UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Systém pro správu a plánování zdrojů v rámci betonářského podniku

František Koranda

Bakalářská práce
2014

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**František Koranda**
Osobní číslo:	**I09153**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Systém pro správu a plánování zdrojů v rámci betonářského podniku**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Teoretická část bakalářské práce bude zaměřena na problematiku ERP systémů. Stručně bude popsána jejich historie a současné trendy. Dále bude popsána problematika přístupu Java aplikací k databázovým systémům Oracle.
Praktická část se bude věnovat implementaci systému pro správu a plánování zdrojů v rámci betonářského podniku. Stěžejní pro aplikaci bude možnost správy všech hmotných a lidských zdrojů společnosti. Systém bude obsahovat i tvorbu reportů.
Použité technologie: Java SE, Oracle XE.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

**1. Lacko, L.: Oracle - Správa, programování a použití databázového systému. Computer Press, 2007.
ISBN 978-80-251-1490-2.
2. Kyte, T.: Oracle - Návrh a tvorba aplikací. Computer Press, 2006. ISBN 80-251-0569-5.
3. Spell, B.: Java - Programujeme profesionálně. Computer Press, 2002. ISBN 80-7226-667-5.
4. Schildt, H.: Java 7 - Výukový kurz. Computer Press, 2012. ISBN: 80-251-1156-3.
5. Oracle Database 11g release 2. Oracle database Documentation library [online]. Oracle America, Inc., 2013 [cit. 2013-10-31]. Dostupné z: http://www.oracle.com/pls/db112/homepage**

Vedoucí bakalářské práce: **Ing. Tomáš Váňa**
Katedra informačních technologií

Datum zadání bakalářské práce: **20. prosince 2013**
Termín odevzdání bakalářské práce: **9. května 2014**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2014

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 01. 05. 2014 Koranda František

Poděkování

Tímto bych rád poděkoval panu Ing. Tomáši Váňovi za odborné rady a vedení při tvorbě této práce.

Anotace

Práce popisuje vznik a vývoj ERP systémů, spolu se současnými trendy těchto systému. Dále je popsána problematika přístupu Java aplikací k relačnímu databázovému systému Oracle pomocí ovladače JDBC. V praktické části je řešena implementace aplikace zabývající se správou a plánováním zdrojů v rámci betonářského podniku spolu s návrhem a implementací relační databáze Oracle. Možnosti a funkcionalita jsou detailně popsány v druhé části práce.

Klíčová slova

JDBC, Java SE, Oracle XE, Java GUI aplikace, relační databáze, ERP systémy

Title

The system of planning and management of resources in the concreting business

Annotation

The paper describes the origin and development of ERP systems, along with the current trends of those systems. It also describes the issue of acces Java applications to relational databases using Oracle JDBC drivers. In the practical part is described the application for the management and distribution of resources within the concret company together with the design and implementation of the Oracle relational database. Options and functions are described in detail in the second part.

Keywords

JDBC, Java SE, Oracle XE, Java GUI applications, relations database, ERP systems

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek

API	Application Programming Interface.
GUI	Graphical User Interface.
Java SE	Java Standart Edition.
JDBC	Java Database Connectivity.
ERP	Enterprise Ressource Planning.
WAN	Wide Area Network.
CRM	Customer Relationship Management.
UI	User Interface.
UX	User Experience.
MRP	Material Requirements Planning.
IS	Information System.
SW	Software.
OS	Operation System.
ODBC	Open Database Connectivity.

Úvod

Před příchodem ERP systémů na trh existoval problém, kde každé oddělení či pracoviště společnosti mělo své aplikace informační systémy. Neexistovalo žádné centrální místo, kde by se veškerá podniková data uchovávala. To způsobovalo velké problémy zejména při komunikaci. Jak se ERP systémy postupně vyvíjely, byl postupně tento problém eliminován. Dnes ERP systémy integrují a automatizují značné množství podnikových činností, jež jsou esenciální pro provoz dané společnosti.

V současnosti jsou tyto systémy nedílnou součástí téměř každé firmy či podniku. Typicky je tyto společnosti využívají k řízení procesů, jako jsou výroba, logistika či distribuce. ERP systémům je věnována první část této práce, kde je detailně popsána jejich filozofie, architektura, vývoj i současné trendy. Na závěr je uvedeno i několik nejznámějších dodavatelů těchto systémů.

Následující část bakalářské práce je zaměřena na programovací jazyk Java a jeho přístup k databázovému systému. Konkrétně se tato práce zabývá přístupem přes jednotné rozhraní JDBC, které se snaží o standardizaci přístupu aplikací napsaných v Javě k databázovým systémům. Je zde popsáno vše od připojení k databázi až po získávání dat a jejich zpracování.

Praktická část práce se věnuje aplikaci, která byla cílem této práce. Jedná se o aplikaci sloužící ke správě a plánování zdrojů v rámci betonářského podniku. Samotná aplikace je napsána v programovacím jazyce Java. Tato aplikace názorně ukazuje, jak lze zjednodušit komunikaci s databází i práci s daty na intuitivní úroveň a zároveň slouží jako ukázka jednoduchého ERP systému. Aplikace je napsaná v jazyce Java a jsou detailně popsány všechny její možnosti i funkce. Dále je jedna kapitola věnována návrhu a struktuře databáze, ve které se ukládají všechna data, se kterými aplikace pracuje.

1 ERP systémy

Enterprise ressource planning [1] je typ informačního systému, který soustřeďuje informace ze všech oddělení firmy do jednoho místa tak, aby se zvýšila hospodárnost, výkonnost a zisk organizace. Dříve než byly vyvinuty ERP systémy, každé oddělení bylo považováno za samostatnou jednotku. Typicky to znamenalo, že v každém oddělení se soustředili jen na vlastní činnost. Toto ale vedlo k četným komplikacím. Dnes, místo toho, aby mělo každé oddělení svoji vlastní databázi informací, ERP soustřeďuje všechny tyto informace do jednoho místa. Toto napomáhá v rozhodovacím procesu každého oddělení a oddělení již nemusí dlouho čekat na odpověď od jiného oddělení ve firmě a mohou se rozhodovat rychleji. Mnoho věcí se tak zjednodušuje.

Informační systém ERP je balík softwarových komponent, který může zahrnovat následující vlastnosti:

1. Integrovaný systém, který funguje v reálném čase a nevyžaduje, aby se uživatel spoléhal na pravidelné dávkové aktualizace.
2. Typický ERP systém ukládá svá data do relačního databázového systému.
3. Každý modul by měl mít stejný vzhled a ovládání tak, aby věci byly přehledné.
4. Každé oddělení by mělo mít přístup do systému bez nutnosti přímé osobní asistence IT oddělení.

Tento systém zajistí, že procesy ve společnosti budou fungovat hospodárně, a že se nebudou vyskytovat duplicitní záznamy a nebudou se provádět duplicitní činnosti. Toto velmi pomůže managementu snížit jeho pracovní zátěž a přinese kvalitu do plánovacího a rozhodovacího procesu. Plánování ve firmě může být prováděno pomocí tohoto systému, jelikož systém již obsahuje všechny potřebné informace pro rozhodování.

ERP systém je třeba nakonfigurovat správným způsobem, aby efektivně fungoval. Tím se předejde komplikacím v komunikaci mezi odděleními a managementem.

ERP systém implementuje funkcionality, jako jsou správa finančních zdrojů a toků, řízení materiálu i lidské zdroje. Umožňuje toky podnikových informací uvnitř organizace i komunikaci se zúčastněnými osobami z vnějšího prostředí. ERP používá databázi a počítačovou síť. Shromáždí a spojí všechny role a funkce firmy do jednoho standardního a celo-firemního systému. Jelikož je systém plně integrovaný, může být nahrán na centrální server nebo dokonce rozdělen do více hardwarových a softwarových komponentů. Tyto komponenty, které se mohou nacházet v různých prostorech, komunikují navzájem prostřednictvím lokální počítačové sítě či WAN. Takto lze ušetřit, protože není nutné instalovat vždy celý systém na více místech.

Existuje mnoho komerčních aplikací ERP systému např.: výroba, řízení dodavatelského řetězce (SCM), řízení vztahů se zákazníky (CRM), kontrola docházky a přístupu, finanční

funkce, personalistika, datové služby. Řízení work-flow, kontrola kvality, kontrola nákladů, výrobní proces a toky jsou příkladem výrobních funkcionalit pokrývaných systémem. Costing, time management a kontrola výdajů, výkonnost, activity management a řada dalších jsou funkcionality projektového řízení, které jsou monitorovány ERP systémem. Finanční funkcionalita jako vedení hlavní knihy, řízení hotovosti, hmotná aktiva, pohledávky a závazky jsou také spravovány pomocí tohoto systému. Dokonce funkčnosti jakými jsou změnové řízení, jakost, skladové hospodářství, obchod a marketing, kontakt se zákazníkem a podpora call centra jsou obsluhovány tímto systémem.

1.1 Historie ERP

Počátky ERP [2] systémů sahají do šedesátých let dvacátého století, kdy větší podniky jako Toyota, IBM nebo M & D začaly vyvíjet a nasazovat celopodnikové informační systémy, které se staraly především o kontrolu zásob a skladů. Tyto systémy byly vyvíjeny v tehdy rozšířených programovacích jazycích jako COBOL, FORTRAN nebo ALGOL.

V roce 1972 vzniká dnes světoznámá společnost SAP (Systém analyse und Programmentwicklung – Systémová analýza a vývoj produktů). Motivací pro vytvoření systému SAP bylo vytvořit standardní software pro trh s integrovanými podnikovými řešeními. O pár let později, roku 1977, byla založena další světová společnost OracleCorporation, která pak o dva roky později nabízí první komerční databázový řídicí systém založený na SQL jazyce.

MRP měl velký nedostatek, že neobsahoval plánování výrobních zdrojů a kapacit.

	1. generace	2. generace	3. generace	4. generace	5. generace
Způsob zpracování	dávkové	v dialogu	v dialogu i v dávce	volitelné	prostřednictvím internetu
Přenositelnost	spojení s určitým počítačem – HW vazba	vazba na určitý operační systém	přenositelnost mezi operačními systémy – např. UNIX, OS400	třívrstvé aplikace (databáze, aplikace, prezentace uživateli)	integrace aplikací
Programové prostředky	nižší programovací jazyky	vyšší programovací jazyky – například COBOL	relační databáze a programovací nástroje SQL – například Oracle, Informix	programovací prostředí JAVA a objektové databáze	prostředky XML
Uživatelské podmínky	neinteraktivní	standardní obrazovky – textový režim	volně konfigurovatelné uživatelské obrazovky – Windows	multimediální aplikace, internetové prostředí a webové stránky	přístup přes mobilní zařízení
Funkčnost	plánování především materiálových požadavků	materiálové a kapacitní plánování a řízení výrobních zakázek	integrovaný informační systém řízení podniku	dodavatelsko odběratelské řetězce	e-business, CRM
Období zavádění	1975	1985	1992	1996	2000

Obrázek 1 – Vývoj ERP systémů 1975 – 2000. Zdroj [3]

Ten byl odstraněn [4] v osmdesátých letech, kdy byl vyvinut systém MRPII. Na přelomu osmdesátých a devadesátých let se již objevují první komplexní ERP systémy, jak je známe dnes. Postupně zahrnují většinu firemních procesů a mnoho z nich dříve vykonávaných lidmi plně automatizují. Organizace již nevyvíjí tyto systémy pouze pro svou vlastní potřebu, ale začínají je nabízet jako komerční zboží stavěné na zakázku nebo snadno upravitelné pro konkrétní podnik nebo průmyslové odvětví. Analýza firemních procesů a zavedení ERP systému přinášely podnikům zefektivnění chodu organizace a konkurenční výhodu a tak trh s ERP systémy spolu se stále klesajícími cenami IT technologií rostl až o desítky procent ročně.

1.2 ERP v České republice

U nás zaznamenáváme [3] nástup ERP po roce 1989, kdy se z centrálně plánované ekonomiky přechází k ekonomice tržní. Podniky byly postaveny před zásadní rozhodnutí. Na výběr měly tři varianty a to:

- rozvoj existujícího řešení,
- vývoj nového systému na míru,
- nákup hotové softwarové aplikace.

Každá varianta však měla svá pro i proti. První dvě varianty zcela přesně odpovídaly současným potřebám podniku, ale už nemusely odpovídat budoucím požadavkům. Celkové náklady jsou vyšší, výsledným produktem může být méně kvalitní systém a existuje tu i riziko negarantovaného konečného produktu a jeho dalšího vývoje. Třetí varianta nemusí úplně přesně splňovat požadavky uživatele, hrozí určité riziko zániku dodavatele, ale z dlouhodobého hlediska je finančně méně náročná a je tu zaručená funkčnost a další vývoj.

Přestože na počátku 90. let existovala v řadě případů preference v té době obvyklejších prvních dvou variant vycházejících z dotváření či vytváření vlastních SW produktů, postupně na trhu začala dominovat varianta třetí, tj. nákup hotových řešení IS. Začaly být ve větší míře implementovány importované zahraniční ERP systémy a postupně se objevila i řešení domácích SW firem.

