky výnosů, nákladů a rozdělení vykázané práce pracovníků. Statistiky je možné rozdělit do čtyř částí.

Sumarizační tabulka – obsahuje informace v číselné formě, které jsou přehledně uspořádány do tabulky. Z tabulky je možné vyčíst celkové náklady, celkové náklady na práci, hrubý příjem, podíl výrobních nákladů v procentech a podobně.

Statistika kumulativních výnosů a nákladů – tento graf přehledně zobrazuje vývoj výnosů a nákladů zakázky v čase. Graf je zobrazován pouze za dobu platnosti zakázky. Na ose x je tedy rozmezí od data zahájení po datum ukončení.

Rozdělení práce pracovníků na zakázce – tento koláčový graf zobrazuje jednotlivé podíly časového fondu pracovníků, kteří na zakázce pracovali.

Statistika odpracovaných hodin – tento graf zobrazuje objem prací všech pracovníků, zobrazený v závislosti na čase. Graf je opět zobrazován pouze v rozsahu platnosti příslušné zakázky. Konkrétní grafické statistiky jsou uvedeny níže na obrázku 28.

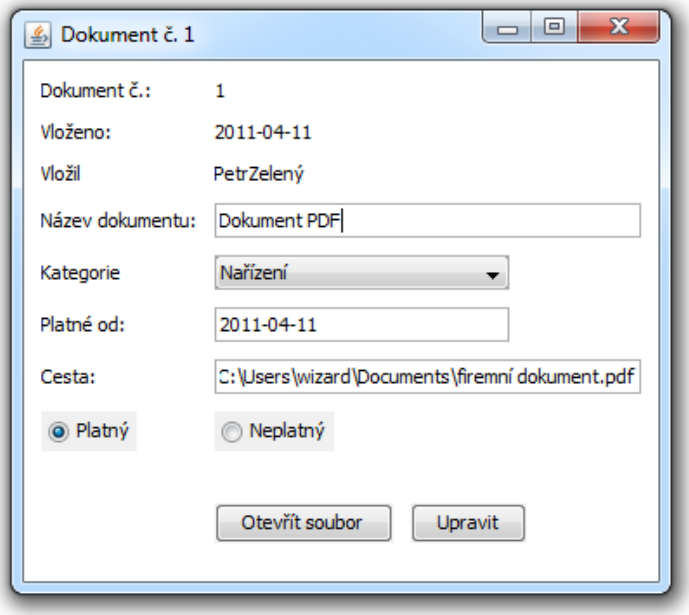


Obrázek 28 - Statistiky zakázky

12.3.6 Modul firemní dokumenty

Tento modul obsahuje výpis záznamů zvaných *firemní dokumenty*. Zobrazení modulu je možné z menu *Zobrazit → Dokumenty*, nebo pomocí klávesové zkratky *Alt + D*. Pomocí tlačítka *Upravit* je možné vstoupit do editace dokumentu. *Firemní dokumenty* je rovněž možné ze systému vymazat pomocí tlačítka *Odstranit*, které je dostupné ve výpisu modulu. Pokud budou dokumenty odstraňovány, dojde k narušení konzistence číselné řady jednotlivých dokumentů.

Nový *firemní dokument* je možné přidat pomocí ikony „listu s tužkou“, nebo použitím klávesové zkratky C*trl + D*. Každý záznam obsahuje kategorii dokumentu, cestu k danému souboru a informace o jeho platnosti. Zobrazení formuláře modulu *firemní dokumenty* je uvedeno na následujícím obrázku 29. Pokud je uvedená cesta k souboru správná, může uživatel pomocí tlačítka *Otevřít soubor* jednoduše spustit daný soubor v defaultně nastaveném programu operačního systému. *Kategorie dokumentů* je možné uživatelsky rozšířit prostřednictvím administrátorského účtu.



