Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Skladový systém pro obalovny živičných směsí

Plzeň, 2014 Martin Hron

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 5. května 2014

Martin Hron

Poděkování

Děkuji panu prof. Ing. Jiřímu Šafaříkovi CSc. za odborné vedení, ochotu při konzultacích a pomoc při psaní této práce. Dále děkuji ekonomickému oddělení firmy SWIETELSKY stavební s.r.o. za poskytnuté materiály a možnost absolvovat tuto práci.

Abstrakt

Tato práce se zabývá problematikou skladových systémů jak z ekonomického hlediska, tak z hlediska programátorského. Cílem práce je navrhnout a implementovat skladový systém pro potřeby zadavatele. Systém je implementován v jazyce C# pod platformou Microsoft .NET a využívá vhodný relační databázový systém. Práce se zabývá problematikou oceňování skladových položek, návrhem vylepšení stávajícího systému či návrhem dalších uživatelsky přívětivých funkcí.

Dále práce obsahuje popis struktury navrhovaného systému, návrh databázového a návrh objektově orientovaného modelu nového systému. V práci jsou také popsány technologie a knihovny použité v systému a technické požadavky pro běh systému. Práce zahrnuje také testování a zhodnocení nové aplikace oproti stávající včetně možných vylepšení do budoucna.

Klíčová slova

sklad, skladový systém, zásoby, oceňování zásob, databáze, systém řízení báze dat, PostgreSQL, Microsoft .NET Framework, C#, ADO.NET, Npgsql, iTextSharp

Abstract

This thesis deals with issues of store systems both from economic and programming point of view. The main aim of this thesis is to design and implement a store system for needs of the client. The system is implemented in C# language under Microsoft .NET platform and uses the appropriate relational database system. The work describes problems of valuation of inventory items, suggests improvements to the existing system or other user-friendly features.

This thesis also contains a description of the structure of the proposed system, a design of the database model, and a design of the object-oriented model of the new system. The work also describes the technology and libraries used in the system and technical requirements for running the system. The thesis also includes testing and evaluation of the new application compared to the existing one, including possible improvements for the future.

Keywords

store, store system, inventories, inventory valuation, database, database management system, PostgreSQL, Microsoft. NET Framework, C#, ADO.NET, Npgsql, iTextSharp

Obsah

1 Úvod

Ekonomické oddělení ve firmě SWIETELSKY stavební s.r.o. od počátku vzniku používá skladový systém běžící v operačním systému MS-DOS. Tento systém je velmi nekomfortní na užívání, neustále způsobuje pády, při kterých dochází ke ztrátě dat a zálohování databáze je možno pouze na disketové jednotky. Z důvodu neustálého vývoje v oblasti informačních technologií a nespokojenosti se stávajícím software, zadala firma požadavek na vytvoření nového systému pro evidenci skladových zásob.

Cílem práce je vytvořit nový skladový systém, který by měl nahradit stávající systém. Systém by měl využívat vhodný relační databázový systém a měl by být přizpůsoben potřebám zadavatele. Aplikace by měla být naprogramována v jazyce C# pod platformou Microsoft .NET a měla by být výhradně v českém jazyce.

Práce popisuje problematiku skladových systémů jak z ekonomického hlediska, tak z hlediska programátorského. Zabýváme se zde například problematikou oceňování skladových položek podle různých způsobů, návrhem vylepšení stávajícího systému či návrhem dalších uživatelsky přívětivých funkcí na základě analýzy různých skladových software.

Dále práce popisuje strukturu navrženého systému, návrh databázového modelu a návrh objektově orientovaného modelu nového systému, které byly přizpůsobeny konkrétním potřebám zadavatele. V implementační části jsou popsány technologie a knihovny použité při implementaci systému a také technické požadavky na tento systém.

V poslední části se pak zabýváme samotným testováním vytvořené aplikace, případnou korekcí drobných nedostatků zjištěných na základě testování a zhodnocení vylepšení nové aplikace oproti stávající.

2 Zásoby

Tato kapitola popisuje co jsou to zásoby, na jaké skupiny se dělí a co do těchto skupin patří. Následně popisuje jak se tyto zásoby oceňují při pořízení a při úbytku.

Zásoby – Zásoby patří mezi krátkodobý oběžný majetek. U tohoto majetku dochází příslušnou činností podniku k jednorázově spotřebě, nebo příslušnou činností vzniká a dochází k přeměně na jiné složky majetku [Zás(2014)].

2.1 Rozdělení zásob

Všechny druhy zásob dělíme do 3 skupin:

1. skladovaný materiál
2. zásoby vlastní výroby
3. skladované zboží

2.1.1 Skladovaný materiál

Materiálem jsou takové položky, které podnik nakupuje od různých dodavatelů, případně je vlastní činností vytváří a do materiálu je převede tzv. aktivací[1] [Car(2010)]. Mezi materiál patří:

- **suroviny** – základní materiál, který při výrobě zcela nebo z části přechází do výrobku a tvoří jeho podstatu
- **pomocné látky** – přecházejí přímo do výrobku, ale netvoří jeho podstatu (např. ochranný lak)
- **provozovací látky** – umožňují provoz nebo výrobu (např. oleje, palivo, čisticí prostředky, ...)

[1] situace, kdy podnik provede určitý výkon sám sobě, např. stavební firma si postaví budovu [Akt(2014)]

- **náhradní díly**
- **obaly a obalové materiály**
- **drobný hmotný majetek** – především hmotný majetek s dobou využitelnosti delší než jeden rok (např. notebook do 40 000Kč)
- **další movité věci** – především věci s dobou použitelnosti do jednoho roku, bez ohledu na výši ocenění (např. pokusná zvířata)

2.1.2 Zásoby vlastní výroby

Zásoby vlastní výroby jsou především produkty ať už úplné nebo částečné, které vznikají ve výrobním procesu z nakoupeného nebo získaného materiálu. Do zásob vlastní výroby se zahrnuje:

- **nedokončená výroba a polotovary**
  - *produkty* – „prošly jedním nebo několika výrobními procedurami, nejsou již materiálem, ale nejsou ani hotovým výrobkem (do této skupiny patří i nedokončené činnosti, při nichž nevznikají hmotné produkty)“ [Car(2010)]
  - *odděleně evidované produkty* – „polotovary, které dosud neprošly všemi výrobními procesy a budou dokončeny nebo zkompletovány do hotových výrobků v dalším výrobním procesu“ [Car(2010)]
- **hotové výrobky** – věci vlastní výroby určené k prodeji nebo ke spotřebě uvnitř firmy
- **zvířata** – veškerá zvířata, která nejsou vykazována jako dospělá zvířata (např. jateční zvířata a jejich skupiny)

2.1.3 Skladované zboží

Skladované zboží se především popisuje jako zásoba, kterou podnik nakupuje (případně vytváří), za účelem dalšího prodeje. Jako zboží je možné označit:

- **movité věci včetně zvířat** – „zásoby získané za účelem prodeje, pokud společnost s těmito věcmi obchoduje“ [Car(2010)]

- **výrobky** – „zásoby vlastní výroby, které byly aktivovány a předány do vlastních prodejen“ [Car(2010)]
- **zvířata vlastního chovu** – „především zvířata, která dospěla, byla aktivována a jsou určena k prodeji“ [Car(2010)]

2.2 Oceňování zásob při pořízení

Oceňováním zásob se rozumí peněžní ohodnocení stavu zásob. V následujících podkapitolách se budeme věnovat oceňování v okamžiku jejich pořízení. Toto oceňování se podle způsobu pořízení dělí do 3 skupin:

- oceňování pořizovací cenou
- oceňování vlastními náklady
- oceňování reprodukční cenou

2.2.1 Oceňování pořizovací cenou

Tímto způsobem se nakupované zásoby oceňují pořizovacími cenami. Pořizovací cenou se rozumí nejen nákupní cena, ale také cena všech nákladů souvisejících s jejich pořízením, tzv. vedlejší pořizovací náklady. Mezi tyto náklady patří především clo, přepravné, pojistné při přepravě nebo DPH (pokud se jedná o neplátce DPH). Součástí vedlejších pořizovacích nákladů nejsou úroky z úvěrů nebo případné slevy snižující cenu pořízení.

2.2.2 Oceňování vlastními náklady

Tento způsob se používá především k oceňování výrobků vlastní výroby. K oceňování se používají jak přímé vlastní náklady, tak i nepřímé vlastní náklady. Přímými náklady se rozumí náklady na konkrétní druh výkonu (např. náklady na materiál) a nepřímé náklady můžeme chápat jako náklady, které se vztahují k určitému druhu výkonu (např. mzdy pracovníků nebo odpisy vybavení) [Ryn(2010)].

2.2.3 Oceňování reprodukční cenou

Tento způsob oceňování spočívá v ocenění zásoby cenou, kterou by daná zásoba měla v době účtování, neboli reprodukční cena, je hodnota dané zásoby v době účtování. Stanovení této ceny není dáno žádný předpisem, může být stanovena např. znaleckým posudkem nebo odborným odhadem. V případě, že při pořízení byly vynaloženy další náklady (například doprava), je nutné přičíst tyto náklady k reprodukční ceně. Oceňování reprodukční cenou se používá například pro zásoby nalezené jako přebytek při inventarizaci, pro zásoby nabyté jako odpad nebo zbytkový produkt výroby, apod.

2.3 Oceňování při úbytku zásob

Úbytkem zásob je myšleno především vyskladnění nebo přeskladnění, vyřazení pro nepotřebnost nebo v důsledku zničení či ztráty, vrácení dodavateli, darování nebo manko. Při úbytku zásob oceňujeme těmito způsoby:

- ocenění individuální pořizovací cenou
- ocenění váženým aritmetickým průměrem
- ocenění metodou FIFO
- ocenění metodou LIFO
- ocenění pevně stanovenou cenou

Dále si popíšeme pouze ocenění váženým aritmetickým průměrem, metodou FIFO a ocenění pevně stanovenou cenou.

2.3.1 Oceněný váženým aritmetickým průměrem

Metoda ocenění váženým aritmetickým průměrem je jednou z nejpoužívanějších metod na světě. Tato metoda spočívá ve výpočtu váženého aritmetického průměru z aktuálních zásob na skladě. Vzorec pro výpočet váženého aritmetického průměru vypadá takto:

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i \cdot x_i}{\sum_{i=1}^{n} w_i} \tag{2.1}$$

Kde x_i je jednotková cena daného materiálu a w_i je množství daného materiálu. Tento průměr je nutné počítat minimálně jednou za měsíc a to buď na začátku měsíce nebo na konci měsíce, případně jej lze počítat i průběžně s každou novou dodávkou. Podle toho se pak vážený aritmetický průměr dělí na:

- **periodický** – vážený průměr se vypočítává pouze jednou za měsíc vždy k určitému datu a při vyskladnění materiálu se k ocenění používá cena z minulého období (v některých společnostech se pro vyskladnění materiálu používá průměrná cena z aktuálního období)
- **proměnlivý** – vážený průměr se počítá vždy při každém novém přírůstku určitého materiálu

2.3.2 Oceňování metodou FIFO

Pojmenování této metody vzniklo z anglických slov „first in, first out“ neboli v překladu první dovnitř, první ven. Při vyskladnění se principiálně oceňuje hodnotou, kterou měly zásoby, které byly pořízeny jako první. Tedy při prvním vyskladnění daného druhu materiálu se oceňuje cenou, kterou měl daný materiál při prvním naskladnění. Při druhém vyskladnění se oceňuje cenou, kterou měl daný materiál při druhém naskladnění atd.

2.3.3 Ocenění pevně stanovenou cenou

Pro oceňování zásob je možné zvolit metodu pevně stanovené ceny. Princip této metody spočívá ve stanovení pevné ceny, která se bude používat k oceňování jak nakupovaných, tak vydávaných zásob. Stanovení pevné ceny není určeno žádným předpisem a závisí tedy pouze na firmě jakým způsobem si cenu zvolí. V praxi se většinou volí cena, která pokud možno co nejvíce odpovídá očekávané ceně, nebo alespoň přibližným cenám nakoupených zásob. Tato metoda se používá především při ocenění zboží v maloobchodních prodejnách, kdy je třeba znát hodnotu zboží v prodejních cenách.

3 Analýza skladových systémů

V této části práce bude popsána analýza především stávajícího systému a také dalších dvou skladových sytémů. Abychom se mohli věnovat analýze skladových systémů je nejprve nutné uvést, co je to **sklad** a také co je to **skladový systém**.

Sklad – Sklad je vyhrazené místo např. budova nebo místnost, která slouží k uložení (uskladnění) zásob.

Skladový systém – Pod tímto pojmem si můžeme představit systém pro evidenci skladu resp. skladovaných zásob (příjem a výdej materiálu), systém pro účtování zásob a jejich objednávek nebo systém pro inventarizaci. Můžeme tedy říci, že skladový systém je systém pro správu ekonomických operací nad skladem.

3.1 Analýza stávajícího systému

Aktuální systém s názvem **Zásoby**, je velmi starý software, který umožňuje zejména skladovou evidenci a inventarizaci. Tento software byl vytvořen přímo pro potřeby firmy a je používán dodnes. Funkčnost software byla z počátku bezproblémová, ovšem s vývojem IT technologií, např. vývojem operačních systémů, začínal postupně ztrácet krok s novějšími systémy a postupně se projevovalo čím dál více nedostatků a problémů.