1.3 Současné trendy ve vývoji ERP systémů

Mezi významné trendy [5] a nejen v případě ERP patří mobilita. Tento požadavek přichází hlavně ze strany uživatelů, kteří vyžadují nové informace ihned a v reálném čase. Většina firem je již dnes vybavena mobilními telefony s OS.

S tímto úzce souvisí uživatelské rozhraní (UI) a uživatelský prožitek (UX). Jedná se o ovládání podnikové aplikace pomocí smartphonů, dotykových displejů. V ERP systémech jde v podstatě o změnu přirovnatelnou při přechodu ze systému DOS na Windows.

Dalším trendem, který je ve srovnání situace před 10 až 20 lety opačný, je dvouvrstvá ERP strategie, oblíbená hlavně u firem s nadnárodní strukturou. Tzn. Mít jeden primární systém na centrále a svým pobočkám umožnit používání řešení, které je lokalizované s místní podporou, kde je rychlejší nasazení nových funkcionalit. Tato varianta je I finančně méně náročná.

Do ERP se postupně začaly integrovat sociální sítě pro lepší firemní komunikaci přes např. Facebook, Twitter. Tím se napomáhá ke zkvalitnění produktových vývojů, průzkumů trhů či marketingu. Jen čas ukáže, zda toto propojení má reálný přínos pro firmy, nebo je to jen módní trend.

Trendy v oblasti ERP, o kterých se mluví, jsou nejčastěji technologické. **SaaS** (Software as a Service – software jako služba) a **SOA** (service - oriented architecture – architektura orientovaná na službu). Hovoří se o spojení s datovými a výpočetními cloudy. Ne každá firma si může dovolit vlastní systém a ne každá firma má kapacitu starat se o něj. Propojení ERP s mobilní sítí má umožňovat rychle rozhodovat a mít informace vždy a všude k dispozici.

1.4 Příklady výrobců ERP

Zde je uvedeno několik známých dodavatelů ERP systémů. [6]

Asseco Solutions

Člen nadnárodní ICT skupiny ASSECO, se zabývá vývojem, implementací a podporou podnikových informačních systémů Helios. Produkty společnosti Asseco Solutions pokrývají nejširší spektrum potřeb podniků.

Epicor Software Corporation

Je dodavatelem informačních systémů pro střední firmy na celém světě. Široké spektrum dodávaných nástrojů pokrývá řízení podnikových procesů v oblasti financí, nákupu, prodeje, výroby, zakázek, řízení servisních činností, CRM, personalistiky a dále v oblasti reportovacích nástrojů.

Společnost J.K.R.

Je s ERP produkty třídy BYZNYS předním českým dodavatelem podnikových informačních systémů. Podnikové informační systémy třídy BYZNYS, které mimo jiné obsahují moduly CRM, Workflow, Business Intelligence, Projektové řízení, využívá v Česku více než 1 200 instalací v nejrůznějších oblastech podnikání.

Společnost Control spol. s r.o.

Byla založena v roce 1994 a hlavním zaměřením firmy byly již od jejího vzniku vývojové aktivity a implementace podnikového ERP řešení. V současné době tento IS znají zákazníci v České repiblice, na Slovensku, v Polsku a Maďarsku pod označením Dialog

3000S. Všeobecně je o firmě Control známo, že se specializuje na řízení výroby, logistiku a ekonomické modely řízení firem.

ESO9 intranet a.s.

Je přední dodavatel informačních technologií v České republice i na Slovensku. Nabízí rovněž řešení přizpůsobené na legislativní podmínky Maďarska či Polska. ESO9 intranet a.s. poskytuje zákazníkům silné zázemí v oblasti vývoje, implementace a služeb souvisejících s vlastním informačním systémem ESO9. Také jako jedna z mála firem nabízí k volnému prodeji samotnou technologii pro vývoj vlastních aplikací.

SAP

Je největším světovým dodavatelem softwaru pro informační systémy podniků a organizací všech velikostí. Na českém trhu působí společnost SAP od roku 1992, její zákazníci jsou z řad finančních institucí, státní správy, samosprávy, menší a střední firmy, ale i velké společnosti a organizace.

Společnost ABRA Software a.s.

Je česká společnost, která už 17 let poskytuje české ERP systémy pro řízení firemních procesů.

2 Přístup Java aplikací k databázovým systémům Oracle (JDBC)

Java Database Connectivity [7] [11**]** je API pro programátory programující v jazyce Java. Základní přístup k databázím je realizován právě pomocí tohoto rozhraní. JDBC je součástí Javy SE od verze JDK 1.1 dne 19. února 1997. JDBC je inspirováno standardem ODBC od firmy Microsoft. Protože je JDBC přímo součástí Java API, téměř všichni velcí účastníci trhu informačních technologií ho podporují. Třídy knihoven JDBC se dají najít v balíčcích *java.sql* a *javax.sql* v knihovně jazyka Java.

2.1 Fungování JDBC

1. Zavádění ovladače JDBC
2. Vytvoření připojení k DBMS
3. Spuštění dotazu
4. Zpracování výsledku dotazu
5. Odpojení od DBMS

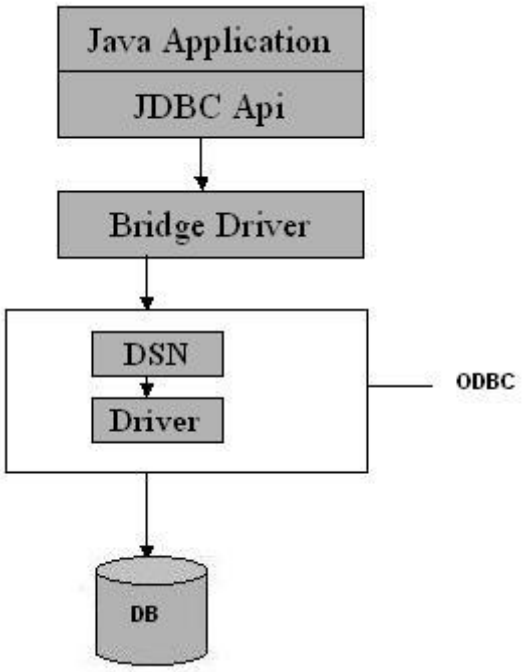
2.2 Ovladač JDBC

Při požadování přístupu [7][8] ke konkrétnímu databázovému serveru je potřeba JDBC ovladač typicky poskytovaný tvůrcem databázového systému. Je to základní koncept využívání JDBC, neboť ovladač překládá své funkce do nativních volání konkrétní databáze. Rozhraní ovladače se nachází v balíčku `java.sql`. V balíčku je definováno rozhraní Driver, s jehož pomocí lze vytvořit databázové připojení.

Typy JDBC ovladačů:

Typ 1

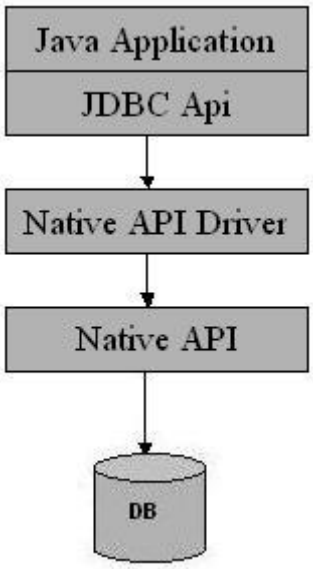
Využívá lokální ODBC ovladač, k němuž přistupuje pomocí *„JDBC-ODBC bridge.“* ODBC bývá zpravidla dodáván se systémy Windows. Za výhodu lze považovat fakt, že jedním ovladačem se lze připojit do všech databází, které jsou dostupné pomocí rozhraní ODBC. Nevýhodou je nutnost nainstalování a nastavení lokálního ODBC ovladače. JDBC ovladače tohoto typu se požívají hlavně pro testování v prostředí Windows a pro internetové/intranetové aplikace využívající JSP/Servlety.



Obrázek 2 - Princip fungování JDBC ovladače typu 1. Zdroj [9]

Typ 2

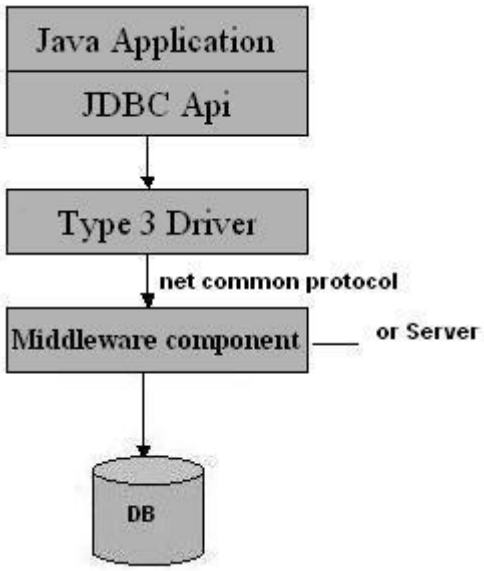
Výhodou je větší výkonnost než u předchozího typu. Nicméně nevýhody zůstávají stejné, protože tento typ ovladače funguje tak, že překládá požadavky JDBC do formy, které rozumí jiný ovladač nainstalovaný v počítači, a který je určen pro jeden konkrétní typ databáze. Obsahuje jak kód napsaný v jazyce Java, tak i nativní kód.



Obrázek 3 - Princip fungování JDBC ovladače typu 2. Zdroj [9]

Typ 3

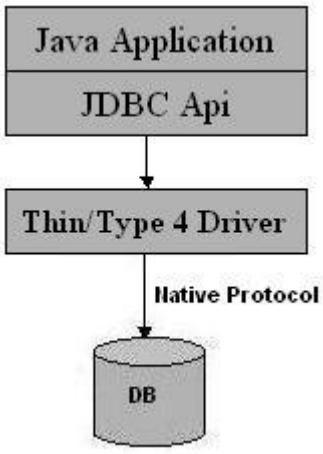
Založen pouze na Javě a JDBC a nepoužívá se zde už nativní kód pro ovladač. Ovladač komunikuje pomocí síťového protokolu (konkrétně síťové vrstvy) s centrálním serverem, který je dále připojen k databázi. Tento server se stará o překlad dotazů do formátu, který je srozumitelný pro konkrétní typy databází. Nutnost existence tohoto serveru lze považovat za nevýhodu při používání JDBC ovladače tohoto typu. Za výhodu lze považovat, že server může konvertovat dotazy do forem pro různé databázové produkty, čímž se pak tváří pro aplikační programátory jako jedna databáze.



Obrázek 4 - Princip fungování JDBC ovladače typu 3. Zdroj [9]

Typ 4

Typ ovladače kompletně napsaný v jazyce Java, což mu zajišťuje snadnou přenositelnost a snadné použití. Za nevýhodu lze považovat nutnost použití různých ovladačů pro různé databáze, protože tento typ ovladače JDBC komunikuje pomocí protokolu, který musí být specifický ke konkrétnímu typu databáze.



Obrázek 5 - Princip fungování JDBC ovladače typu 4. Zdroj [9]

2.3 Objekty JDBC

Všechny objekty JDBC API lze rozdělit do následujících 6 skupin:

1. Objekt správce ovladačů.
2. Objekty připojení.
3. Objekty příkazů.
4. Objekty výsledků.
5. Objekty metadat.
6. Objekty vyjímek.

2.4 Registrace JDBC ovladače

Registrace [7] je vázána na třídu `DriveManager`, která obsahuje metody pro práci s JDBC. Dá se říci, že `DriveManager` je vrstva mezi aplikačním kódem a JDBC ovladačem. Registrace probíhá ve chvíli, kdy je zavolána metoda `forName()`. Ovladač, který je předán jako parametr metody `forName()`, pak zavolá metodu `registerDriver()` ve třídě `DriveManager`. Toto se ovšem děje bez vědomí aplikačního programátora a starají se o to dodavatelé ovladače. Tímto je ovladač registrován a poskytován pro práci s databází.

```
try{
      Class.forName("oracle.jdbc.driver.OracleDriver");
}catch(ClassNotFoundException e){
      //kód pro obsluhu vyjímky
}
```

Od verze Java SE 6 (vydána 11. 12. 2006) již není nutná tato explicitní registrace ovladače, ale je nahrazena implicitní registrací na základě prvního parametru metody getConnection().