Obrázek 29 - Formulář firemního dokumentu

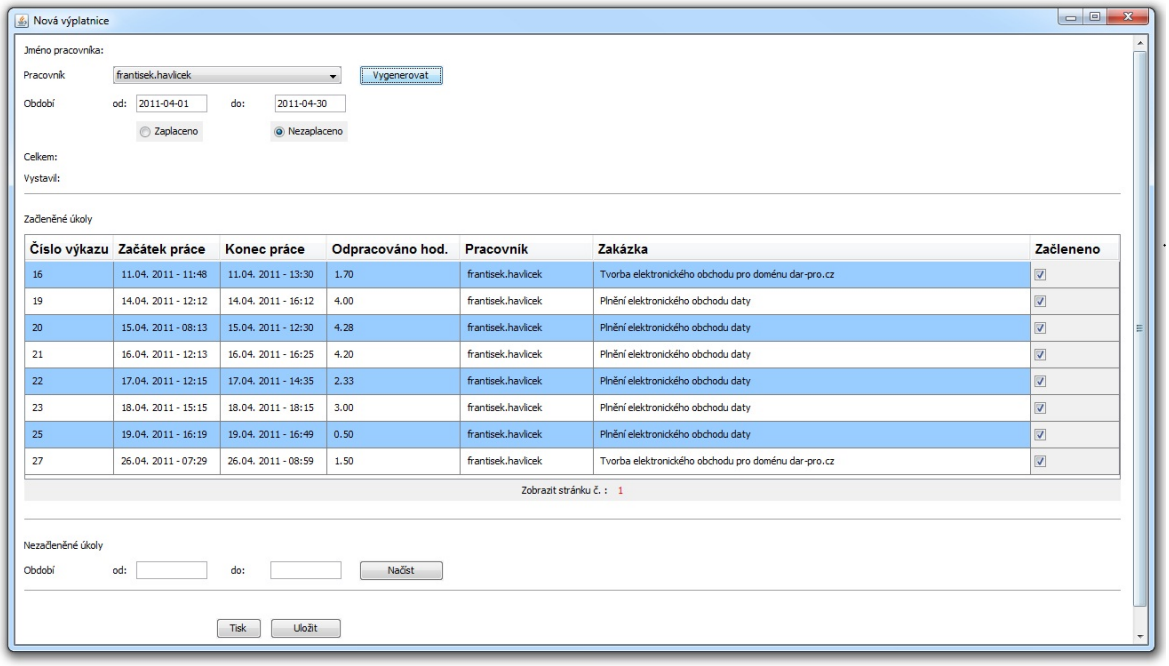
12.3.7 Modul výplatní listiny

Modul *výplatní listiny* umožňuje jednoduše vygenerovat seznam odpracovaných úkolů, jejich celkovou časovou náročnost a příslušnou finanční odměnu, která z těchto úkolů plyne. Samozřejmostí je i možnost vytisknutí vytvořené výplatní listiny. Tento modul je dostupný z menu *Zobrazit → Výplatnice*, nebo pomocí klávesové zkratky *Alt+ W*.

Novou výplatní listinu je možné vytvořit pomocí ikony „listu s plusem“, nebo použitím klávesové zkratky *Ctrl + W*.

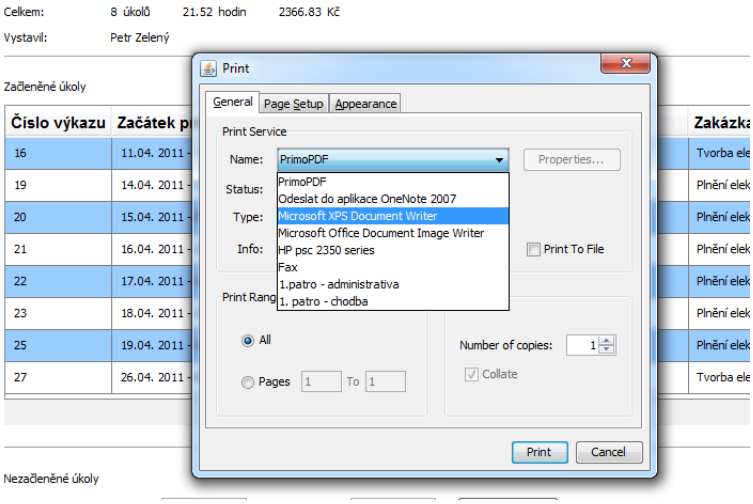
Pomocí pole *Pracovník* vybere uživatel příslušnou osobu, které chce výplatní listinu vygenerovat. Následně je třeba zadat do příslušných polí časový interval, pro který je výplatní listina generována. Po stisku tlačítka *Vygenerovat* budou načteny všechny odpracované úkoly spadající do zadaného období. Všechny úkoly budou pomocí zaškrtávacího pole ve sloupci *Začleněno* automaticky přidány do výplatní listiny. V případě, že některý úkol nechceme pracovníkovi proplatit, stačí tuto položku tabulky odstranit z výpisu odškrtnutím zmíněného pole *Začleněno*.

Nezačleněné úkoly mohou být opět přidány do výplatní listiny pomocí sekce *Nezačleněné úkoly*. Tato sekce vyhledá všechny nezačleněné úkoly v zadaném časovém intervalu a stejným způsobem, tedy zaškrtnutím pole *Začleněno*, je umožní přidat do výplatní listiny. Formulář výplatní listiny je uveden níže na obrázku 30.



Obrázek 30 - Formulář výplatní listiny

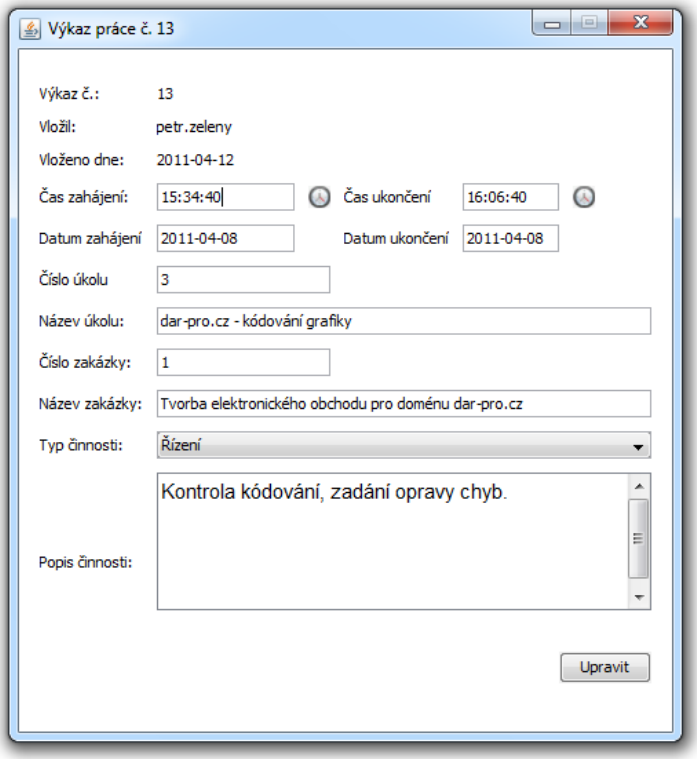
Stisknutím tlačítka *Uložit* dojde k začlenění všech zaškrtnutých položek, jsou spočítány celkové sumy úkolů a finanční odměny. Pokud je výplatní listina úspěšně uložena, je možné pomocí tlačítka *Tisk* spustit tiskové okno a výplatní listinu libovolným způsobem vytisknout (obr. 31).