Aplikace byla primárně vytvořena pro systém MS-DOS a pro spuštění na novějších operačních systémech (především na 64 bitových systémech) je bezpodmínečně nutné nastavit režim kompatibility, případně spustit aplikaci v programu DOSBox. Během užívání programu se ze zatím nezjištěných důvodů náhodně objevuje neočekávaná chyba programu, program se ukončí a dochází buď k částečné, nebo i úplné ztrátě dat. Tento problém by se dal alespoň částečně vyřešit pravidelným zálohováním. Zde ovšem nastává problém, protože software umožňuje zálohu dat pouze na disketové jednotky, které se v dnešní době takřka nepoužívají.

Další nedostatek, který vznikl postupem času je ten, že aplikace nedokáže komunikovat s moderními tiskárnami a je prakticky nemožné přímo z programu vytisknout potřebné dokumenty. Tento problém se podařilo vyřešit

utilitou DOSPrinter, který dokáže z daného programu vytisknout potřebný dokument, ale rozvržení textu obzvláště při více stránkovém tisku je velice špatné.

Jedním z nedostatků, které byly při analýze nalezeny je také duplicita dat obsažených v dané databázi. Všechna používaná data jsou uložena ve dvou tabulkách. První tabulka obsahuje informace o přijatých nebo vydaných materiálech. Druhá tabulka obsahuje informace o množstvích a průměrných cenách materiálu na skladě. V obou tabulkách vzniká duplicita dat (např. vždy je nutné uvést název materiálu a jeho kód), která by se dala v běžných relačně orientovaných databázových systémech snadno odstranit. Je nutné podotknout, že aplikace využívá jako databázový systém SŘBD[1] dBase, který byl prvním široce používaným databázovým systém pro mikropočítače a i když jej tvůrci považovali za relační databázový systém, nesplňoval všechna kritéria pro toto označení.

3.2 Analýza jiných skladových systémů

V této části si stručně popíšeme další dva skladové programy, jejichž analýza by mohla přinést pozitiva pro nově vznikající software. Prvním je demoverze programu **Skladík** [Bán(2002)] a druhým pak ne přímo skladový software, ale spíše pokladní systém **AWIS** [Awi(2013)].

3.2.1 Skladík 5.5.5 Demo

Tento software je poměrně uživatelsky přívětivý a jednoduchý na ovládání. Velkým pozitivem toho programu je jeho přehledné uživatelské rozhraní a především možnost nastavení způsobu oceňování cen při výdeji. Mezi tyto způsoby se řadí především metoda **váženého aritmetického průměru** (jak periodického, tak proměnlivého), metoda **FIFO** a také metoda **ocenění pevně stanovenou cenou**. Program používá tzv. skladové karty materiálu, což mu umožňuje provádět hromadnou změnu cen na kartách.

Další výbornou funkcí softwaru je možnost hromadné fakturace výdejek, kde je ovšem nutné evidovat i zakázky podniku. Aplikace také umožňuje import a export do souboru, kde využívá jak vlastní formáty souborů, tak

[1] co je to SŘBD se dozvíte dále v kapitole 6

soubory formátu csv a pro export také PDF. S tím souvisí možnost zálohování a obnovování dat.

V neposlední řadě program umožňuje editaci všech příjmových a výdajových dokladů a také souhrnný přehled všech dodavatelů a odběratelů, které je možné různě editovat nebo informovat emailem.

3.2.2 Pokladní systém AWIS – Sklad

Jak již bylo řečeno, tento software je primárně určen jako pokladní systém, ale obsahuje poměrně rozsáhlou skladovou část. Důvodem pro výběr tohoto programu k popisu je fakt, že obsahuje poměrně široké možnosti skladových operací. Tyto možnosti jsou způsobeny především tím, že program umožňuje pouze metodu **váženého proměnlivého průměru** při výdeji zásob.

Hlavními výhodami této aplikace jsou poměrně velké možnosti pro filtraci položek na skladě a také jednoduché a rychlé přidávání (příjem) a vydávání (výdej) položek. Při těchto operacích program zároveň hlídá maximální a minimální množství položek na skladě a také hlídá dobu expirace jednotlivých zásob. S příjmem a výdejem položek také souvisí možnost jednoduché editace příjmových i výdajových položek.

Tato aplikace, stejně jako již zmíněná aplikace **Skladík 5.5.5 Demo**, umožňuje fakturaci výdajových dokladů a také využívá soubory vlastního formátu pro zálohování dat. Dalšími přednostmi tohoto programu jsou export do souboru formátu xls, také import ze souboru stejného formátu a jednoduchá editace položek na skladě. Aplikace také využívá možnost správy jednotlivých uživatelů a je zde možno každému uživateli nastavit různá práva pro správu různých skladů.

Zajímavostí softwaru je možnost spárování s mobilním zařízením, což je využitelné spíše pro pokladní systém jako takový, ale tato funkce umožňuje přes mobilní zařízení prohlížet i stav zásob na skladě.

4 Návrh funkčních vlastností

V této kapitole si nejprve upřesníme požadavky zadavatele a následně popíšeme návrh nových funkčních vlastností, které by bylo ve stávajícím funkčním systému potřeba vylepšit nebo případně doplnit. Všechny návrhy na vylepšení vznikly na základě předchozí kapitoly nebo na základě potřeb zadavatele.

4.1 Specifikace požadavků zadavatele

Podle požadavků zadavatele by měl nový systém obsahovat alespoň stejné funkční vlastnosti jako stávající. Mezi nejdůležitější požadavky patří:

- **správa více skladů** – možnost uživatele spravovat více skladů, které nejsou vzájemně propojeny, ale každý sklad by měl aktuálně spravovat vždy pouze jeden uživatel
- **operace na skladě** – možnost příjmání a vydávání materiálu a také rozlišení typu příjmů (Příjem, Příjem dopravy) a výdejů (Výdej do spotřeby, Výdej – prodej)
- **oceňování cen při výdeji** – automatický výpočet průměrných cen podle typu oceňování zásob daného skladu
- **evidence dokladů** – uchovávání všech dokladů daného skladu
- **generování sestav** – generování obratových soupisek, inventurních soupisek a kontrolního opisu zásob vždy za určité období
- **tisk sestav** – umožnění tisku výše zmíněných vygenerovaných sestav
- **záloha dat** – možnost zálohování a obnovování dat

4.2 Specifikace nových potřebných vlastností

- **snaha o minimalizaci duplicity dat**
- **úprava rozvržení tisku** – vylepšení rozvržení dat na stránce především u více stránkového tisku
- **vylepšení interakce s uživatelem** – například po zadání kódu materiálu se programově doplní příslušné položky k materiálu
- **export dokumentů do pdf** – především export sestav a inventárních soupisek
- **umožnění zálohy dat** – především záloha dat jednotlivých skladů
- **zajištění připočítávání dopravy** – při zadávání příjmových dokladů je nutné připočítat cenu dopravy k jednotkové ceně každé položky dokladu
- **možnost změny velikosti hlavního okna programu**

Tyto vlastnosti je bezpodmínečně nutné začlenit do nového skladového systému.