2.5 Připojení k databázi

Po úspěšné registraci [8] ovladače je možno vytvořit připojení k databázi. To se provádí metodou `getConnection()`, která vrací rozhraní typu `Connection`. Pomocí tohoto rozhraní pak komunikuje program a ovladač. Při navázání spojení aplikačního kódu s databází testuje `DriveManager` jednotlivé registrované ovladače a první úspěšné spojení vrátí. Pokud není ani jeden test úspěšný tak vyvolá `DriveManager` vyjímku.

```
try{
      String adresa = "adresa";
      pripojeni = DriverManager.getConnection(adresa, login,
      heslo);
}catch(SQLException e){
      //kód pro obsluhu vyjímky
```

```
}
```

2.6 Provádění databázových operací - Statement

Komunikace [8] probíhá pomocí třídy Statement. Po úspěšném připojení do databáze může instance třídy `Connection` vytvořit instanci Statement, a to pomocí metody `createStatement()`. Poté lze už začít provádět datové operace. Nutno dodat, že třída implementuje rozhraní `java.sql.Statement`. Pro posílání příkazů a dotazů do databáze se nejčastěji používají tři základní metody:

1. `executeUpdate()`
2. `executeQuery()`
3. `execute()`

Základní rozdíl je ve vracených výsledcích těchto metod. Metoda `executeUpdate()` vrací číslo typu `Integer`, které reprezentuje počet ovlivněných řádků SQL dotazem. Proto se tato metoda používá pro příkazy INSERT, UPDATE, DELETE atd. Naopak pokud chceme použít dotaz příkazem SELECT, musíme použít metodu `executeQuery()`, protože ta vrací výsledky dotazu. Tyto výsledky vrací ve formě instance třídy `ResultSet`. Metoda `execute()` lze pak použít pro všechny databázové operace.

2.7 Komunikace s databází – PreparedStatement

Je potomkem [10] třídy Statement a zaručuje bezpečnější a snadnější sestavování SQL dotazů. Používá se, hlavně pokud chceme do databáze zasílat podobné dotazy lišící se například pouze v hodnotách parametrů. Protože každý dotaz musí být zkompilován JDBC ovladačem, třída Statement by byla v tomto případě značně nepraktická a nevýkonná. Fungování `PreparedStatement` je založeno na tzv. **substitučních parametrech**. Při vytváření instance třídy `PreparedStatement` pomocí třídy `Connection` a její metody `prepareStatement()` zadáváme jako parametr SQL dotaz, který bude tento `PreparedStatement` vykonávat. Avšak místo konkrétních hodnot atributů se zapisují otazníky (placeholder), to jsou právě výše zmíněné substituční parametry. Tyto parametry jsou pak nahrazovány konkrétními hodnotami pomocí metod `set*()` třídy `PreparedStatement`, kde * je datový typ parametru např. `setInt()` nebo `setString()`. Tyto metody mají dva parametry. První udává pořadí substitučního parametru v dotazu – čísluje se od 1. A druhý konkrétní hodnotu, kterou má být substituční parametr nahrazen. Výhodou tedy oproti třídě Statement je menší režie parametrů (např. uvozovky, správné formáty času atd.) a pouze jedna kompilace SQL dotazu.

```
try{
//Posílání dotazů pomocí třídy Statement
Statement statement = pripojeni.createStatement();

String dotazDB = "UPDATE zakaznici SET jmeno = 'Petr' WHERE id_zakaznika
```

```
= 1";
statement.executeUpdate(dotazDB);

dotazDB = "UPDATE zakaznici SET jmeno = 'Jan' WHERE id_zakaznika = 2";
statement.executeUpdate(dotazDB);

dotazDB = "UPDATE zakaznici SET jmeno = 'Karel' WHERE id_zakaznika = 3";
statement.executeUpdate(dotazDB);

//Posílání stejných dotazů pomocí třídy PreparedStatement
PreparedStatement preparedStatement = pripojeni.prepareStatement(
      "UPDATE zakaznici SET jmeno = ? WHERE id_zakazanika = ?");

preparedStatement.setString(1, "Petr");
preparedStatement.setInt(2, 1);
preparedStatement.executeUpdate();

preparedStatement.setString(1, "Jan");
preparedStatement.setInt(2, 2);
preparedStatement.executeUpdate();

preparedStatement.setString(1, "Karel");
preparedStatement.setInt(2, 3);
preparedStatement.executeUpdate();
}catch(SQLException e){
      //kód pro obsluhu vyjímky
}
```

2.8 Zpracování výsledků SQL dotazů

Výše bylo [8] zmíněno, že instance třídy `ResultSet` obsahuje výsledku dotazu, které nám vrátí dotaz odeslaný do databáze metodou `executeQuery()` ve třídě Statement nebo `PreparedStatement`. K pohybu mezi jednotlivými řádky výsledku nám slouží primárně metoda `next()`. Tato metoda nepřijímá žádné argumenty a vrací hodnotu typu *boolean*. Pokud vrácená hodnota je *true*, pak se metodě `next()` podařilo přejít na další řádek výsledku dotazu. Samozřejmě *false* metoda vrací, pokud žádný další řádek neexistuje. Další metody pro pohyb v řádcích jsou například:

1. `first()` – vrátí ukazatel na řádek na první řádek výsledku
2. `last()` – nastaví ukazatel na poslední řádek výsledku

Co se týče pohybu ve výsledku dotazů, je třeba ještě zmínit, že možnosti pohybu lze nastavit při vytváření třídy Statement. Metoda `createStatement()` totiž může přijímat žádný, dva nebo tři argumenty. Možnosti nastavení Statement při vytváření jsou:

Typ

1. **ResultSet.TYPE_FORWARD_ONLY** – kurzor se může pohybovat pouze dopředu
2. **ResultSet.TYPE_SCROLL_INSENSITIVE** – rolovací ale obecně necitlivý na změny provedené ostatními

3. **ResultSet.TYPE_SCROLL_SENSITIVE** - rolovací a obecně citlivý na změny provedené ostatními

Možnost měnit

1. **ResultSet.CONCUR_READ_ONLY** – pouze pro čtení
2. **ResultSet.CONCUR_UPDATABLE** – možnost měnit

Možnost držení záznamů

1. **ResultSet.HOLD_CURSORS_OVER_COMMIT** - podržení kurzorů i přes commit
2. **ResultSet.CLOSE_CURSORS_AT_COMMIT** - zavření kurzorů při commit

Nyní při možnosti pohybu v záznamech už chybí jen načíst konkrétní data. K tomu slouží metody `get*()`, kde * se opět nahrazuje datovým typem (např. `getInt()`, `getString`). Za zmínku stojí metoda `getObject()`, která vrací instanci typu *Object* a používá se, pokud neznáme datový typ sloupce. Protože dotaz poslaný do databáze, může vracet výsledek s více sloupci, je ještě nutné upřesnit, z kterého sloupce chceme data získat. K tomu slouží argumenty metod `get*()`. Metody přijímají vždy jeden argument. Buď celé číslo reprezentující index sloupce, nebo argument typu *String*, reprezentující název sloupce.

```
ResultSet rs = statement.executeQuery(dotazDB);
while(rs.next()){
      //kód pro zpracování jednotlivých řádků dotazu
      String dataString = rs.getString(1);
      Int dataInt = rs.getInt(2);
      System.out.printl(dataString + ", " + dataInt);
}
```

3 Aplikace BP

V této kapitole bude představen cíl této práce, a to aplikace pro správu a plánování zdrojů v rámci betonářského podniku a demonstrace jednoduchého ERP sytému. Ve fiktivním betonářském podniku bylo stanoveno několik problému, které by takový podnik mohl mít a které má aplikace pomoci vyřešit (např. několik různých uložišť dat, neefektivní možnosti plánování i správa některých entit, absence funkcí pro reporty či jinou zpětnou kontrolu).

Aplikace je navržena jako standardní GUI aplikace a splňuje následující požadavky:

1. Správa všech hmotných i nehmotných zdrojů.
2. Plánování zdrojů pro snadnější chod podniku.
3. Tvorba reportů a export jejich výsledků pro zpětnou kontrolu chodu podniku.
4. Přehlednost a jednoduchost ovládání aplikace.
5. Bezproblémová a bezchybná komunikace s databází a daty v ní uložené.

3.1 Databáze

Všechna data, se kterými aplikace pracuje, jsou uložena v databázi. Pro tuto práci byla použita databáze Oracle Database verze 11g pro systém Windows 7. Pro vytváření entit, jejich atributů a celkovou správu byl použit program Oracle SQL Developer 1.5.5, který je možno nainstalovat společně s databází. Do databázového návrhu byly zahrnuty pouze entity a atributy, které je nutné vždy spravovat a plánovat a jsou nezbytné pro základní chod aplikace. Atributům byly přiřazeny datové typy dle potřeby, stejně jako příslušná omezení. Při implementaci databáze do konkrétního podniku by bylo nutné upravit atributy tabulek dle specifických požadavků dané firmy.

3.1.1 Seznam a popis entit a jejich atributů

Celý ERD diagram znázorňující strukturu databáze nebyl z důvodu jeho velikosti do této práce zahrnut. Nicméně je dodán spolu se zdrojovými kódy, a to jako soubor ***modelDB.txp***. Návrh databáze byl vytvořen v programu ToadDataModeler 3.3. Ten byl zvolen pro jeho předchozí znalost a pro možnost vygenerování kódu na vytvoření všech objektů návrhu.

3.1.1.1 Tabulka prava

V této tabulce je uložen seznam práv, která mohou být v aplikaci realizovány. Tyto práva jsou přiřazována pracovním pozicím pomocí cizího klíče.

Tabulka 1 - Přehled atributů tabulky `prava`. Zdroj vlastní

Název	Datový typ	Omezení	Popis
Id_prava	Number(3,0)	Primární klíč	Jednoznačný identifikátor
Popis	Varchar2(30)	Not null	Popis, co konkrétní právo dovoluje a co zakazuje

3.1.1.2 Tabulka pracovniPozice

V této tabulce jsou uloženy pracovní pozice, které jsou přiřazovány uživatelům aplikace.

Tabulka 2 - Přehled atributů tabulky `pracovniPozice`. Zdroj vlastní

Název	Datový typ	Omezení	Popis
Id_pozice	Number(3,0)	Primární klíč	Jednoznačný identifikátor
Id_prava	Number(3,0)	Cizí klíč	Určuje, k čemu uživatel s konkrétní pracovní pozicí má přístup
Nazev	Varchar2(30)	Not null	Název pracovní pozice, pro snazší práci s pozicemi
Plat	Number(10,2)	Not null	Určuje základní mzdu každé pozice

3.1.1.3 Tabulka uzivatel

Uchovává všechny nezbytné informace o uživatelích aplikace. V této bakalářské práci je každý uživatel zaměstnanec.

Tabulka 3 - Přehled atributů tabulky `uzivatel`. Zdroj vlastní

Název	Datový typ	Omezení	Popis
Id_uzivatel	Number (5, 0)	Primární klíč	Jednoznačný identifikátor
Id_pozice	Number (3, 0)	Cizí klíč	Přiřazuje každému uživateli pracovní pozici ve firmě
Jmeno	Varchar2 (30)	Not null	Jméno uživatele
Prijmeni	Varchar2 (30)	Not null	Příjmení uživatele
Ulice	Varchar2 (30)		Adresa
Cislo_popisne	Number (15, 0)		Adresa
Mesto	Varchar2 (30)		Adresa
PSC	Number (10, 0)		Adresa
Telefon	Number (15, 0)	Not null	Telefonní číslo uživatele
Uzivatelske_jmeno	Varchar2 (20)	Not null, Unique	Jméno, pod kterým se uživatel přihlašuje do aplikace
Heslo	Varchar2 (20)	Not null	Druhý atribut nutný k přihlášení do aplikace

3.1.1.4 Tabulka objednavkyMaterialu

Uchovává všechny vytvořené objednávky, kde si uživatelé (zaměstnanci) objednávají nějaký materiál od dodavatelů.

Tabulka 4 - Přehled atributů tabulky `objednavkyMaterialu`. Zdroj vlastní

Název	Datový typ	Omezení	Popis
Id_objednavky	Number(10, 0)	Primární klíč	Jednoznačný identifikátor
Id_uzivatel	Number(5, 0)	Cizí klíč	Uchovává informace o tom, který uživatel objednávku vytvořil
Datum_objednavky	Date	Not null	Datum, kdy byla objednávka vytvořena, má pouze informativní charakter
Prevzato	Date		Pokud *null*, objednávka ještě nebyla převzata. Při potvrzení převzetí v aplikaci dojde ke vložení aktuálního data
Zaplaceno	Date		Podobné jako atribut `prevzato`. Ale jak název napovídá, atribut informuje o zaplacení objednávky

3.1.1.5 Tabulka dodavatele

Informace o dodavatelích materiálu.

Tabulka 5 - Přehled atributů tabulky `dodavatele`. Zdroj vlastní

Název	Datový typ	Omezení	Popis
Id_dodavatele	Number(5,0)	Primární klíč	Jednoznačný identifikátor
Firma	Varchar2(30)	Not null	Název firmy, která dodává zboží
Telefon	Number(15,0)	Not null	Předpokládá se, že objednávky se budou domlouvat telefonicky, takže jeden z nejdůležitějších atributů
Cislo_uctu	Varchar2(30)	Not null	Je nutné udržovat aktuální informace, kam se mají objednávky platit
Email	Varchar2(30)		Doplňující informace kvůli další možnosti komunikace

3.1.1.6 Tabulka materialy

Seznam všech materiálů, se kterými firma obchoduje nebo které potřebuje na výrobu.