Obrázek 31 - Tisk výplatní listiny

12.3.8 Modul výkazy práce

Pomocí modulu *výkazy práce* jsou evidovány veškeré pracovní aktivity na zadaných *úkolech* či *zakázkách*. Každý výkaz tedy musí být vykázaný na existující *úkol* nebo *zakázku*. V případě, že zakázka nemá vytvořeny žádné úkoly, je možné práci vykázat přímo na číslo zakázky. Formulář výkazu práce je uveden na obrázku 32.



Obrázek 32 - Formulář výkazu práce

Ve formuláři je do polí *zahájení* a *ukončení* automaticky vyplněno aktuální datum a aktuální čas. Pomocí kliknutí na ikonu hodin bude pole *Čas zahájení* nebo *Čas ukončení* aktualizováno.

Pokud je práce vykazována na konkrétní úkol, stačí vyplnit pole *Číslo úkolu* a stisknout klávesu enter. Následně budou automaticky načtena pole *Název úkolu*, *Číslo zakázky* a *Název zakázky*. Pokud je práce vykazována pouze na zakázku, stačí uvedený postup zopakovat pro pole *Číslo zakázky*.

Pole *Typ činnosti* slouží pro rozlišení jednotlivých druhů práce, a jeho položky je možné editovat pomocí číselníku.

Pole *Popis činnosti* obsahuje konkrétní popis práce, která byla odvedena.

Po stisknutí tlačítka *Uložit* dojde k uložení výkazu práce a paralelně bude k dané zakázce přidána jedna nákladová položka s hodnotou nákladu, která je shodná s cenou vykázané práce. Tímto způsobem jsou náklady práce přenášeny do nákladových položek každé zakázky.

12.4 Nástroje programu

Jak již bylo dříve zmíněno, systém obsahuje několik nástrojů ulehčujících manažerskou práci. V dalších kapitolách budou tyto nástroje popsány.

12.4.1 Automatická kontrola nově přiřazených úkolů

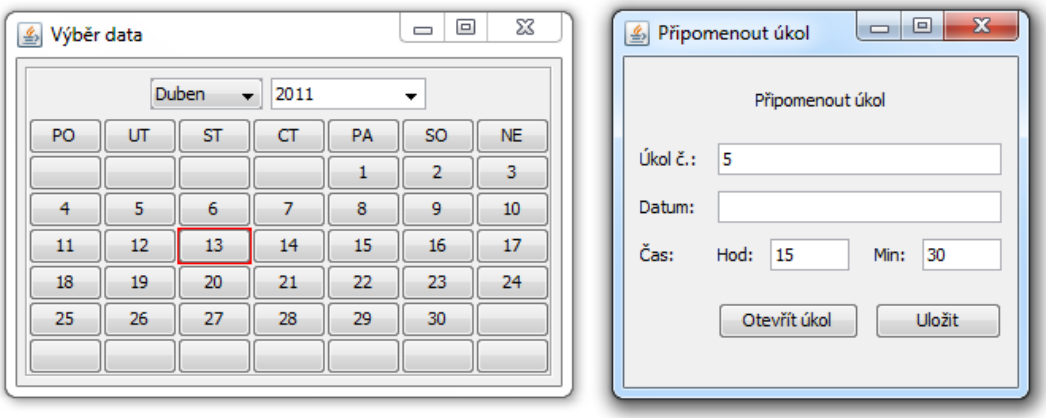
Aby byl manažer včas informován o nově vzniklé povinnosti řešit úkol, má možnost nastavit v systému automatickou kontrolu nově přiřazených úkolů. Nastavení se provádí v menu *Nastavení → Vlastnosti → Lokální nastavení*. Na kartě *Watch dog je* možné nastavit interval kontroly v minutách a samotné kontrolování spustit. Následně systém v databázi periodicky kontroluje, zda byly uživateli přiřazeny nové úkoly. V případě, že ano, bude po časovém intervalu zobrazena přehledná tabulka nově přiřazených úkolů, se kterými je ihned možné známým postupem.

Vzhledem k tomu, že kontrola běží na pozadí ve vlastním vlákně, není systém tímto nástrojem příliš zatěžován a jeho rychlost není snížena.

12.4.2 Připomínání úkolů

Jelikož je třeba některé úkoly začít řešit v přesný čas, systém umožňuje nastavení připomínání úkolů v konkrétním čase.

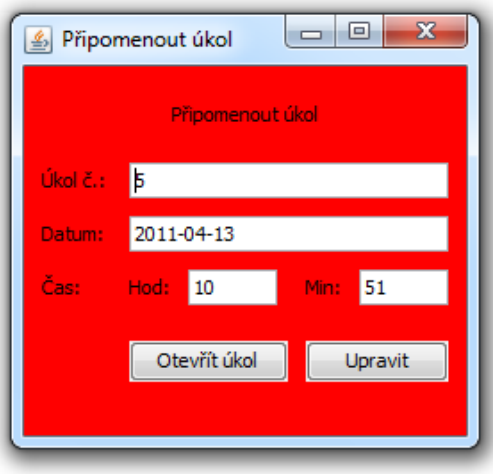
Nastavení připomínání se provádí v menu *Nástroje → Připomínání úkolů* (klávesová zkratka *Ctrl + G*). Formulář připomínání úkolu je zobrazen na následujícím obrázku 33 spolu s panelem kalendáře, který se zobrazí po kliknutí do pole *Datum*. Následným kliknutím na konkrétní den je automaticky vloženo datum do příslušného pole.