4.3 Další možná rozšíření

- **zavedení více možností filtrování dat v sestavách**
- **možnost generování příjemek/výdejek** – na základě zadaných položek a informací možnost vygenerovat a následně vytisknout novou příjemku/výdejku
- **možnost individuálních uživatelských nastavení** – například možnost nastavení cest ukládání vyexportovaných dokumentů
- **možnost úpravy příslušné příjemky/výdejky včetně položek**

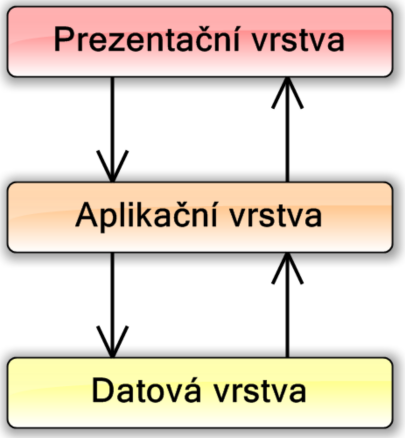
Tyto vlastnosti by bylo vhodné začlenit do nového software především kvůli zkvalitnění funkčnosti.

5 Návrh nového skladového sytému

V následující kapitole bude vysvětleno co to je třívrstvá architektura a proč byla při návrhu zvolena. Následně popíšeme strukturu systému a návrh jednotlivých částí systému pomocí UML diagramu.

5.1 Třívrstvá architektura

Třívrstvá architektura je jedna z nejznámějších vícevrstvých architektur používaných v softwarovém inženýrství. Tuto architekturu lze považovat za návrhový vzor a je nejčastěji používána u webových aplikací. Tento návrhový vzor je tvořen, jak již napovídá název, třemi vrstvami. Tyto vrstvy se nazývají **prezentační**, **aplikační** a **datová** a jejich uspořádání je vidět na obrázku 5.1 [Man(2011)].



Obrázek 5.1: Třívrstvá architektura

Stručná charakteristika těchto vrstev je následující:

- **Prezentační vrstva** – je někdy také nazývána vrstvou uživatelského rozhraní. Její hlavní funkce je zobrazovat informace v uživatelsky přívětivé a srozumitelné podobě. V této vrstvě je možné například kontrolovat zadávané vstupy, neměla by ovšem provádět složité operace nad daty. Ve většině případů je prezentační vrstva závislá na platformě a může být různá pro různá zařízení či platformy.

- **Aplikační vrstva** – je prostřední vrstvou v třívrstvém modelu a lze ji také nazývat funkční vrstvou. Ve většině případů tvoří tuto vrstvu jádro aplikace. Vrstva je odpovědná za přenos dat mezi prezentační a datovou vrstvou a provádí různé transformace těchto dat.
- **Datová vrstva** – někdy také nazývána databázovou vrstvou, je nejčastěji reprezentována různými uložišti dat. Ve většině případů tvoří tyto uložiště systémy řízení báze dat nebo jiná perzistentní uložiště.

Tato architektura byla zvolena především z důvodu **použití databázového systému**. Následně pak k výběru přispěly výhody této architektury. Mezi tyto výhody patří **oddělení aplikační logiky od prezentační vrstvy** a také **nezávislost na datové vrstvě**. Další výhodou této architektury je **možnost výměny jednotlivých vrstev**, především prezentační vrstvy, kde se tato výměna uplatňuje nejčastěji. Neposlední výhodou, která vedla k výběru této architektury byla také **možnost jednoduchého testování**.

5.2 Struktura aplikace

Systém je postaven na již výše zmíněné třívrstvé architektuře, která ovšem není využívána v její striktní podobě. Hlavním rozdílem oproti striktně dané třívrstvé architektuře je ten, že některé operace nad daty (například průměrování materiálu za období) jsou prováděny na základě volání uživatelských funkcí v databázovém systému a probíhají tedy na datové vrstvě. Důvodem této odlišnosti je případná jednoduchost výměny aplikační vrstvy, kde by některé funkce musely být implementovány znovu.

Datovou vrstvu tedy reprezentují uživatelské funkce databázového systému a databáze systému, která je blíže popsána v kapitole 6. Aplikační vrstvu tvoří jádro aplikace (`Jadro`). Tato část systému je zodpovědná za spojení s databází a tvoří část funkcionality systému. To znamená, že zajišťuje přenos dat mezi aplikací a databází a zároveň upravuje tato data do kompatibilního formátu pro přenos. Jádro aplikace je detailněji popsáno v sekci 5.3.3.

Prezentační vrstva je tvořena grafickým uživatelským rozhraním aplikace (`GUI`). Pomocí této části systému je především umožněno uživateli komunikovat s aplikací. Tato část také převádí data do uživatelsky přívětivého formátu a zároveň hlídá správný formát dat zadaných od uživatele. Prezentační

vrstva této aplikace také umožňuje export a tisk dat a je blíže specifikována v sekci 5.3.4

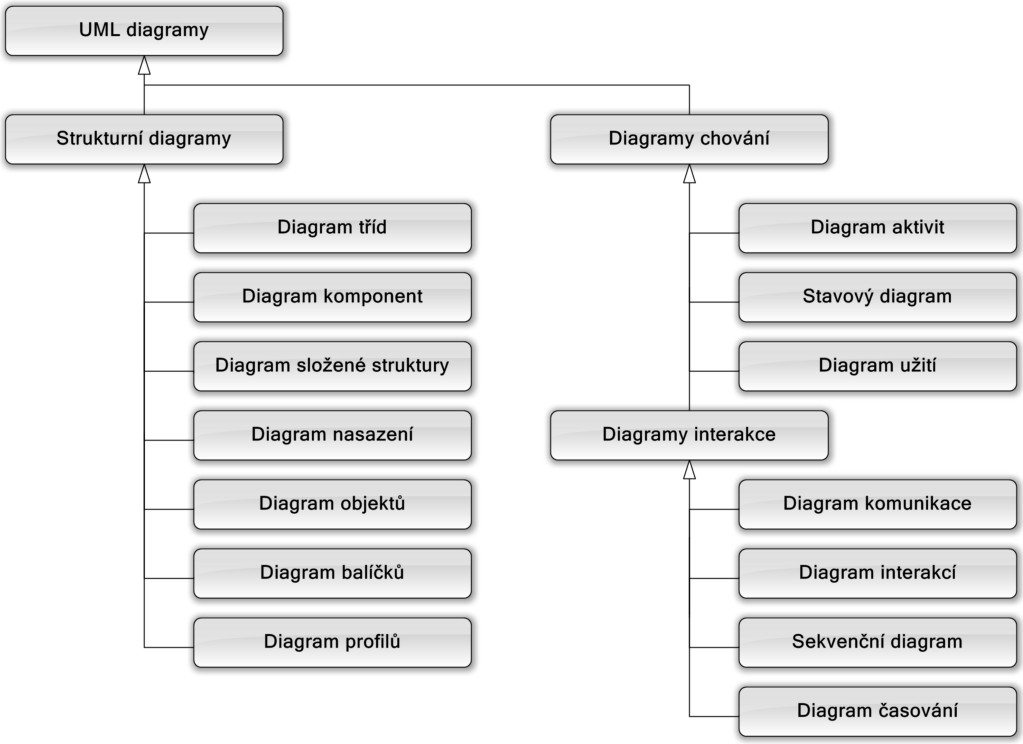
5.3 Popis UML diagramů tříd

V této podkapitole přistoupíme k popisu jednotlivých částí systému (konkrétně jádra aplikace a grafického uživatelského rozhraní). Pro správné pochopení je nejprve nutné uvést co je to **UML**, co to je **diagram tříd** a jaký význam mají jednotlivé prvky tohoto diagramu.