Tabulka 6 - Přehled atributů tabulky `materialy`. Zdroj vlastní

Název	Datový typ	Omezení	Popis
Id_material	Number (5, 0)	Primární klíč	Jednoznačný identifikátor
Id_dodavatele	Number (5, 0)	Cizí klíč	Atribut udává, který dodavatel nám dodává materiál
Nazev	Varchar2(30)	Not null	Název materiálu kvůli snazší práci
Cena_nakupni	Number (10, 2)	Not null	Cena, za kterou se materiál nakupuje od dodavatele
Cena_prodejni	Number (10, 2)	Not null	Cena, za kterou se prodává materiál zákazníkovi
Mnozstvi_sklad	Number (10, 3)	Not null	Aktuální množství materiálu na skladě

3.1.1.7 Tabulka vyrobky

Informace o výrobcích, které lze prodávat zákazníkům.

Tabulka 7 - Přehled atributů tabulky `vyrobky`. Zdroj vlastní

Název	Datový typ	Omezení	Popis
Id_vyrobek	Number (5, 0)	Primární klíč	Jednoznačný identifikátor
Cena	Number (10, 2)	Not null	Cena, za kterou se výrobek prodává zákazníkovi
Nazev	Varchar2 (30)	Not null	Název výrobku, pro snazší práci
Doba_vyroby	Number (10, 1)		Informace určená pro plánování

3.1.1.8 Tabulka vyroba

Informace o složení každého výrobku.

Tabulka 8 - Přehled atributů tabulky `vyroba`. Zdroj vlastní

Název	Datový typ	Omezení	Popis
Id_vyrobek	Number (5, 0)	Primární cizí klíč	Část jedinečného identifikátoru
Id_material	Number (5, 0)	Primární cizí klíč	Část jedinečného identifikátoru
Mnozstvi	Number (8, 2)	Not null	Informace, kolik konkrétní výrobek obsahuje konkrétního materiálu

3.1.1.9 Tabulka objednavkyZakaznici

Informace o objednávkách zákazníků.

Tabulka 9 - Přehled atributů tabulky `objednavkyZakazniku`. Zdroj vlastní

Název	Datový typ	Omezení	Popis
Id_objednavky	Number (10,0)	Primární klíč	Jednoznačný identifikátor
Id_zakaznik	Number (5, 0)	Cizí klíč	Informace o zákazníkovi, který vytvořil objednávku
Datum_objednavky	Date	Not null	Pouze informativní atribut kdy byla objednávka vytvořena
Datum_vyzvednuti	Date	Not null	Atribut sloužící pro plánování objednávek
Doba_splatnosti	Number (3, 0)		Doba, do kdy by měla být objednávka zaplacena
Vyzvednuto	Date		Pokud *null*, objednávka zatím nebyla vyzvednuta
Zaplaceno	Date		Pokud *null*, objednávka zatím nebyla zaplacena
Sleva	Number (5, 2)		Nepovinný atribut, umožňuje slevit z celkové ceny objednávky
DPH	Number (5, 2)	Not null	Udává jaká daň je použita na zboží

3.1.1.10 Tabulka zakaznici

Tabulka ukládá všechny potřebné informace o zákaznících firmy.

Tabulka 10 - Přehled atributů tabulky `zakaznici`. Zdroj vlastní

Název	Datový typ	Omezení	Popis
Id_zakaznik	Number (5, 0)	Primární klíč	Jednoznačný identifikátor
Firma	Varchar2(30)		Pouze pokud zákazník je firma a ne živnostník
Jmeno	Varchar2(30)	Not null	Jméno zákazníka
Prijmeni	Varchar2(30)	Not null	Příjmení zákazníka
Telefon	Number (15, 0)	Not null	Telefon pro hlavní komunikaci se zákazníkem
Název	Datový typ	Omezení	Popis
Cislo_uctu	Varchar2 (30)	Not null	
Ulice	Varchar2 (30)		Adresa
Cislo_popisne	Number (15, 0)		Adresa
Mesto	Varchar2 (30)		Adresa
PSC	Number (10, 0)		Adresa

3.1.1.11 Tabulka polozkyObjednavky, polozkyObjednavkyM, polozkyObjednavkyV

Tabulky vznikly kvůli odstranění možnosti objednat na každou objednávku pouze jedno zboží. Uchovávají tedy informace, jaké konkrétní zboží jsou objednaná v každé

objednávce, ať už se jedná o objednávky pro dodavatele nebo objednávky zákazníků. Všechny jsou velmi podobné, takže struktura tabulky bude ukázána jen na jednom příkladu.

Tabulka 11 - Přehled atributů tabulky `polozkyObjednavky`. Zdroj vlastní

Název	Datový typ	Omezení	Popis
Id_material	Number (5, 0)	Primární cizí klíč	Část jedinečného identifikátoru
Id_objednavky	Number (10, 0)	Primární cizí klíč	Část jedinečného identifikátoru
Mnozstvi	Number (5, 2)	Not null	Atribut udávající množství objednaného materiálu

3.1.2 Automatická inkrementace primárních klíčů

Protože ruční zadávání primárních klíčů by bylo velmi nepraktické a nesprávné, bylo nutné zajistit automatickou inkrementaci primárních klíčů při každém vložení nového řádku do tabulky. V této práci byla použita varianta sekvence a triggeru.

3.1.2.1 Sekvence

Speciální databázový objekt, který pomocí funkce *nextval* vrací další číslo z předem nadefinované řady.

```
CREATE SEQUENCE Seq_zakaznici
INCREMENT BY 1
START WITH 1
NOMAXVALUE
MINVALUE 0;
```

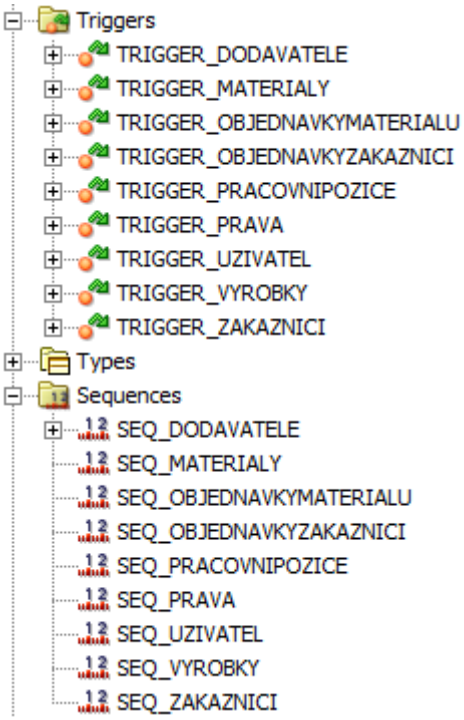
Při uvažování vytvoření sekvence na uvedeném kódu nahoře, bychom mohli vkládat do tabulek hodnoty primárních klíčů pomocí příkazu *seq_zakaznici.nextval*.

3.1.2.2 Trigger

Spouštěč, který provede příkaz nebo skupinu příkazů při výskytu události na kterou je nastaven.

```
CREATE OR REPLACE TRIGGER trigger_zakaznici
BEFORE INSERT ON zakaznici
FOR EACH ROW
BEGIN
SELECT seq_zakanici.nextval INTO :new.id_zakaznik FROM dual;
END;
```

Na uvedeném kódu je vidět vytvoření triggeru s názvem *trigger_zakaznici*. Tento trigger před každým vložením řádku do tabulky zákazníci, vloží pomocí sekvence *seq_zakaznici* další hodnotu z řady do atributu *id_zakaznik*.



Obrázek 6 - Přehled všech použitých sekvencí a triggerů v této práci. Zdroj vlastní

3.2 Aplikace

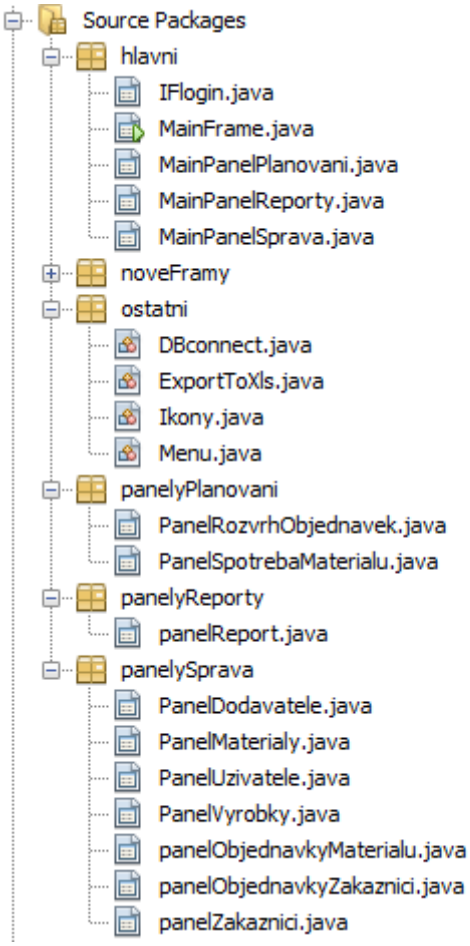
Aplikace byla napsána v programovacím jazyce Java SE verze 7. Jako vývojové prostředí bylo použito vývojové prostředí Netbeans IDE 7.4. Tento program byl zvolen kvůli možnosti snadného vytvoření GUI aplikace. Aplikace umožňuje správu všech hmotných i lidských zdrojů, plánování a tvorbu reportů, které je možné exportovat pro další zpracování. Aplikace pracuje jen se základními zdroji, které jsou nezbytné pro každou firmu fungující v oblasti betonářství, a byla navrhnuta pro menší firmu s pouze jednou pobočkou. Pro praktické použití by bylo nejspíše nutné, aplikaci přizpůsobit na míru případnému zájemci.

3.2.1 Struktura aplikace

Protože aplikace pracuje s větším počtem tříd, bylo by nepraktické a nepřehledné mít všechny tyto třídy pohromadě. K tomu nám slouží balíčky (packages). V této práci bylo celkem použito 6 balíčků, které jsou vidět na obrázku [Obrázek 7].

Tabulka 12 - Přehled balíčků a jejich popis. Zdroj vlastní

Název	Popis
hlavni	Obsahuje základní třídy pro spuštění aplikace.
noveFramy	Obsahuje třídy vytvářející okna při vytváření, editaci či výpisu záznamů.
ostatni	Obsahuje doplňkové třídy aplikace.
panelyPlanovani	Obsahuje třídy panelů pro plánování.
panelyReporty	Obsahuje třídy panelů pro reporty.
panelySprava	Obsahuje třídy panelů pro správu.



Obrázek 7 - Seznam balíčků, většinou s jejich obsahem. Zdroj vlastní

3.2.2 Přihlášení a komunikace s databází

Komunikace s databází je základní kámen celé aplikace. V této práci se používá pro komunikaci s databází ovladač JDBC verze5. Nyní zde bude podrobně popsáno, jakým způsobem tato komunikace funguje. Začátek komunikace začíná hned po zavolání metody `main()` a následném vytvoření okna `MainFrame`, které ve svém konstruktoru vytváří instanci třídy `IFlogin`. Tato třída je potomkem třídy JInternalFrame a má pouze jediný úkol a to od uživatele získat přihlašovací údaje potřebné pro připojení k databázi. Důvodem k použití potomka třídy JInternalFrame byl fakt, že takto vytvořené okno nemůže svou lokací opustit třídu, která ho vytvořila. Bylo by nežádoucí, aby okno, kde se píší přihlašovací údaje, bylo mimo hlavní okno celé aplikace. S tímto souvisí první řádek v konstruktoru třídy `IFlogin`:

```
public JInternalFrame(String title, boolean resizable, boolean closable,
boolean maximizable, boolean iconifiable);
```

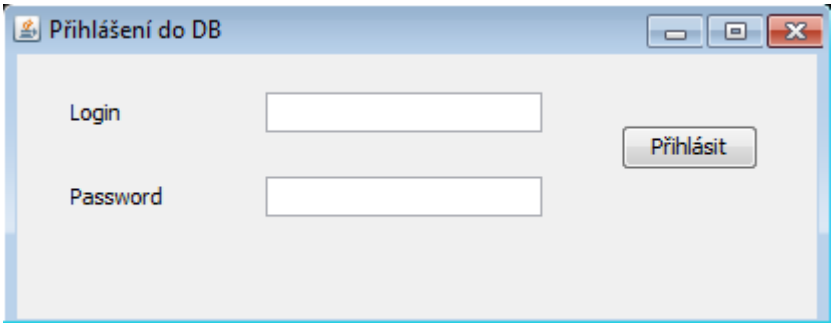
Konkrétní použití:

```
super("Přihlášení do DB", true, true, true, true);
```

Popis parametrů:

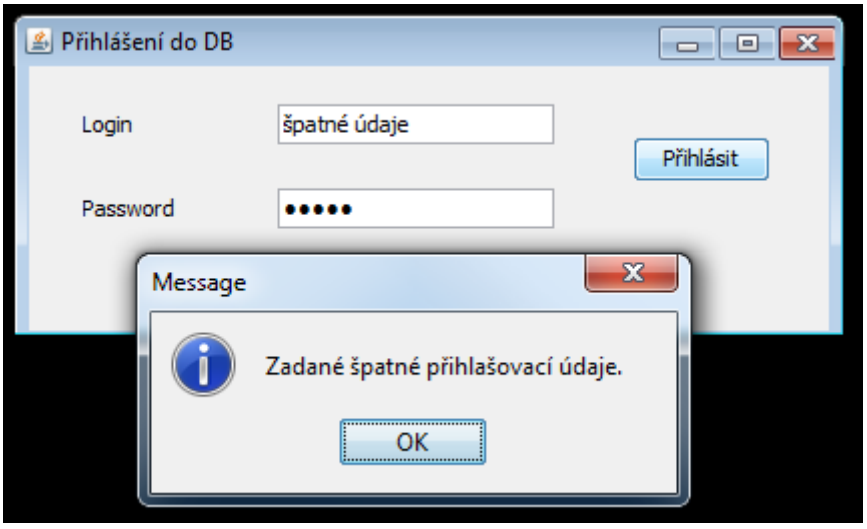
1. Titulek vytvořeného JInternalFramu,
2. Pokud true, možno měnit velikost vytvořeného okna,
3. Pokud true, možno zavírat vytvořené okno,
4. Pokud true, možno maximalizovat vytvořené okno,
5. Pokud true, možno minimalizovat vytvořené okno.