Obrázek 33 - Nástroj pro připomínání úkolů

Po stisknutí tlačítka *Uložit* bude událost připomenutí úkolu uložena do kalendáře událostí a rovněž bude naplánováno její připomenutí. Ve správný čas zobrazí systém automaticky formulář připomínání v červené barvě. Pomocí tlačítka *Otevřít úko*l je pak následně možné přejít na editaci připomínaného úkolu. Rovněž je možné upravit čas a datum připomínání. Tímto způsobem dojde k posunutí řešení daného úkolu. Všechny sledované akce je možné prohlížet a upravovat prostřednictvím modulu dostupného

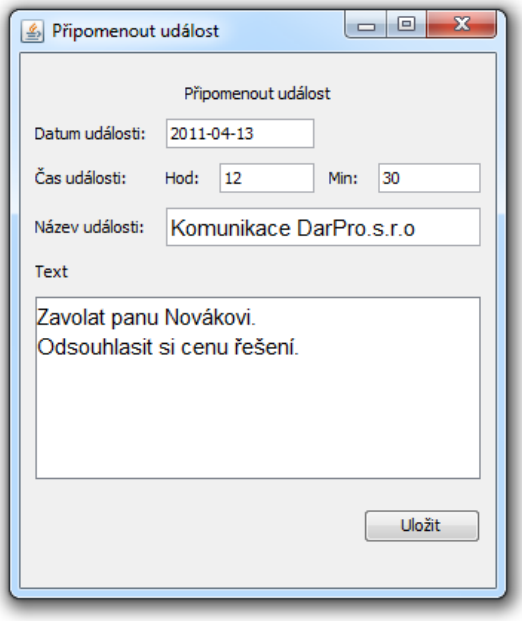
z menu *Zobrazit → Sledované akce* (Klávesová zkratka *Alt + S*). Akci je možné standardním způsobem editovat. Na obrázku 34 je uveden zobrazený formulář s připomenutím úkolu.



Obrázek 34 - Automatické okno s připomenutím úkolu

12.4.3 Připomínání událostí

Připomínání událostí funguje na obdobném principu jako připomínání úkolů. Tento nástroj však slouží k připomínání textových poznámek, které si uživatel sám vytvoří. Modul je dostupný z menu *Nástroje → Připomínání událostí (klávesová zkratka Ctrl + A)*. Formulář pro připomínání událostí je uveden na obrázku 35.



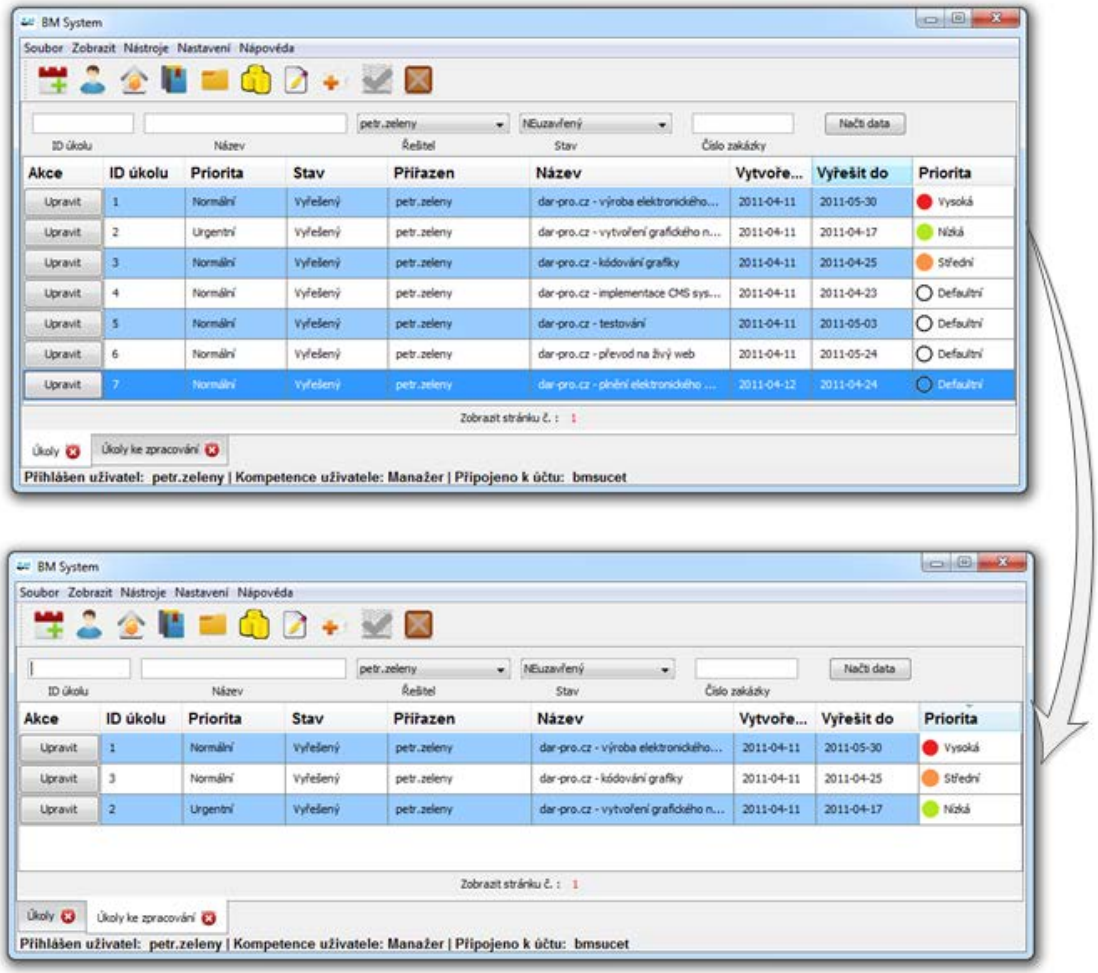
Obrázek 35 - Formulář pro připomínání událostí

12.4.4 Řízení vlastních priorit v souladu s metodou GTD

Vzhledem k tomu, že náročnost úkolů, které se danému manažerovi nashromáždí, může značně přesáhnout jeho časové možnosti, umožňuje systém individuální nastavení priorit úkolů. Všechny úkoly, které mají uživatelsky nastavenou prioritu vyšší, než *Defaultní*, jsou přesunuty do sekce *Úkoly ke zpracování*. Tato sekce je dostupná z menu *Zobrazit → Úkoly ke zpracování*.