5.3.1 UML

UML neboli Unified Modeling Language je jednotný modelovací jazyk, který slouží k návrhu, specifikaci či vizualizaci softwarových systémů při jejich vývoji. Tento jazyk je standardizován a využívá se zejména při návrhu objektově orientovaných systémů. Historie UML sahá do 90. let minulého století, kdy se firmě Rational Software podařilo vytvořit standard UML na základě spojení několika metodik. Později vzniklo sdružení OMG (Object Management Group), které tento jazyk rozšiřuje a dohlíží na jeho specifikaci [Čáp(2014)]. V současné době je oficiální verzí tohoto jazyka verze 2.4.1 a je vyvíjena a testována verze 2.5 [Omg(2014)].

Jazyk UML zahrnuje několik diagramů zachycujících různé aspekty modelovaného systému. Hierarchie těchto diagramů je vidět na obrázku 5.2. Tyto diagramy se dělí do dvou základních skupin, na **strukturní diagramy** a **diagramy chování**. Strukturní diagramy popisují z jakých částí je systém složený a diagramy chování charakterizují, jak systém funguje. Diagramy chování zahrnují ještě jednu podskupinu diagramů a to **diagramy interakce**, které popisují interakci jednotlivých částí systému [Čáp(2014)].

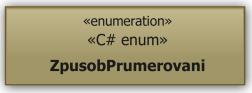


Obrázek 5.2: Hierarchie diagramů UML

5.3.2 Diagram tříd

Diagram tříd představuje pohled na statickou strukturu systému a jeho hlavním úkolem je znázornit objekty systému, jejich typy a vztahy mezi nimi. Tento diagram je tvořen dvěma základními prvky **třídami** a **vztahy**. **Třída** reprezentuje objekty se společnými vlastnostmi a stejným chováním. Každá třída obsahuje vždy *název* a také může obsahovat *výčet atributů a operací včetně znázornění jejich modifikátorů přístupnosti* [Kuč(2007)]. Tyto součásti tříd není důležité blíže popisovat, protože nebyly v diagramech tříd jednotlivých částí systému použity.

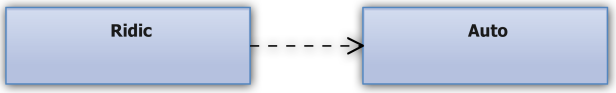
Dále pak některé třídy mohou zahrnovat tzv. *stereotyp* (případně několik stereotypů), které konkrétněji specifikují typ třídy. Stereotyp se pozná tak, že je uzavřen do dvojitých ostrých závorek (<<stereotyp>>). Příklad třídy diagramu tříd, včetně stereotypu můžeme vidět na obrázku 5.3. Ten vyobrazuje třídu s názvem `ZpusobPrumerovani`, typu C# enum, která reprezentuje výčtový typ (enumeration).



Obrázek 5.3: Příklad třídy diagramu tříd

Vazby mezi třídami popisují **vztahy**. Tyto prvky popisují chování a komunikaci mezi třídami. Vztahy jsou realizovány různými typy čar, které vyjadřují konkrétní typy chování a směr chování je možno vyjádřit šipkou. Nejčastější typy vztahů jsou tyto:

- **Závislost** – je nejslabší vztah, který udává závislost jedné třídy na druhé. Závislost bývá realizována použitím druhé třídy jako lokální proměnné nebo parametru v metodě první třídy. Tento typ vztahu znázorňuje obrázek 5.4, který říká, že třída `Ridic` používá třídu `Auto`.



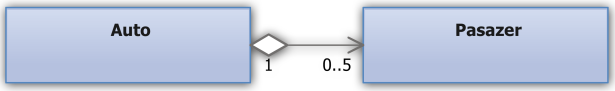
Obrázek 5.4: Příklad závislosti

- **Asociace** – určuje vztah mezi dvěma a více instancemi[1] tříd. Tento typ vazby typicky charakterizuje schopnost poslat zprávu jinému objektu pomocí referenční proměnné. Asociaci můžeme vidět na obrázku 5.5. Ta vyjadřuje, že instance třídy `Auto` obsahuje instanci třídy `Radio`.

Obrázek 5.5: Příklad asociace

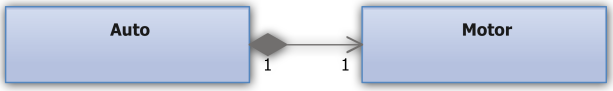
- **Agregace** – je volnější než kompozice (viz dále) a zároveň konkrétnější než asociace. Hlavním z rozdílů tohoto typu vztahu oproti kompozici je, že při zániku instance hlavní třídy nemusí zaniknou instance třídy vedlejší. Příklad agregace můžeme vidět na obrázku 5.6. Lze z něj vyčíst, že třída `Auto` využívá třídu `Pasazer` resp. několik jejích instancí a také, že při zániku instance třídy `Auto` budou instance třídy `Pasazer` existovat dál.

[1] instance – synonymum k objektu třídy



Obrázek 5.6: Příklad agregace

- **Kompozice** – je nejsilnějším typem vazby. U kompozice, na rozdíl od agregace, dochází při zániku instance hlavní třídy i k zániku instance třídy vedlejší. Na obrázku 5.7 je zobrazeno, jak kompozice vypadá. Tento obrázek nám říká, že třída `Auto` využívá třídu `Motor` a pokud dojde k zániku instance třídy `Auto`, instance třídy `Motor` ztrácí smysl a zaniká také.



Obrázek 5.7: Příklad kompozice

- **Realizace** – sděluje, že jeden předmět splňuje určitou podmínky toho druhého. Z programátorského hlediska většinou realizace určuje implementaci rozhraní. Příklad realizace znázorňuje obrázek 5.8. Tento příklad vyjadřuje, že třída `Auto` implementuje rozhraní `IRiditelny`, tedy obsahuje implementaci metod tohoto rozhraní.

Obrázek 5.8: Příklad realizace

- **Generalizace (dědičnost)** – vyjadřuje odvození jedné třídy od druhé. Z programového hlediska říkáme, že jedna třída dědí od druhé. Příklad tohoto typu vztahu je vidět na obrázku 5.9, kde třída `NakladniAutomobil` je odvozena (dědí) od třídy `Automobil`.