Na obrázku [Obrázek 8] je vidět jak vypadá vytvořená instance IFlogin.



Obrázek 8 - Přihlašovací InternalFrame do aplikace. Zdroj vlastní

Po zadání a potvrzení přihlašovacích údajů se naváže připojení s databází. K tomu už dochází ve třídě DBconnect metodou `vytvorPripojeni()`. Tato metoda se vždy úspěšně připojí a poté dochází k autentifikaci uživatele aplikace. Pokud se podaří najít uživatele se stejným přihlašovacím jménem i heslem, aplikace vyhodnotí zadané údaje od uživatele jako správné, v opačném případě zahlásí chybu, viz obrázek [Obrázek 9].



Obrázek 9 - Ukázka vypsání chyby při zadání špatných přihlašovacích údajů. Zdroj vlastní

Po úspěšném přihlášení je potřeba vytvořit instanci třídy Statement, aby bylo možné zasílat dotazy do databáze. K tomuto slouží metoda `vratStatement()`, která vrací vytvořenou instanci třídy Statement. Nutno podotknout, že tato metoda je statická a je možné ji zavolat

odkudkoli v celé aplikaci. Takže celá aplikace má pouze jednu instanci třídy Connection, ale každá komponenta pracující s databází má svou vlastní instanci třídy Statement.

```
public static Statement vratStatement() {
    try {
        Statement st;
        St = pripojeni.createStatement(
        ResultSet.TYPE_SCROLL_INSENSITIVE,
        ResultSet.CONCUR_UPDATABLE);
        return st;
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, e.getMessage());
    }
    return null;
}
```

3.2.3 Popis práv a pracovních pozic

Práva a pracovní pozice spolu úzce souvisí, jak je snadno pochopitelné z popisu použité databáze. Práva a pracovní pozice v této práci byly vytvořeny pouze pro demonstraci funkčnosti a možností aplikace. Stejně jako celá aplikace i tyto entity by v praktickém použití museli být upraveny na míru případnému zájemci.

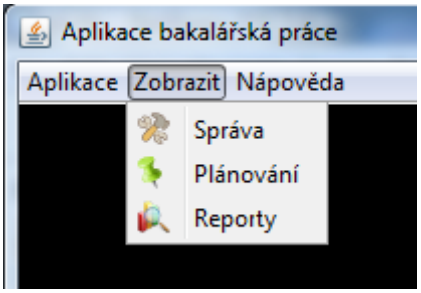
Seznam pracovních pozic s popisem jejich práv v aplikaci:

1. Administrátor – tato pracovní pozice má všechna práva a možnosti, které aplikace dovoluje,
2. Manažer – v aplikace má tato pozice stejné možnosti jako administrátor, kromě možnosti mazaní a editování ostatních uživatelů aplikace,
3. Účetní – už značně omezené možnosti hlavně v mazání (např. dodavatele, zákazníky atd.), pracovníci s touto pozicí už vůbec nemohou ani vidět informace o ostatních uživatelech aplikace,
4. Dělník – pouze možnosti nezbytně nutné pro vykonávání práce, nemožnost cokoliv mazat, vypisovat reporty nebo editovat jakékoliv zdroje.

Na obrázcích s ukázkami aplikace se v této práci vždy používá přihlášení jako administrátor, aby byly ukázány všechny funkce a možnosti aplikace.

3.2.4 Menu a možnosti aplikace

Menu je základním ovládacím prvkem aplikace. Umožňuje uživateli přepínat mezi panely pro správu, plánování a reporty. Dále umožňuje uživateli se odhlásit nebo ukončit celou aplikace. A poslední možností menu je vypsat základní informace o aplikaci. Pro sestavení menu se v této práci používají instance třech tříd – `JMenuBar`, `JMenu`, `MenuItem`. Po sestavení menu je potřeba ho přiřadit k hlavnímu oknu aplikace a to pomocí metody `setJMenuBar()`.



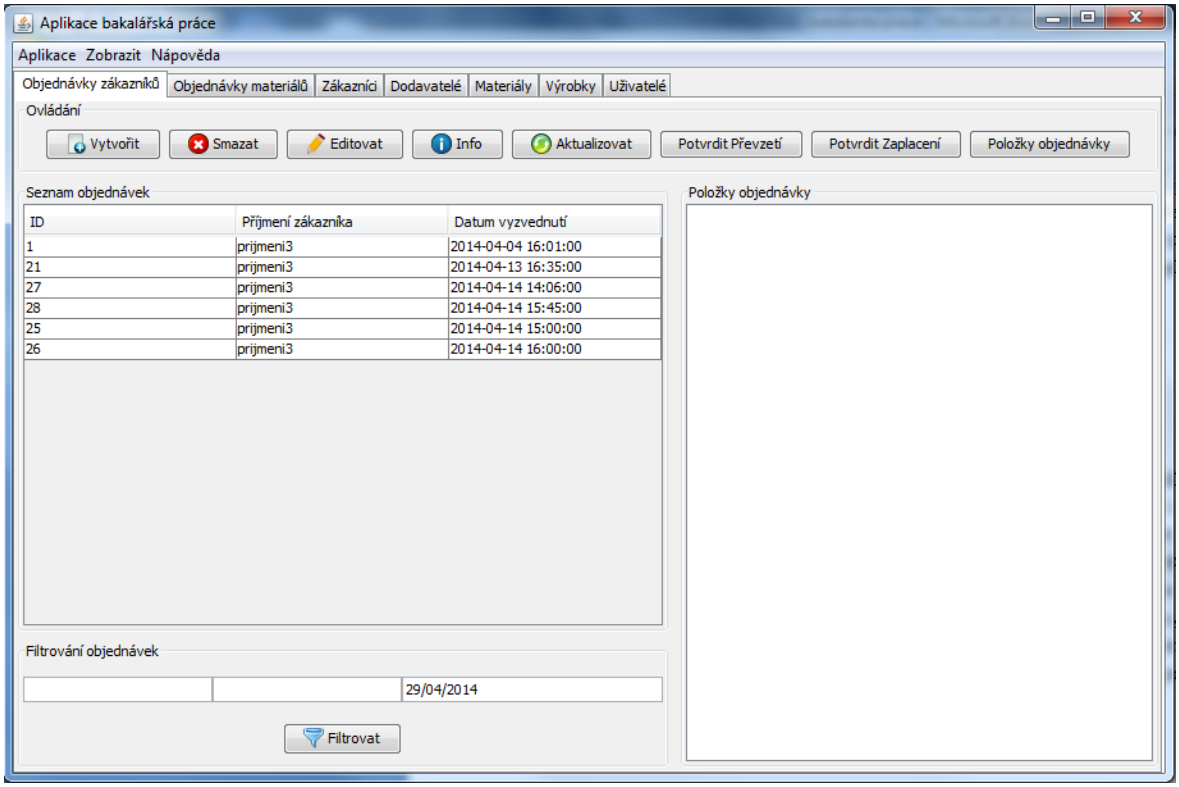
Obrázek 10 - Ukázka menu v aplikaci. Zdroj vlastní

Na obrázku [Obrázek 10] je ukázka menu v této aplikaci. Je z něj vidět, že `JMenuBar` obsahuje 3 instance třídy Menu (s popisy Aplikace, Zobrazit a Nápověda). Přiřazování se provádí metodou `add()`. A každá instance Menu pak obsahuje jednotlivé instance `JMenuItem`. Přiřazování `JMenu` i `JMenuItem` je závislé na právech přihlášeného uživatele. Pro správnou funkčnost je ale nutné nastavit položkám menu akce, které se mají vykonat při jejich stisknutí. V následující ukázce kódu je vidět nastavení akce pro ukončení aplikace po stisknutí příslušné položky v menu.

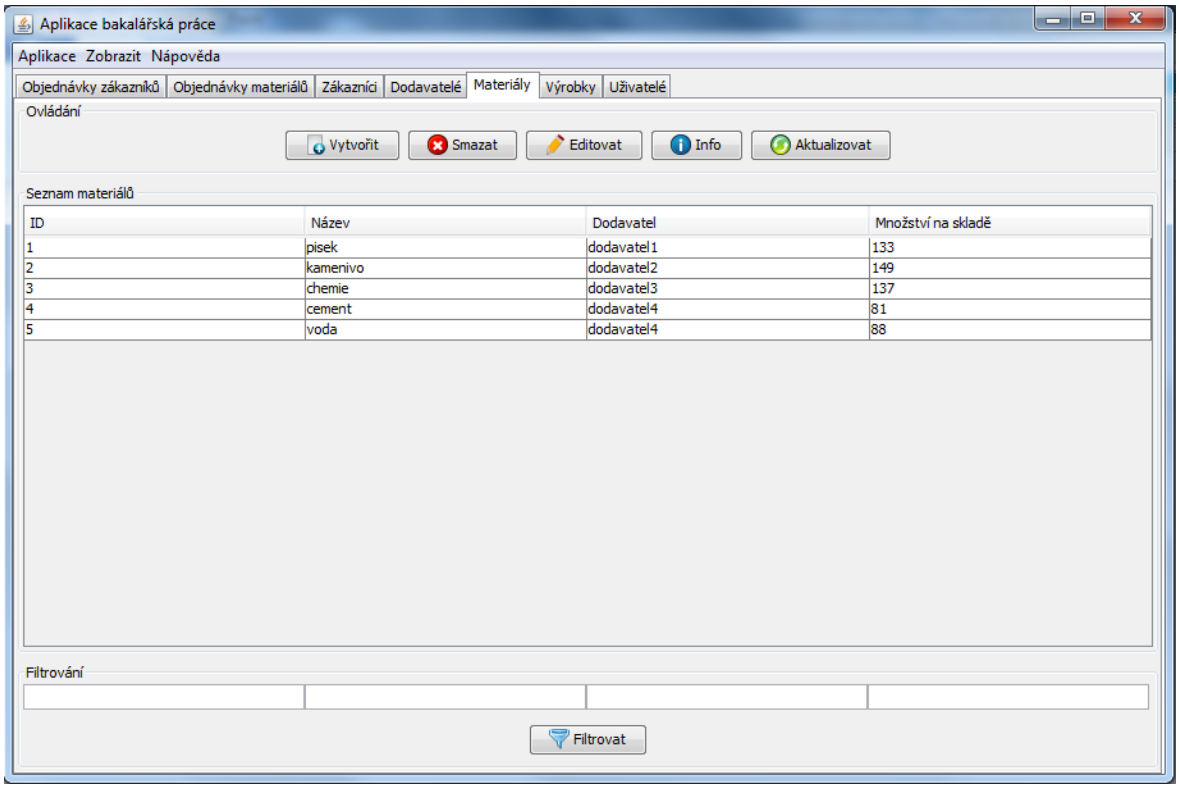
```
aplikaceKonec.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        System.exit(0);
    }
});
```

3.2.5 Správa

Správa hmotných a nehmotných zdrojů je nejdůležitější funkcionalitou celé aplikace. Bylo by možné, aby aplikace splňovala základní funkčnost bez možností plánování i reportů, ale nikoli bez možnosti spravování zdrojů. Pro možnost začít cokoliv spravovat je nutné v menu vybrat příslušnou funkci. Po tomto je vytvořen základní panel pro správu (instance třídy `MainPanelSprava`), který je přiřazen hlavnímu oknu aplikace. Na tomto panelu je jediná komponenta a to instance třídy `JTabbedPane`. Ta obsahuje metodu `add()`, pomocí které jsou přiřazovány panely pro jednotlivé zdroje. To je samozřejmě také závislé na právech přihlášeného uživatele. Na obrázcích [Obrázek 11] [Obrázek 12] jsou vidět ukázky panelů pro správu.