Manažer se následně může věnovat pouze řešení prioritních úkolů a není rozptylován úkoly s nízkou prioritou. Po vyřešení úkolu a nastavení jeho priority opět na *Defaultní* bude úkol automaticky odstraněn ze sekce *Úkoly ke zpracování*.

Na obrázku 36 je uvedeno zobrazení modulů *Úkoly* a *Úkoly ke zpracování* při využití nástroje řízení vlastních priorit.



Obrázek 36 - Uživatelské řízení priorit

13 Závěr

Cílem této práce bylo vyvinout funkční aplikaci, která umožňuje řízení a evidenci práce ve firmě. Systém měl v závislosti na provedené analýze dostupných řešení vynikat především jednoduchým ovládáním, moderním uživatelským vzhledem a rychlostí.

Vyvinutá aplikace obsahuje všechny moduly, které byly stanoveny v zadání. Systém rovněž splňuje zadaný cíl, tedy umožnění plnohodnotného řízení a evidence práce. Aplikace je navíc rozšířena o nástroje, které podporují efektivní řízení manažerského času. Vzhledem k rozsahu problematiky týkající se manažerského řízení je stále velký prostor pro vylepšování funkcí. Jedná se především o vylepšení manažerských nástrojů v souladu s metodou GTD, vylepšení modulu úkolů, zakomponování více ekonomických ukazatelů a vytvoření exportního můstku pro účetní programy.

Při porovnání vzniklé aplikace s dostupnými nástroji na trhu si troufám říci, že aplikace je konkurenceschopná a nabízí i nástroje, kterými ostatní aplikace nedisponují. Samotnému nasazení do ostrého provozu by mělo předcházet důkladné otestování, které zatím nebylo možné realizovat kvůli omezené době vývoje aplikace.

Díky vývoji aplikace jsem se podrobně zabýval oblastmi programovacího jazyka Java, se kterými jsem se během studia nesetkal. Tím se značně rozšířily moje odborné znalosti tohoto programovacího jazyka. Dále jsem hlouběji pronikl do širokého spektra praktických dovedností, které zahrnuje samotný návrh složitější aplikace, konkrétní implementaci systému dle zásad objektově orientovaného programování a tvorbu dokumentace.

14 Literatura

[1] NOVOTNÝ, P., POUR, J., SLÁNSKÝ, D. *Business Inteligence: Jak využít bohatství ve vašich datech*. 1. vyd. Praha: Grada, 2005. 256 s. ISBN 80-247-1094-3.

[2] *Business Intelligence (BI)* [online]. c2009 [cit. 2011-04-19]. Dostupné z www: <http://n-i-c.cz/bi_basic.html>.

[3] LANGE, Andreas. *OLAP - pohled na svět podnikání* [online]. 11. 5. 2005 [cit. 2011-04-19]. Dostupné z www: <http://www.cvis.cz/hlavni.php?stranka=novinky/clanek.php&id=259>.

[4] GATINA, Petr. *Programování v jazyku Java (1) – Úvod* [online]. 9. 7. 2004 [cit. 2011-04-19]. Dostupné z www: <http://www.linuxsoft.cz/article.php?id_article=244>.

[5] BOSÁK, Petr. *Úvod do programovacího jazyka Java* [online]. 24. 4. 2006 [cit. 2011-04-19]. Dostupné z www: <http://programujte.com/?akce=clanek&cl=2006041804-uvod-do-programovacieho-jazyka-java>.

[6] FLOWER, Amy. *A Swing Architecture Overview* [online]. c2011 [cit. 2011-04-19]. Dostupné z www: <http://java.sun.com/products/jfc/tsc/articles/architecture>.

[7] ŠEDA, Jan. *Úvod do JDBC* [online]. 4. 3. 2003 [cit. 2011-04-19]. Dostupné z www: <http://interval.cz/clanky/uvod-do-jdbc>.