Obrázek 5.9: Příklad generalizace (dědičnosti)

U vztahů *asociace*, *agregace* a *kompozice* je možné uvádět ještě **násobnosti** vztahů. Tento prvek diagramu určuje kolik instancí jedné třídy se vztahuje k instanci druhé třídy. Násobnost se zapisuje dvěma způsoby. Prvním způsobem je zápis konkrétního počtu instancí (např. 1, 2, 5, ...). Zápis tohoto způsobu lze vidět na obrázku 5.7, který vyjadřuje, že jedna instance třídy `Auto` využívá **právě** jednu instanci třídy `Motor` a obráceně.

Druhou možností je zápis intervalem instancí (např. 0..5, 0..*, 1..*, ...), který určuje rozsah počtu instancí. Symbol * určuje neurčitý počet instancí. Na obrázku 5.6 je tento způsob zápisu znázorněn. Ten vyjadřuje, že jedna instance třídy `Auto` využívá až 5 instancí třídy `Pasazer`, ale nemusí využívat žádnou instanci této třídy. Naopak jedna instance třídy `Pasazer` využívá právě jednu instanci třídy `Auto`.

5.3.3 Jádro aplikace – `Jadro`

Jak již bylo uvedeno v sekci 5.2, tato část systému tvoří aplikační vrstvu a zajišťuje především spojení s databází. Strukturu této části systému znázorňuje diagram tříd jádra aplikace, který je přiložen v příloze A. Většina tříd, kromě tříd výčtových typů, třídy `Spojeni` a třídy `PrevodEnum` se svými atributy ztotožňují s příslušnými tabulkami v databázi. V tomto diagramu je nejdůležitější třídou třída `Spojeni`. Tato třída zajišťuje komunikaci s databázovým systémem a využívá ji proto většina tříd.

Další důležitou třídou je třída `Doklad`, která zajišťuje veškeré skladové operace. Každý doklad také musí obsahovat položky, které sami o sobě nemají žádný význam, proto je mezi třídou `Doklad` a třídou `Polozka` kompozice. Zároveň také instance třídy `Polozka` nedává bez materiálu žádný smysl a proto je tento vztah znázorněn rovněž kompozicí. Tato část systému také převádí data do formátu vhodného pro přenos mezi aplikací a databází. K tomuto účelu přispívá třída `PrevodEnum`, která, jak již napovídá název, převádí instance tříd výčtových typů na řetězce a opačně.

5.3.4 Grafické uživatelské rozhraní aplikace – `GUI`

Grafické uživatelské rozhraní aplikace reprezentuje prezentační vrstvu systému a jak bylo uvedeno v sekci 5.2, zajišťuje hlavně komunikaci s uživatelem. Struktura této části aplikace je vyobrazena v diagramu tříd v příloze B.

V tomto diagramu je nejdůležitější třídou třída `HlavniOkno`. Tato třída zprostředkovává největší část komunikace s uživatelem a proto využívá téměř všechny ostatní třídy. Z této třídy je také možné ovládat celou aplikaci.

Další důležitou třídou je třída `DokladFormular`. Ta zajišťuje všechny typy skladových operací a to i včetně jejich úprav. Tato třída využívá třídu `DokladKontrolka`, která je tvořena třídami `MujDateTimePicker` a `MujDataGridView`. Tyto 3 třídy jsou na sobě existenčně závislé, jak ukazuje diagram a bez instance třídy `DokladFormular` nemají význam. Proto jsou vztahy mezi nimi vyjádřeny kompozicí.

Dále je nutné zmínit třídy `PrehledDokladu` a `InventurniSoupiska`. Díky těmto třídám je umožněno prohlížet veškeré sestavy a díky implementaci rozhraní `IExportovatelny` je také možno vytisknout či exportovat tyto sestavy. I tyto třídy, stejně jako třída `DokladFormular`, jsou tvořeny instancemi jiných tříd, které jsou v kompozitním vztahu k těmto třídám.

6 Návrh databáze

V této kapitole si krátce popíšeme několik databázových systémů vhodných pro nový skladový systém a také zdůvodníme jaký SŘBD byl vybrán. Následně popíšeme návrh relačního databázového modelu nově vznikající aplikace. Než ovšem začneme charakterizovat jednotlivé části této kapitoly, bude vhodné nejprve vysvětlit co je to **SŘBD** a **relační databázový model**.

SŘBD – Systém řízení báze dat je v podstatě počítačový systém zabezpečující definování struktury dat, možnost manipulace s daty, ochranu dat a také komunikaci systému s uživatelem.

Relační databázový model – Tento model vychází z matematického relačního modelu a liší se především tím, že obsahuje schéma relace. Základem relačního modelu je **relace**, kterou reprezentuje tabulka. Každá tabulka obsahuje přesně definované **atributy** (sloupce) a tato definice udává schéma relace. Jednotlivé záznamy dat v těchto tabulkách se pak nazývají n-tice a vztahy mezi tabulkami se nazývají **vazby**.

6.1 Vhodné databázové systémy

Z velkého množství dostupných databázových systémů byli z těch nejznámějších zvoleni 3 zástupci. Tyto SŘBD byly voleny zejména na základně cenové dostupnosti (především zdarma dostupné databázové systémy) a také na základě možnosti přímé komunikace s platformou Microsoft .NET. Mezi 3 zástupce patří **MySQL**, **PostgreSQL** a **Microsoft SQL Server**.

U posledně zmiňovaného systému bylo pohlíženo spíše na druhé kritérium výběru a to zejména zde dvou důvodů. Prvním je, že komunikace mezi tímto systémem a platformou Microsoft .NET je oproti ostatním systémům na bezkonkurenčně nejlepší úrovni a druhým důvodem je „integrace běhového prostředí .NET CLR do databázového systému, což umožňuje implementovat databázové objekty v některém z dostupných jazyků této platformy“ [Aga(2009), s. 21]. Tyto zástupce si dále stručně charakterizujeme.

6.1.1 MySQL

Databázový systém MySQL je jedním z nejrozšířenějších open source[1] databázových systémů. Podle [Mya(2014)] bylo v celé historii MySQL staženo nebo distribuováno více než 100 miliónů kopií. Tento systém vytvořila firma MySQL AB a v současné době je vyvíjen a vlastněn firmou Oracle Corporation. Popularita tohoto systému je především v tom, že je nabízen jako součást softwarového balíčku LAMP (Linux, Apache, MySQL, PHP), který je využívám pro tvorbu webových aplikací.

Mezi hlavní výhody toho systému patří jeho **multiplatformnost**, tedy možnost instalace a provozování především na operačních systémech Microsoft Windows, Linux a také na několika dalších unixových systémech. Další výhodou tohoto systému je jeho **vysoký výkon**, **rychlost** a **relativní jednoduchost**.