Obrázek 11 - Ukázka panelu pro správu objednávek zákazníků. Zdroj vlastní



Obrázek 12 - Ukázka panelu pro správu materiálů. Zdroj vlastní

3.2.5.1 Popis komponent v panelech pro správu

Základní komponentou každého panelu pro jednotlivé zdroje je tabulka s výpisem zdrojů uložených v databázi. Tato tabulka je reprezentovaná komponentou JTable. Nastavení tabulky v této práci probíhá formou modelu. Tento model je nejdříve nastaven tak, jak chceme, aby tabulka vypadala, a jaké obsahovala hodnoty, poté je tento model předán komponentě JTable pomocí metody `setModel()`. Vytváření a přiřazování modelu vypadá takto:

```
TableModel model = new AbstractTableModel(){};

jTableSeznam.setModel(model);
```

Ještě nutno dodat, že při vytváření anonymního typu odvozeného od třídy AbstractTableModel je nutné implementovat 3 metody. Bez toho není možné model vytvořit. Jedná se o metody `getRowCount()`, `getColumnCount()` a `getValueAt()` vracející počet řádků, sloupců a hodnot v každé buňce tabulky. Nadále byla použita již nepovinná metoda `getColumnName()`, která vrací popisy jednotlivých sloupců tabulky. Protože komponenty `JTable` jsou používány v celé aplikaci a jejich nastavování je vždy obdobné, je zde uveden na ukázku celý kód pro nastavení tabulky z obrázku [Obrázek 12] výše.

```
@Override
public int getRowCount() {
    return pocet;
}
@Override
public int getColumnCount() {
    return 4;
}
@Override
public String getColumnName(int column) {
    switch (column) {
        case 0:
            return "ID";
        case 1:
            return "Název";
        case 2:
            return "Dodavatel";
        case 3:
            return "Množství na skladě";
        default:
            return "Column";
    }
}
@Override
public Object getValueAt(int rowIndex, int columnIndex) {
    try {
        rs.absolute(rowIndex + 1);
        return rs.getString(columnIndex + 1);
    } catch (SQLException ex) {
    }
    return 0;
}
```

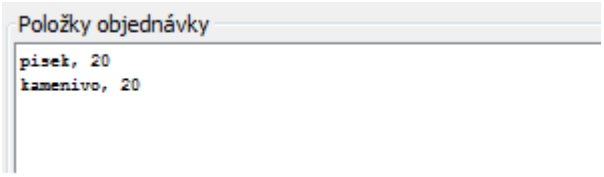
Dalšími komponentami, které jsou společné pro všechny panely, jsou ovládací tlačítka. Tyto tlačítka jsou reprezentovány komponentami `JButton`. Jejich seznam spolu s jejich funkcemi:

1.  Vytvořit – tlačítko určené k vytváření nových záznamů podle právě aktivního panelu. Samotné vytváření probíhá v instancích potomků `JFrame`, které jsou vytvořeny po stisku tohoto tlačítka a které obsahují formulář k doplnění informací,
2.  Smazat – smaže vybraný záznam jak z tabulky, tak z databáze. Pokud není žádný záznam vybrán, aplikace na to uživatele upozorní,
3.  Editovat – po stisku tlačítka se vytvoří stejná instance potomka `JFrame` jako po stisku tlačítka Vytvořit. Rozdíl je v tom, že formulář obsažený v tomto okně, je vyplněn informacemi z databáze o vybraném záznamu, které je možno měnit a znovu uložit. Aplikace opět kontroluje, zda je při stisku tlačítka nějaký záznam vybrán,
4.  Info – stejná funkcionalita jako tlačítko Editovat s rozdílem, že změny není možné nijak uložit. Slouží např. k porovnávání různých záznamů, kde nechceme nic měnit, a proto z hlediska bezpečnosti to aplikace neumožňuje,
5.  Aktualizuj – aktualizuje výpis záznamů v tabulce bez jakýchkoli filtrovacích kritérií,
6.  Filtrovat – filtruje výpis záznamů v tabulce. Pod každým sloupcem tabulky se nachází komponenta `JTextField`. Filtrování probíhá pomocí hodnot těchto komponent. Je tedy zřejmé, že aplikace umožňuje filtrovat pouze pomocí základních informací, které jsou uvedeny v tabulce. Nicméně je možné filtrovat pomocí více kritérií najednou. Dotaz do databáze se sestavuje podle toho, zda jsou jednotlivé `JTextFieldy` prázdné nebo ne.

Tlačítka, která se vyskytují pouze u panelů s objednávkami:

7. Potvrdit převzetí – protože objednávky bývají vytvořeny dopředu na určitý datum a čas, je nutné evidovat, které objednávky byly převzaty a které ne. A k tomu slouží právě toto tlačítko. Po jeho stisku se k vybranému záznamu zapíše do databáze do atributu `převzato` aktuální datum. Tím, že atribut už není null víme, že objednávka byla převzata,
8. Potvrdit zaplacení – objednávky, které nejsou zaplaceny hotově při vytvoření objednávky, mívají určitou sobu splatnosti. Je tedy nutné evidovat, které objednávky už byly zaplaceny a které ne. K tomuto slouží právě toto tlačítko. Má podobnou funkčnost jako tlačítko Potvrdit převzetí, ale zapisuje datum do databáze do jiného atributu, a to do atributu `zaplaceno`,
9. Položky objednávky – Každá objednávka musí obsahovat alespoň jednu položku, která udává, jaké zboží je objednáno. Maximální počet položek však omezen nijak není. Toto tlačítko po stisku vypisuje položky objednávky do komponenty `JTextArea`, která se nachází také pouze na panelech s objednávkami. Tento výpis je také možný po vytvoření příslušného potomka `JFramu`, ale právě kvůli

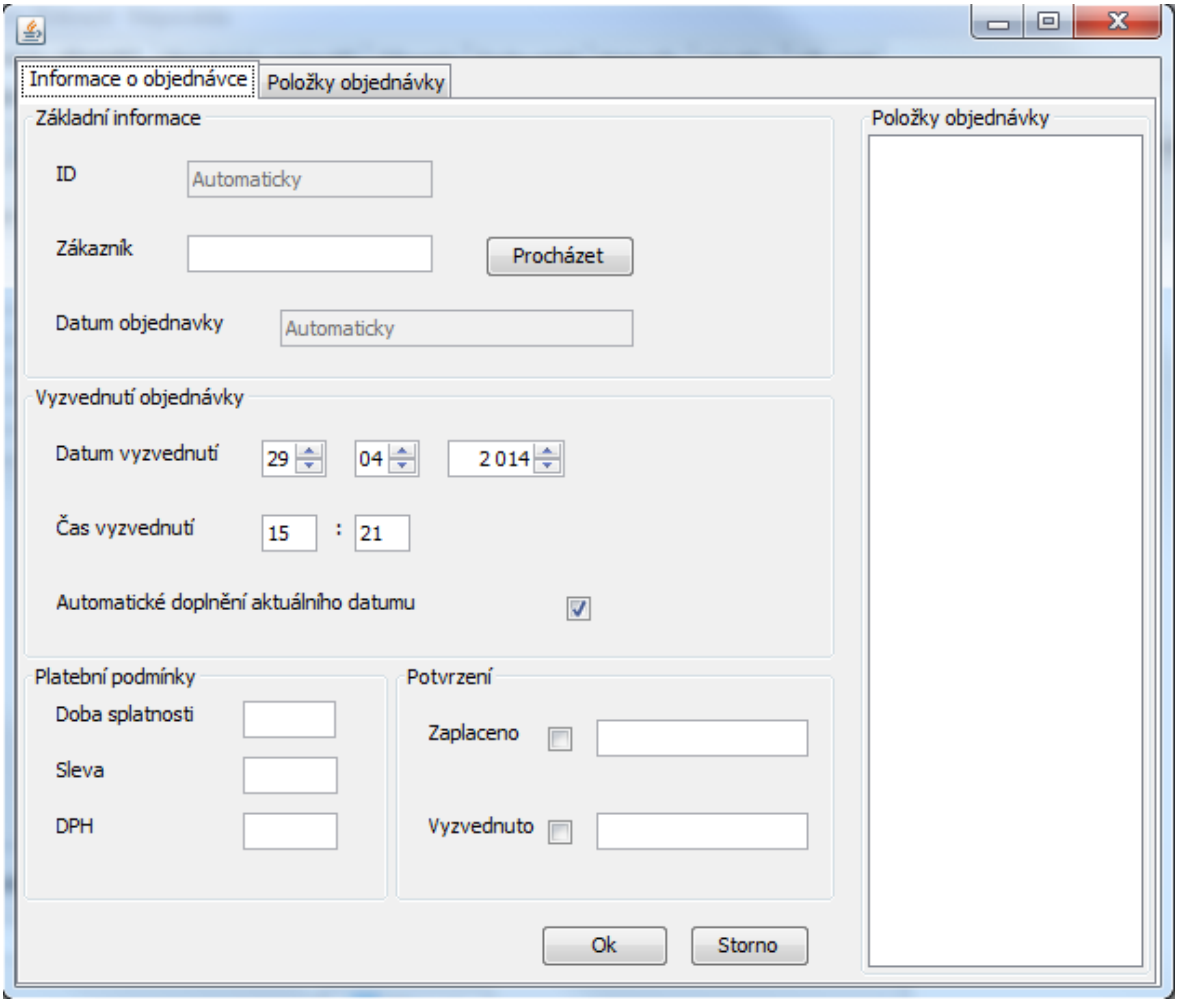
usnadnění byla tato funkčnost přidána i do panelů pro správu objednávek. Výpis je vidět na následujícím obrázku [Obrázek 13].



Obrázek 13 - Ukázka výpisu položek objednávky. Zdroj vlastní

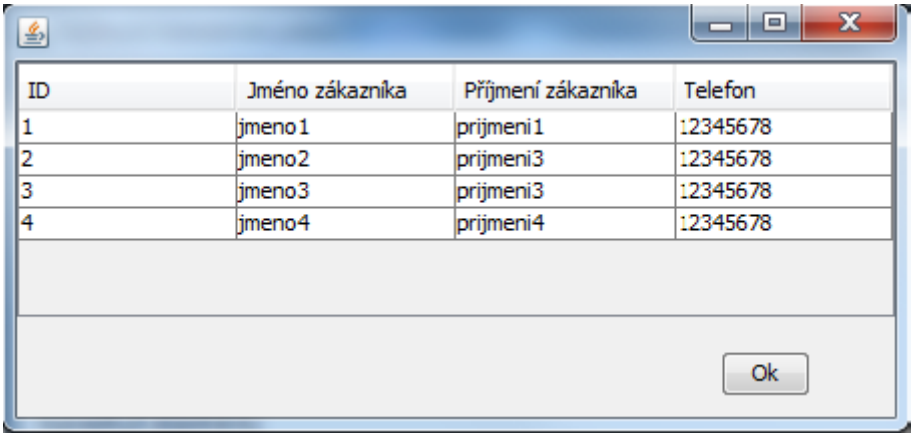
3.2.5.2 Vytváření a funkcionalita JFramů

V předchozí kapitole u popisu funkcí tlačítek bylo uvedeno, že po stisku se vytváří instance potomků `JFrame`. O těchto instancích bude dále mluveno jako o oknech. Tyto vytvářená okna jsou rozdílná pro každou entitu, kterou je možno v aplikaci spravovat (např. pro materiály, výrobky, zákazníky atd.) Je nezbytné dále uvést, že po stisku tlačítka Vytvořit, Editovat i Info se vytváří stejné okno, ale eviduje se, které tlačítko vytvoření podnítilo. To probíhá pomocí parametrů konstruktoru. Ty jsou dva, a to `lastFunction` (číslo, které udává, pomocí jakého tlačítka bylo okno vytvořeno.) a `selectedID` (parametr typu `String`, uchovávající informaci o ID záznamu, který má být editován nebo zobrazen). Funkcionalita všech vytvářených oken bude demonstrována na okně pro vytváření a editování objednávek zákazníků. Na obrázku [Obrázek 14] je vidět vytvořené okno pro nový záznam objednávky od zákazníka.