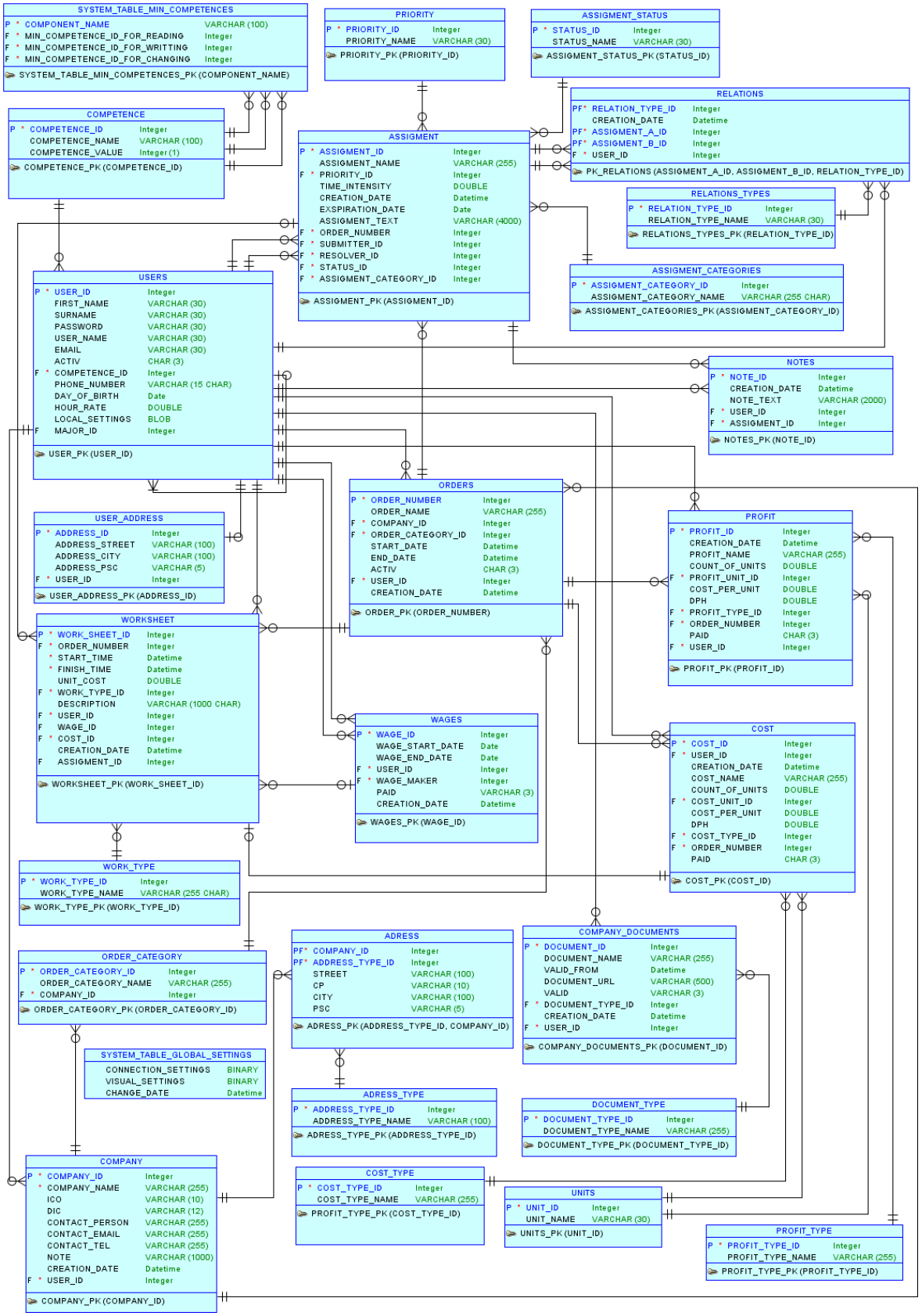
[8] *Using Prepared Statements* [online]. c2011 [cit. 2011-04-19]. Dostupné z www: <http://download.oracle.com/javase/tutorial/jdbc/basics/prepared.html>.

[9] *Welcome To JFreeChart!* [online]. c2009 [cit. 2011-04-19]. Dostupné z www: <http://www.jfree.org/jfreechart/index.html>.

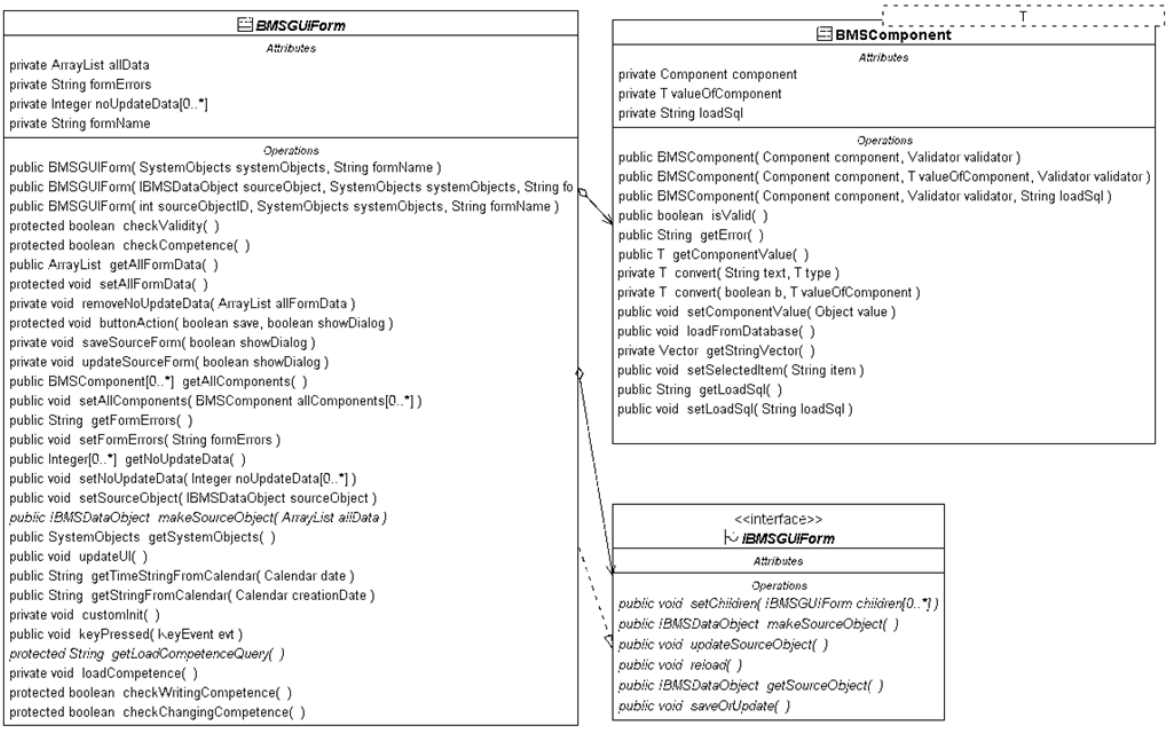
[10] WELLING, L., THOMSON, L. *MySQL: Průvodce základy databázového systému*. 1. vyd. Brno: CP Books, 2005. 255 s. ISBN 80-251-0671-3.

[11] DUBOIS, Paul. *MySQL profesionálně*. Brno: Mobil Media, 2003. 1071 s. ISBN 80-86593-41-X.