SŘBD MySQL je distribuován v několika edicích. Základní edicí je MySQL Community Edition, která je volně dostupná. Další tři nabízené edice MySQL Standard Edition, MySQL Enterprise Edition a MySQL Cluster Carrier Grade Edition jsou komerční a placené verze, které se od základní edice liší různými rozšiřujícími funkcemi a nástroji.

6.1.2 PostgreSQL

Databázový systém PostgreSQL je objektově-relační open-source databázový systém, který je jedním z hlavních konkurentů MySQL. Tento systém původně vznikl jako projekt Ingres na Kalifornské univerzitě v Berkley. Vedoucím tohoto projektu byl prof. Michael Stonebraker. V současné době systém vyvíjejí vývojáři z PostgreSQL Global Development Group a podléhá licenci PostgreSQL License, což je svobodná open-source licence podobná BSD nebo MIT licencím [Poh(2014)].

Hlavní výhodou PostgreSQL je stejně jako u MySQL jeho **multiplatformnost**. Tento systém je možné provozovat na většině Unixových systémů včetně Linuxu a také na platformě Microsoft Windows. Další značnou výhodou oproti jiným databázovým systémům je **spolehlivost** a **stabilita** tohoto SŘBD. Neposlední a také nespornou výhodou je **možnost běhu uživatelských funkcí v několika programovacích jazycích**, mezi které patří

[1] **open source** – software s volně dostupnými zdrojovými kódy

kromě SQL také jazyky C, Perl, Python nebo speciální PL/pgSQL, který vychází z PL/SQL od firmy Oracle.

Zajímavou vlastností toho systému je **dědičnost**. Systém dovoluje oddědit od rodičovské tabulky některé vlastnosti, což se v praxi uplatňuje především na dědění sloupců z rodičovské tabulky do tabulky potomka [Poi(2014)].

6.1.3 Microsoft SQL Server

První verze databázového systém Microsoft SQL Server byla uvedena na trh v roce 1989 a byla určena pouze pro operační systém OS/2 od firmy IBM. Na tehdejším vývoji tohoto systému se podílely kromě firmy Microsoft také firma Sybase a firma Ashton-Tate. V roce 1992 s nástupem operačního systému Windows NT 3.1, vyšel i databázový systém pro tuto platformu. Následně začal Microsoft databázový systém vyvíjet samostatně bez spolupracujících firem. Kromě SQL serveru tato firma vydala ještě dva populární databázové systémy s názvy Microsoft Access a Microsoft FoxPro. [You(2012)]

Tento SŘBD je nabízen v několika komerčních edicích a také v jedné bezplatné edici s názvem Express. Tato edice je vyvíjena pro snadnou a spolehlivou spolupráci s platformou Microsoft .NET a obsahuje oproti jiným databázovým systémům značná omezení. Jedním z omezení je **velikost databáze** (maximální velikost je 10GB) a druhým omezením je **využívání pouze 1 GB operační paměti**. Do nevýhod tohoto sytému lze zařadit i nemožnost provozování na jiných systémech než je Microsoft Windows.

Naopak výhodou toho systému je jazyk Transact-SQL, který stejně jako několik jazyků u PostgreSQL, umožňuje definovat uživatelské funkce pomocí procedurálního programování. Další výhodou tohoto systému je, jak již bylo uvedeno výše, **výborná kompatibilita** s platformou Microsoft .NET. Mezi výhody můžeme podle [Mic(2012)] zařadit i **vysokou úroveň zabezpečení** a také **výkonnost** tohoto systému.

6.2 Zvolený databázový systém

Ze tří výše uvedených vhodných databázových systémů byl i po poradě se zadavatelem vybrán databázový systém PostgreSQL. Hlavním důvodem pro

výběr tohoto systému byl fakt, že server zadavatele běží na operačním systému Linux a proto Microsoft SQL Server nepřipadal v úvahu. Při rozhodování mezi MySQL a PostgreSQL byla brána v úvahu především výkonnost při složitějších a náročnějších operacích. Dále také skutečnost, že PostgreSQL je více propracovanější z hlediska transakcí a integrity dat. Rovněž byla brána v úvahu výše zmíněná spolehlivost a stabilita. Ve prospěch vybraného databázového sytému, hrála také méně důležitá kritéria jako výborně propracovaná dokumentace nebo uživatelsky přívětivé grafické administrační rozhraní pro správu databáze s názvem pgAdmin, které obsahuje i českou lokalizaci.

6.3 Návrh relačního databázového modelu

V této části práce se budeme zabývat návrhem samotné databáze systému. Databáze byla vytvořena na základě požadavků zadavatele a obsahuje 9 entit. Tyto entity, jejich atributy a vztahy mezi těmito entitami si dále stručně popíšeme.

Schéma každé entity bude popsáno tabulkou. Každá tabulka obsahuje název atributu, datový typ a stručný popis. Dále pak sloupec s označením **„Klíč“**, který určuje, zda daný atribut figuruje jako primární (PK) nebo cizí (FK) klíč a také sloupec s označením **„Null“** určující, zda-li příslušný atribut může být prázdný (Ano) nebo nikoli (Ne).

Pro přehlednost je v příloze C uveden relační model databáze, který mimo jiné obsahuje také definice primárních a cizích klíčů, povinné položky každé tabulky a v některých tabulkách také unikátní položky.

6.3.1 Tabulka `Sklad`

Tato tabulka obsahuje informace o všech evidovaných a spravovaných skladech. Jednotlivé atributy jsou uvedeny v tabulce 6.1.

Název atributu	Datový typ	Popis	Klíč	Null
ID_skladu	integer	identifikátor skladu	PK	Ne
Nazev_skladu	varchar (50)	název skladu	–	Ne
Mesto_ID_mesta	integer	identifikátor města, ve kterém sklad sídlí	FK	Ne
Ulice	varchar (50)	ulice, ve které sklad sídlí	–	Ne
Zpusob_prumerovani	varchar (15)	typ způsobu průměrování skladu (Periodický, Proměnlivý)	–	Ne
Spravovan	boolean	určuje, zda-li je sklad aktuálně spravován	–	Ne

Tabulka 6.1: Atributy tabulky `Sklad`

6.3.2 Tabulka `Uzivatel`

Tato tabulka slouží k evidenci uživatelů a také obsahuje informace potřebné pro přihlášení uživatele. Atributy této entity jsou zobrazeny v tabulce 6.2.