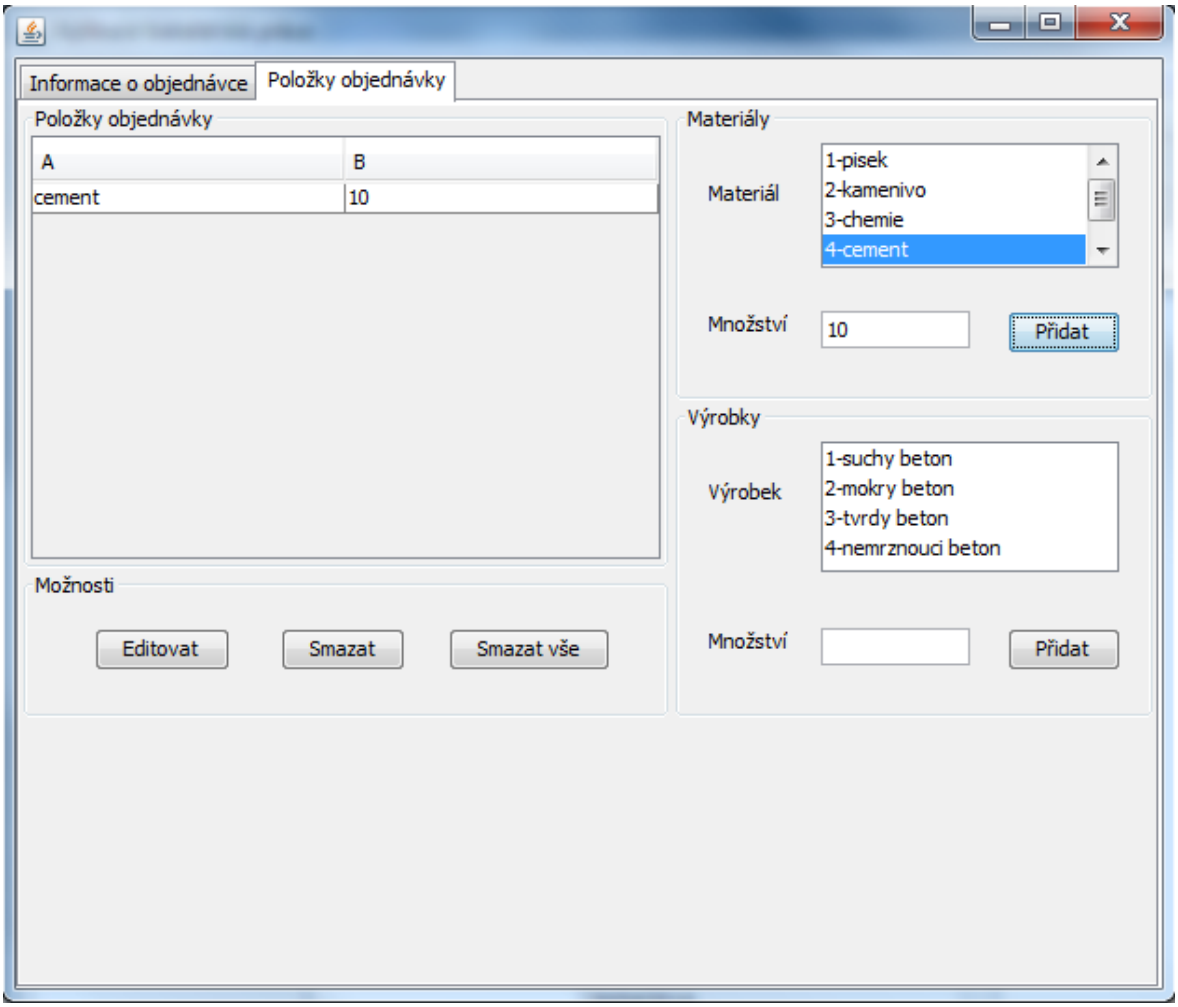
Obrázek 14 - Ukázka vytvořeného okna pro vkládání nových objednávek. Zdroj vlastní

Je vidět, že není možné doplnit všechny informace, protože některé informace doplňuje aplikace sama. Důležitou komponentou je tlačítko procházet, díky kterému je možné vytvořit další `JFrame` s seznamem zákazníků [Obrázek 15].



Obrázek 15 - Ukázka vytvořeného okna pro přiřazování zákazníků k objednávkám. Zdroj vlastní

To značně zjednodušuje přiřazování objednávek k zákazníkům, protože stačí pouze vybrat zákazníka a potvrdit. Dalším důležitým aspektem vytváření objednávek jsou její položky. Aplikace umožňuje objednávat zákazníkům jak materiály, tak i výrobky z těchto materiálů. To se provádí přepnutím panelu ve vrchní části okna. Při vkládání materiálů i výrobků je nutné zadat objednávané množství, což je kontrolováno aplikací.



Obrázek 16 - Ukázka vkládání položek do objednávky. Zdroj vlastní

Z obrázku [Obrázek 16] je patrné, že je možné jednotlivé položky objednávky editovat i mazat. Aby toto bylo možné, musí být položky někde uloženy. Kvůli tomuto byla vytvořena datová struktura, reprezentovaná třídou `PolozkaObjednavky`. Ta uchovává o každé objednávce následující informace:

1. Typ – rozlišuje, zda je položka výrobek nebo materiál
2. Id objednávky
3. Id zboží
4. Název zboží
5. Množství

Každá položka je potom ukládána do `ArrayListu` pomocí metody `add()`. Vytvoření tohoto `ArrayListu` vypadá takto:

Deklarace:

```
private List<PolozkaObjednavky> polozkyObjednavky;
```

Inicializace:

```
polozkyObjednavky = new ArrayList<PolozkaObjednavky>();
```

Zdrojový kód okna kromě obslužných metod komponent obsahuje i metody pro doplňování informací a dále metody pro kontrolování zadaných informace.

Metody pro doplnění informací:

1. `nastavZakaznika()`
2. `nastavAktualniDatum()`
3. `doplnInformace()`

Metody pro kontrolu zadaných informací:

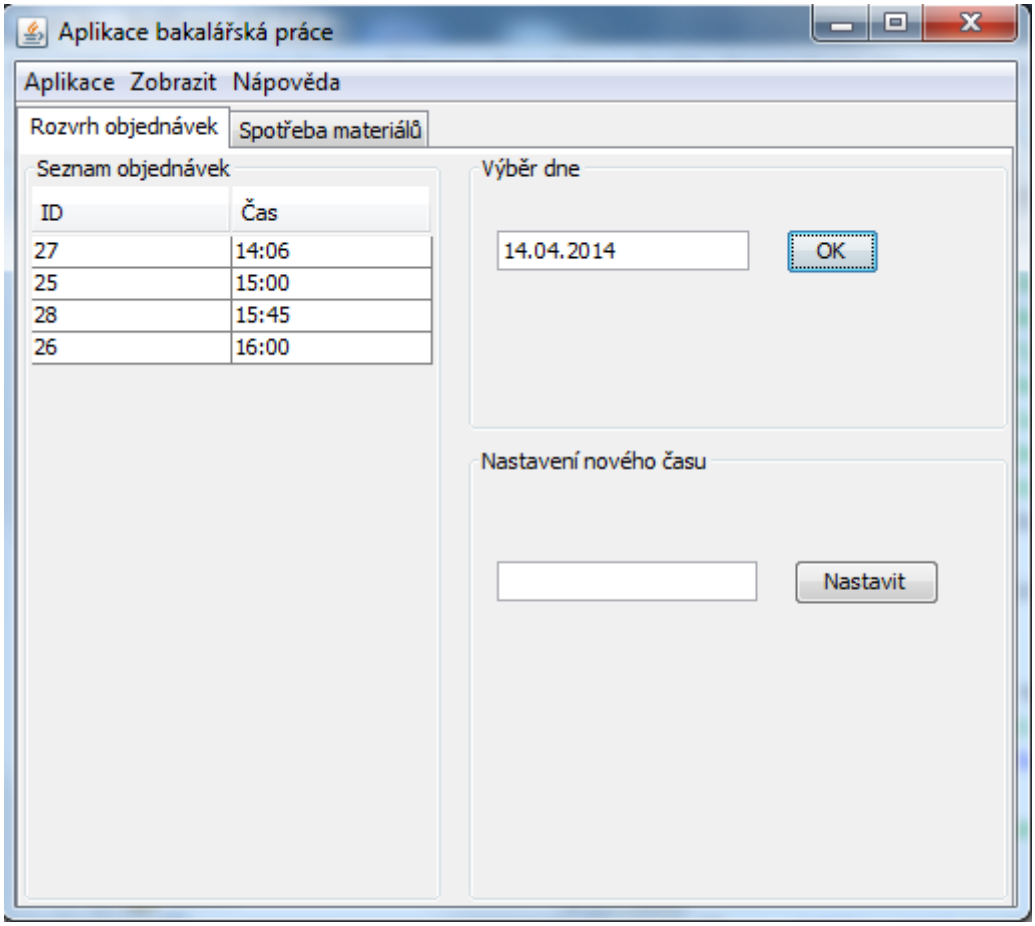
1. `vyplneneNutne()`
2. `spravneFormaty()`

3.2.6 Plánování

Plánování je nedílnou součástí dnešních aplikací. Značně zlepšuje chod firem i spokojenost zákazníků. Aplikace vyvinutá pro tuto práci umožňuje plánování zdrojů a objednávek. I toto by bylo nutné při praktickém použití upravit na míru pro jednotlivé zájemce. Pro nastavení hlavního panelu pro plánování je nutné vybrat příslušnou volbu v menu.

3.2.6.1 Rozvrh objednávek

Pro použití rozvrhu objednávek je nutné vytvořit instanci třídy `PanelRozvrhObjednavek`. Tento panel umožňuje vypsání objednávek pro zadaný den společně s jejich předběžným časem vyzvednutí. Toto umožňuje při vytváření nových objednávek předejít příjezdu několika zákazníků najednou. Což by mohlo způsobit potíže a značnou nespokojenost. Ukázka panelu pro rozvrh objednávek je na obrázku [Obrázek 17].

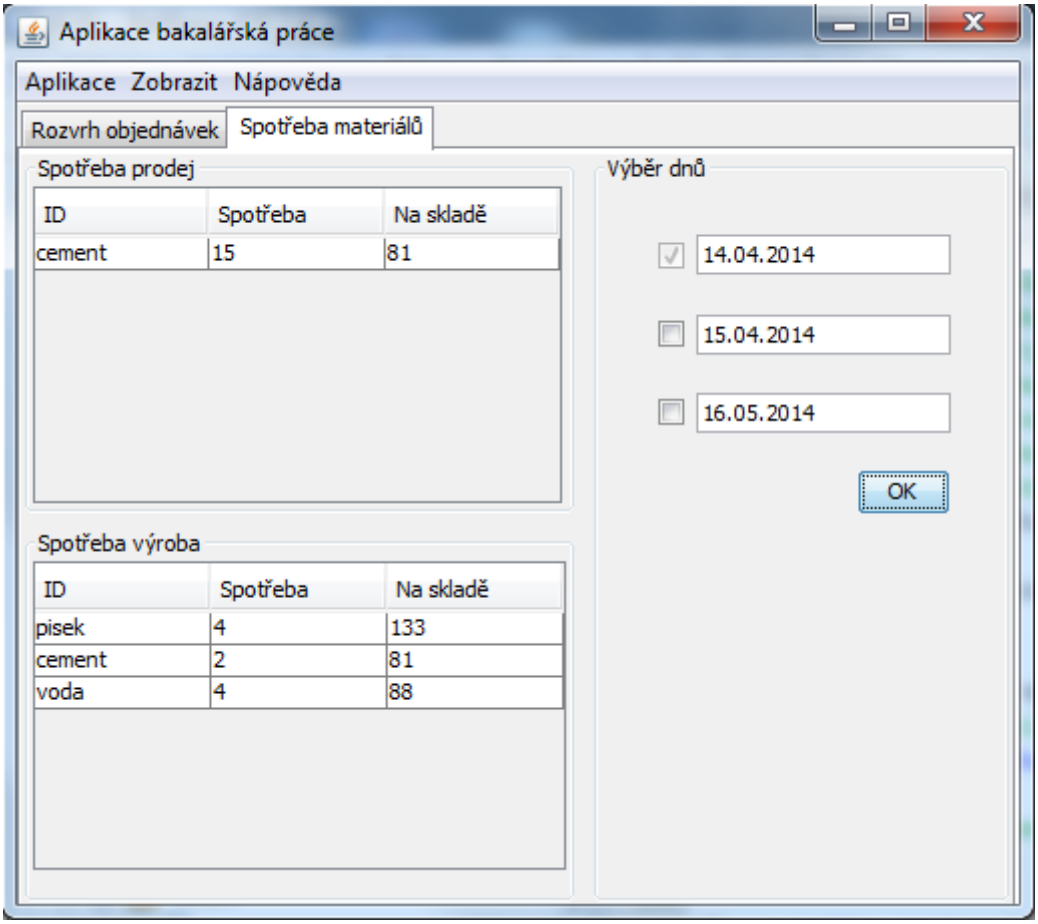


Obrázek 17 - Ukázka rozvrhu objednávek. Zdroj vlastní

Další možností tohoto panelu je změnit čas vyzvednutí každé objednávky. Je předpokladem, že to by se provádělo až po domluvě se zákazníkem.

3.2.6.2 Spotřeba materiálů

Protože dodání materiálu od dodavatele není nikdy okamžité, je nutné ho objednávat dopředu. Předpokládá se, že skladovací prostory každé firmy mají své limity. Proto by bylo vhodné znát spotřebu materiálů dopředu. Procházet jednotlivé záznamy v seznamu objednávek a zapisovat si množství objednávaného zboží by bylo značně neefektivní a nejspíše i nepřesné. Kvůli tomuto problému vznikl tento panel, který umožňuje vypsat spotřebu jednotlivých materiálů na jeden, dva nebo tři dny dopředu.



Obrázek 18 - Ukázka plánování spotřeby materiálů. Zdroj vlastní

Z obrázku [Obrázek 18] je vidět že výpis probíhá ve dvou tabulkách. V jedné se zobrazuje předpokládaný prodej jednotlivých materiálů, ve druhé potom množství materiálů, které je nutné pro výrobu objednaných výrobků. Data v levé části jsou doplňována automaticky, přičemž do prvního vždy aktuální datum, do druhého zítřejší atd. Pomocí zaškrtávacích boxů je možné plánování rozšířit na 2 nebo i 3 dny dopředu. Předpokládá se, že dodání materiálů od dodavatele netrvá více než 3 dny od podání požadavku.