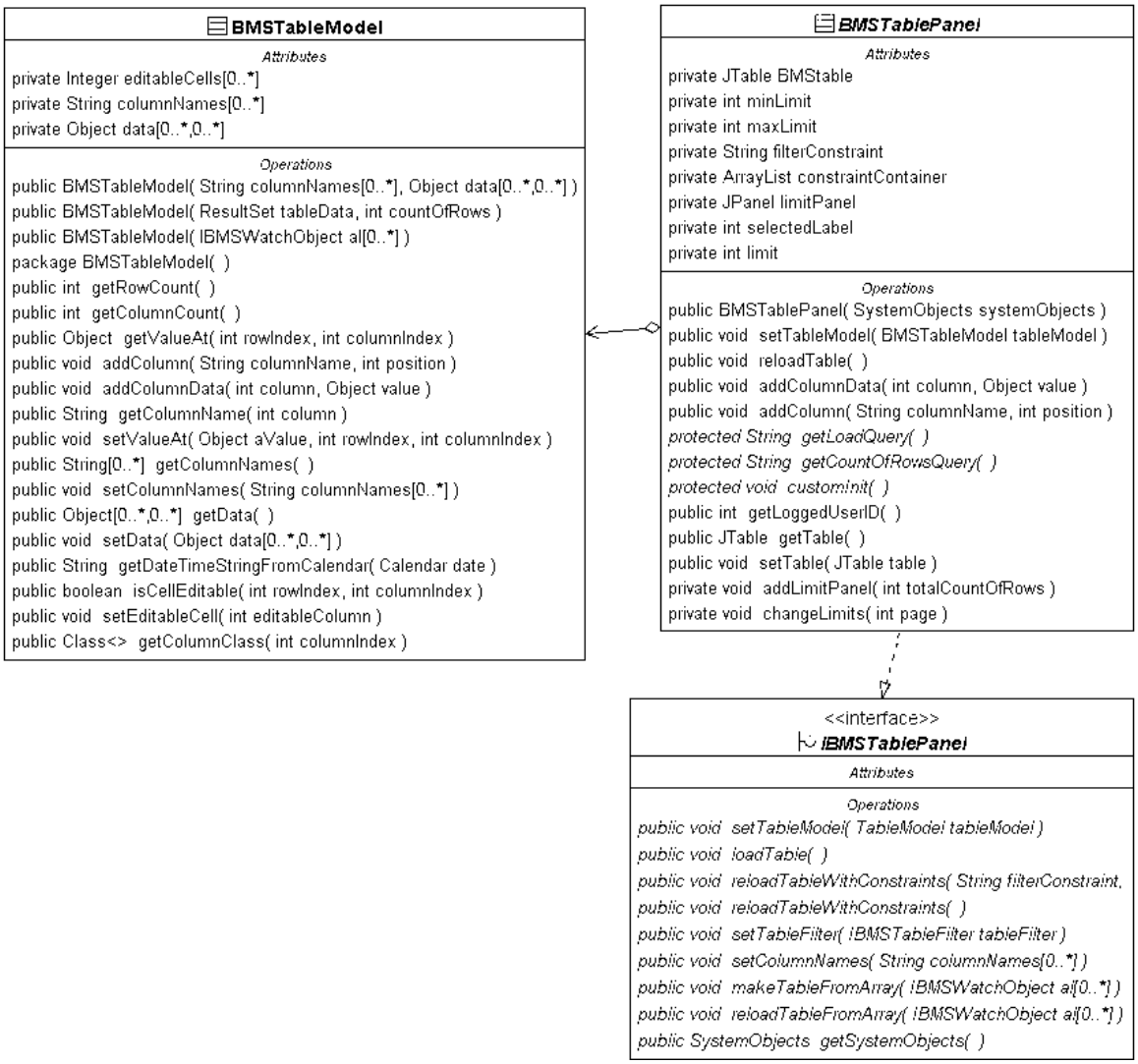
Příloha A – ER diagram databáze



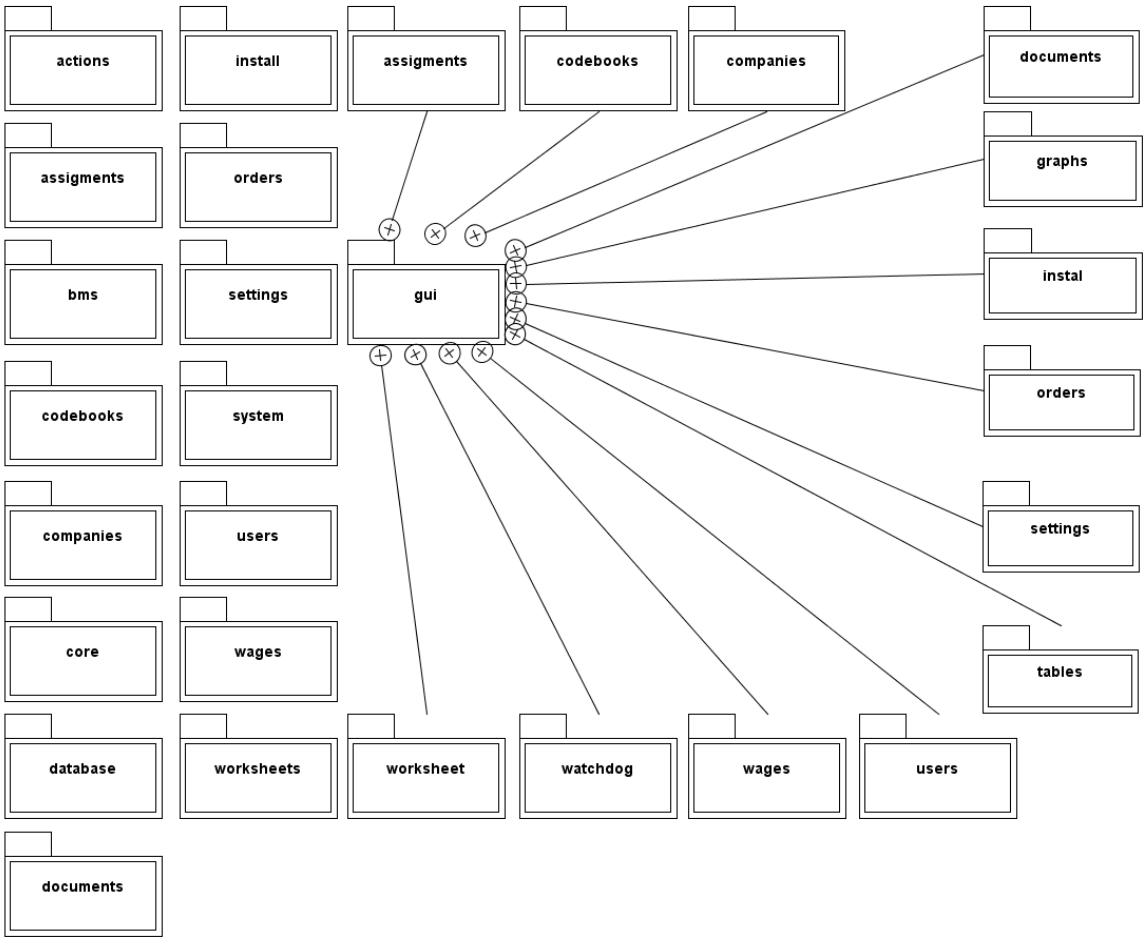
Příloha B – UML diagram tříd grafických formulářů