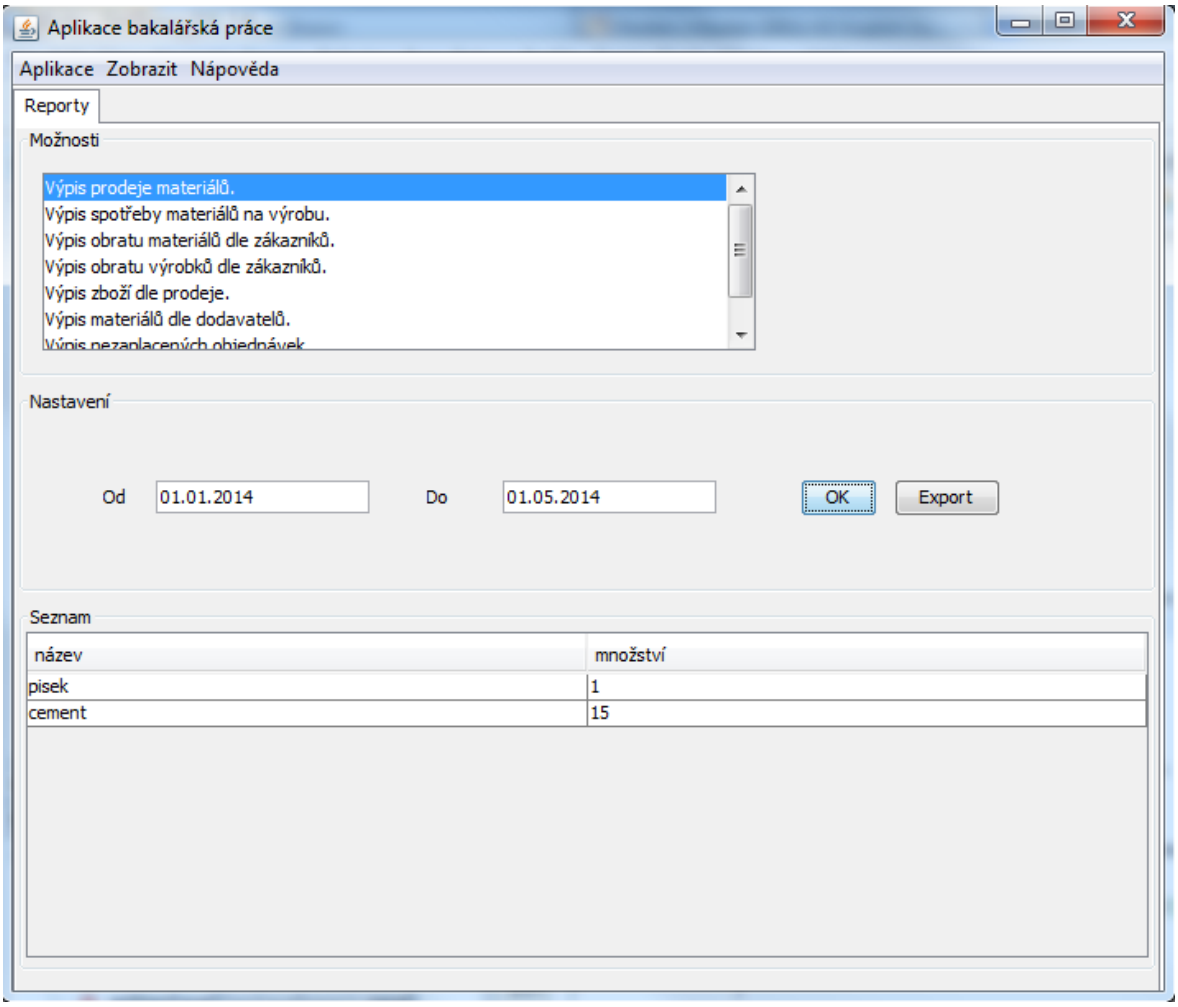
3.2.7 Reporty

Reporty umožňují zpětnou kontrolu ve firmě. Nejsou tedy nezbytně nutné pro chod firmy, ale velice ho usnadňují. Do panelu pro reporty se lze opět dostat v menu. Reporty, které jsou v této práci použity, demonstrují možnosti aplikace. Většina reportů se váže na nějaké období (např. týdenní, měsíční atd.), proto aplikace umožňuje zadat z jakého období se má většina reportů vypisovat.

Seznam reportů s popisem:

1. Výpis prodeje materiálů – seznam materiálů a množství jejich prodeje za určité období.
2. Výpis spotřeby materiálů na výrobu – seznam materiálů a jejich množství použité na výrobu za určité období.

3. Výpis obratu materiálů dle zákazníků – seznam zákazníků a množství materiálů, které za určité období zakoupil.
4. Výpis obratu výrobků dle zákazníků – seznam zákazníků a množství výrobků, které za určité období zakoupil.
5. Výpis zboží dle prodeje – seznam materiálů a výrobků společně s množstvím jejich prodeje.
6. Výpis materiálů dle dodavatelů – seznam dodavatelů a množství materiálů, které dodali.
7. Výpis nezaplacených objednávek – seznam objednávek, které doposud nebyly zaplaceny.
8. Výpis nevyzvednutých objednávek – seznam objednávek, které zatím nebyly vyzvednuty.



Obrázek 19 - Ukázka panelu s reporty. Zdroj vlastní

Protože každý report vlastně reprezentuje dotaz do databáze, je zde uveden dotaz, který stojí za vypsáním výsledku z obrázku výše [Obrázek 19].

```
dotazDB = "select nazev,sum(mnozstvi) from materialy join
polozkyobjednavkym using(id_material) join objednavkyzakaznici
using(id_objednavky) where datum_vyzvednuti BETWEEN to_date('" +
```

```
jTextFieldOd.getText() + "','DD.MM.YYYY') AND" + " to_date('" +
jTextFieldDo.getText() + "','DD.MM.YYYY') group by nazev";
```

3.2.7.1 Export XLS

Tato funkce byla do aplikace implementována kvůli dalšímu zpracovávání výsledků reportů a možnosti ukládání. Do `.xls` souboru se vždy exportuje obsah tabulky s výsledky reportů. Export probíhá pomocí metody `export()`, která je součástí třídy ExportToXls. Tato metoda přijímá dva argumenty:

1. Instanci třídy `JTable` – tabulka s daty.
2. Instanci třídy `File` – soubor, do kterého se bude zapisovat.

K zápisu se používají instance tříd `FileWriter` a `BufferedWriter`. Celý kód je k vidění zde:

```
public void export(JTable table, File file) throws IOException {
    TableModel model = table.getModel();
    FileWriter out = new FileWriter(file);
    BufferedWriter bw = new BufferedWriter(out);
    for (int i = 0; i < model.getColumnCount(); i++) {
        bw.write(model.getColumnName(i) + "\t");
    }
    bw.write("\n");
    for (int i = 0; i < model.getRowCount(); i++) {
        for (int j = 0; j < model.getColumnCount(); j++) {
            bw.write(model.getValueAt(i, j) + "\t");
        }
        bw.write("\n");
    }
    bw.close();
}
```

A volání této metody pomocí vytvořené instance takto:

```
ostatni.ExportToXls exp = new ExportToXls();
try {
exp.export(jTable1, new File("E:/bakalarska prace/zkouska.xls"));
} catch (IOException ex) {
}
```

3.2.8 Vkládání ikon

Kvůli lepšímu vzhledu byly k důležitým tlačítkům a položkám menu přidány ikony. Jedná se o malé obrázky s příponou .png a velikostí 16x16 pixelů. Ikony jsou reprezentovány datovým typem `ImageIcon`. V aplikaci jsou všechny ikony vytvořeny ve třídě Ikony a poté pomocí tzv. getterů přiřazovány komponentám metodou `setIcon()`.

Ukázka vytvoření ikony:

```
novy = new ImageIcon("E:\\bakalarska prace\\new.png");
```

Ukázka přiřazení ikony komponentě:

```
aplikaceOdhlasit.setIcon(ikony.getOdhlasit());
```

3.2.9 Přepnutí vzhledu

Protože jsou komponenty v Javě *„skinovatelné“*, je možné každému formuláři i komponentě nastavit různý vzhled, tzv. look and feel. Provádí se to pomocí metody `setLookAndFeel()` ve třídě `UIManager`. V aplikaci je tento příkaz zavolán v metodě `main()`, tedy ještě před vytvořením základního okna aplikace, takto:

```
UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
```

Tímto příkazem byl nahrazen standardní layout Javy za layout Windows.

Závěr

Protože ERP systémy, se kterými pracují společnosti a podniky na trhu, musejí být vždy přizpůsobeny na míru všem nejrůznějším potřebám, lze předpokládat jejich vývoj i do budoucna a celkově označit tento trh jako perspektivní. Aplikaci vytvořenou jako cíl této práce lze považovat za jednoduchý ERP systém, neboť umožňuje spravovat různé odvětví firemního chodu (logistiku, účetnictví, zaměstnance atd.).

Aplikace byla navržena jako demonstrace ERP systému, nelze ji tedy srovnávat s velkými firemními aplikacemi na trhu. Nicméně ale splňuje všechny funkce ze zadání práce a mohla by být v omezeném rozsahu nasazena v reálném provozu.

Tvorba aplikace probíhala ve dvou krocích, a to: návrh a vytvoření databáze a implementace aplikace. Návrh a vytvoření databáze bylo mnohem snazší díky předchozí znalosti této problematiky. Oproti tomu při vlastní implementaci aplikace jsem byl postaven před spoustu nových prvků (např. práce s ovladačem JDBC nebo tabulkou `JTable`). Při implementaci jsem narazil na mnoho problémů, ať už malých nebo velkých. Za největší považuju problém editaci položek objednávek, z důvodu rozsahu nutného řešení.

Pro případné budoucí rozšíření se nabízí několik možností. Jednou z funkcionalit je možnost jakéhokoli tisku (např. tisk objednávek nebo dodacích listů), která z časových důvodů není v aplikaci implementována. Mezi další možnosti bych zařadil rozšířenou správu skladovacích prostor a implementaci nějakých funkcí pro snadnější a pohodlnější vedení účetnictví (např. funkce pro roční a měsíční uzávěrky).

V závěru bych rád uvedl, že tvorba této práce mě bavila a přinesla mi spoustu nových znalostí a zkušeností. Věřím, že tato práce může pro někoho dalšího posloužit jako zdroj inspirace, ať už pro možnosti komunikace s databází nebo pro vytváření GUI aplikací v programovacím jazyce Java.

Literatura

[1] ERP systém - Informační systémy. [online]. [cit. 2014-05-02]. Dostupné z: http://www.vaclavkeil.cz/erp-system/

[2] BICAN, Jiří. *Aktuální stav na trhu s ERP systémy, systémy nabízené v cloudu*. Praha, 2013. Bakalářská práce. ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE. Vedoucí práce Ing. Pavel Náplava.

[3] BASL, Josef. *Podnikové informační systémy: podnik v informační společnosti*. 1. vyd. Praha: Grada, 2002, 142 s. ISBN 80-247-0214-2.

[4] Historie ERP systémů : ERP SYSTÉMY. *erp-systemy.cz* [online]. 1.5.2011 [cit. 2014-05-02]. Dostupné z: http://erp-systemy.cz/historie-erp-systemu/

[5] Informační systémy ERP v roce 2014. *jkr.cz* [online]. 21.01.2014 [cit. 2014-05-02]. Dostupné z: http://www.jkr.cz/trendy-v-podnikovych-informacnich-systemech-erp-v-roce-2014#more-34480

[6] Dodavatelé informačních a ERP systémů. *combitrading.cz* [online]. [cit. 2014-05-02]. Dostupné z: http://www.combitrading.cz/partnerstvi/dodavatele-informacnich-a-erp-systemu.html

[7] Úvod do JDBC. ŠEDA, Jan. *Interval.cz* [online]. 04.03.2003 [cit. 2014-05-02]. Dostupné z: http://interval.cz/clanky/uvod-do-jdbc/

[8] VÁŇA, Tomáš. *SQL manažer*. Pardubice, 2009. Bakalářská práce. Univerzita Pardubice. Vedoucí práce Ing. Zdeněk Šilar.

[9] Chapter 1: JDBC Drivers. *Nguyenlamminhdieu.com* [online]. [cit. 2014-05-02]. Dostupné z: http://nguyenlamminhdieu.com/zone/209/news/254-chapter-1-jdbc-drivers.aspx

[10] Oracle a Java - Štěpán Roh - Naživu leč ospalý. ROH, Šťepán. *Alivebutsleepy.srnet.cz* [online]. 30.11.2013 [cit. 2014-05-02]. Dostupné z: http://alivebutsleepy.srnet.cz/oracle-a-java/

[11] Java Database Connectivity. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2014 [cit. 2014-05-02]. Dostupné z: http://cs.wikipedia.org/wiki/Java_Database_Connectivity