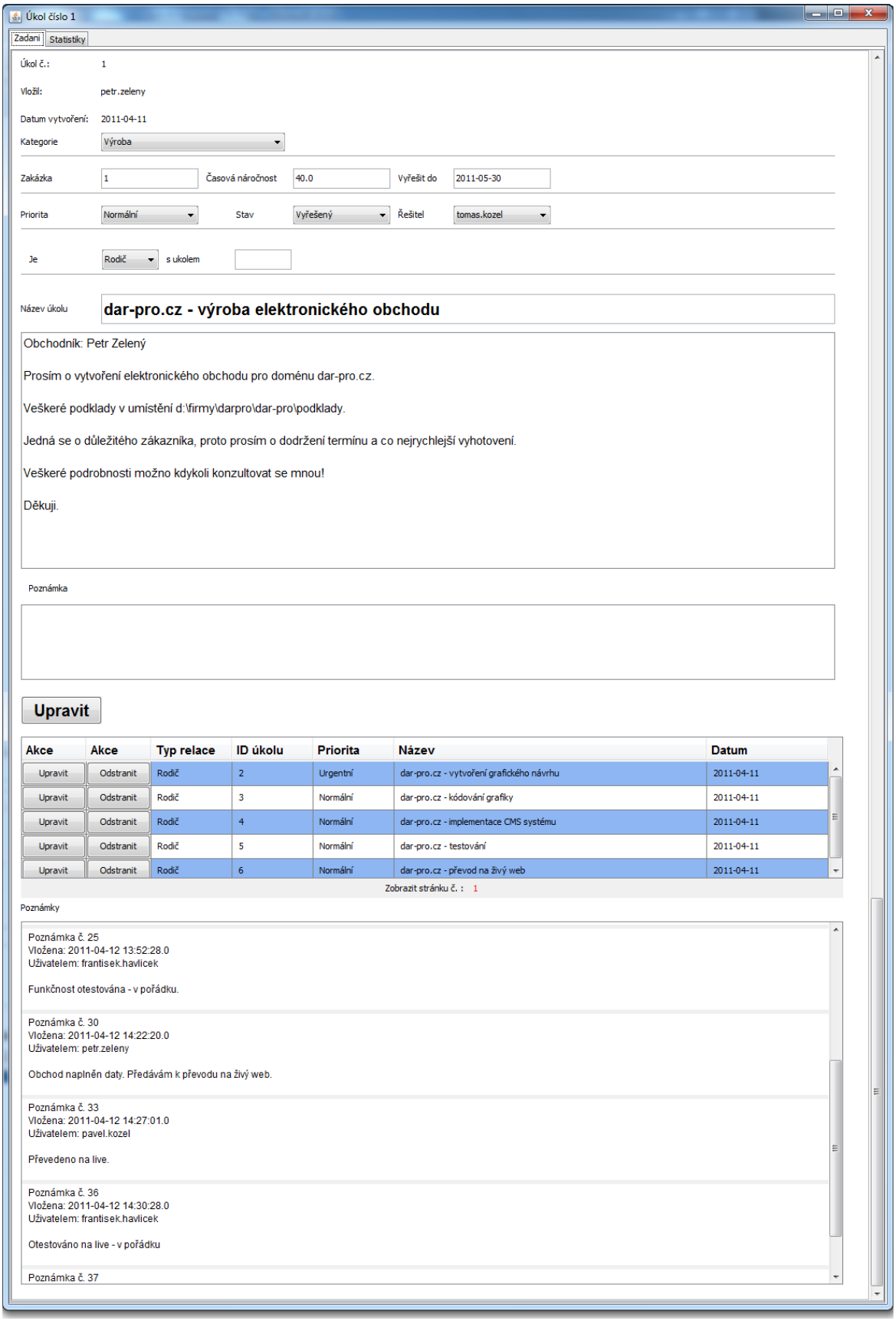
Příloha C – UML diagram tříd tabulek



Příloha D – znázornění struktury balíčků aplikace



Příloha E – formulář úkolu



Příloha F – formulář zakázky