Název atributu	Datový typ	Popis	Klíč	Null
ID_uzivatele	integer	identifikátor uživatele	PK	Ne
Login	varchar (35)	přihlašovací jméno uživatele	–	Ne
Heslo	varchar (35)	heslo uživatele	–	Ne
Jmeno	varchar (50)	jméno uživatele	–	Ne
Prijemni	varchar (50)	příjmení uživatele	–	Ne

Tabulka 6.2: Atributy tabulky `Uzivatel`

6.3.3 Tabulka `Spravuje`

Tabulka `Spravuje` slouží jako spojovací tabulka, která spojuje tabulky `Sklad` a `Uzivatel`. Tato entita nám definuje, jaké sklady mohou spravovat příslušní uživatelé. Schéma této entity můžeme vidět v tabulce 6.3

Název atributu	Datový typ	Popis	Klíč	Null
Sklad_ID_skladu	integer	identifikátor skladu	PK, FK	Ne
Uzivatel_ID_uzivatele	integer	identifikátor uživatele	PK, FK	Ne

Tabulka 6.3: Atributy tabulky `Spravuje`

6.3.4 Tabulka `Mesto`

Tabulka `Mesto` obsahuje informace o městech, resp. názvy měst a slouží především k lokalizaci skladů a dodavatelů. V tabulce 6.4 jsou popsány atributy této entity.

Název atributu	Datový typ	Popis	Klíč	Null
ID_mesta	integer	identifikátor města	PK	Ne
Nazev	varchar (50)	název města	–	Ne

Tabulka 6.4: Atributy tabulky `Mesto`

6.3.5 Tabulka `Dodavatel`

Tato tabulka eviduje dodavatele materiálů. Zde je nutné podotknout, že tato evidence je značně zjednodušená a to zejména z toho důvodu, že informace o dodavatelích na dokladech jsou pouze informativní a případné bližší informace nejsou pro zadavatele podstatné. V následující tabulce (tabulka 6.5) jsou popsány atributy této entity.

Název atributu	Datový typ	Popis	Klíč	Null
ID_dodavatele	integer	identifikátor dodavatele	PK	Ne
ICO	integer	identifikační číslo organizace dodavatele	–	Ne
Nazev	varchar (50)	název dodavatele	–	Ne
Mesto_ID_mesta	integer	identifikátor města, kde se nachází hlavní sídlo dodavatele	FK	Ne

Tabulka 6.5: Atributy tabulky `Dodavatel`

6.3.6 Tabulka `Doklad`

Tato tabulka je jednou z nejdůležitějších tabulek celé databáze. Tabulka obsahuje evidenci všech dokladů vztahujících se k danému skladu a určuje veškeré pohyby na skladě. Atributy této entity jsou charakterizovány v tabulce 6.6.

Název atributu	Datový typ	Popis	Klíč	Null
ID_dokladu	integer	identifikátor dokladu	PK	Ne
Cislo_dokladu	integer	číslo dokladu	–	Ne
Datum	date	datum dokladu	–	Ne
Typ_dokladu	varchar (20)	typ dokladu (Příjem, Příjem dopravy, Výdej do spotřeby, Výdej – prodej)	–	Ne
Uzivatel_ID_uzivatele	integer	identifikátor uživatele, ke zjištění, kdo spravoval daný doklad	FK	Ne
Dodavatel_ID_dodavatele	integer	identifikátor dodavatele, pro informaci o dodavateli na dokladu	FK	Ano
Sklad_ID_skladu	integer	identifikátor skladu, pro zjištění k jakému skladu doklad patří	FK	Ne
Moznost_upravy	boolean	informace o možnosti změny položek dokladu	–	Ne

Tabulka 6.6: Atributy tabulky `Doklad`

6.3.7 Tabulka `Polozka`

Tato entita reprezentuje jednotlivé položky dokladu. Schéma tabulky definuje atributy potřebné pro všechny typy dokladů. Přehled těchto atributů je uveden v tabulce 6.7.

Název atributu	Datový typ	Popis	Klíč	Null
ID_polozky	integer	identifikátor položky	PK	Ne
Mnozstvi	double	množství materiálu dané položky	–	Ano
Celkova_cena	double	celková cena materiálu položky	–	Ne
Vcetne_dopravy	boolean	informuje, zda-li je celková cena položky včetně dopravy	–	Ano
Material_ID_materialu	integer	identifikátor materiálu, který udává k jakému materiálu se položka vztahuje	FK	Ne
Doklad_ID_dokladu	integer	identifikátor dokladu, určuje k jakému dokladu se váže položka	FK	Ne

Tabulka 6.7: Atributy tabulky `Polozka`

6.3.8 Tabulka `Material`

Tato tabulka eviduje jednotlivé materiály v rámci skladu a informace o těchto materiálech. Jednotlivé atributy této entity vyobrazuje tabulka 6.8

Název atributu	Datový typ	Popis	Klíč	Null
ID_materialu	integer	identifikátor materiálu	PK	Ne
Kod_materialu	integer	kód materiálu, který je unikátní vzhledem ke skladu	–	Ne
Nazev	varchar (50)	název materiálu	–	Ne
Merna_jednotka	varchar (5)	měrná jednotka materiálu (kg, t, l, ks)	–	Ne
Sklad_ID_skladu	integer	identifikátor skladu, který udává k jakému skladu se materiál vztahuje	FK	Ne

Tabulka 6.8: Atributy tabulky `Material`

6.3.9 Tabulka `Prumer`

Tabulka `Prumer` je důležitou tabulkou pro sklady s periodickým průměrováním. V tabulce jsou uchovávány periodické vážené průměry všech materiálů v daných obdobích, které se využívají při výdejích materiálů. Tabulka 6.9 popisuje jednotlivé atributy této entity.

Název atributu	Datový typ	Popis	Klíč	Null
ID_prumeru	integer	identifikátor průměru	PK	Ne
Sklad_ID_skladu	integer	identifikátor skladu, který určuje, ke kterému skladu se průměr vztahuje	FK	Ne
Material_ID_materialu	integer	identifikátor materiálu, který určuje, ke kterému materiálu se průměr vztahuje	FK	Ne
Mnozstvi_k_datu	double	množství materiálu v daném období	–	Ne
Prumerna_cena_k_datu	double	průměrná cena materiálu v daném období	–	Ne
Datum	date	datum určující období, ke kterému se vztahuje daný průměr	–	Ne

Tabulka 6.9: Atributy tabulky `Prumer`

Stejně jako tabulka `Spravuje` sloužila tato entita jako spojovací tabulka s názvem Skladuje, kde primární klíč tvořily atributy *Sklad_ID_skladu*, *Material_ID_materialu* a atribut *Datum*. Později byl ovšem vytvořen umělý primární klíč *ID_prumeru* a to především z důvodu jednoduššího přístupu k záznamům tabulky, protože přistupovat k záznamům, zejména pomocí data, není příliš vhodné. Následně pak byla tabulka přejmenována na současný název, protože lépe odpovídá obsahu tabulky.

6.3.10 Popis vztahů databáze

V této části práce si stručně popíšeme vztahy mezi jednotlivými entitami databáze. U vztahů nebudou uváděny kardinality, protože jsou uvedeny v databázovém modelu (příloha C) a není nutné je zde uvádět znovu.

- vztah `Sklad - Mesto` – slouží k identifikaci polohy skladu. Ze vztahu vyplývá, že každý Sklad sídlí právě v jednom městě a také, že v jednom městě může sídlit jeden nebo několik skladů.
- vztah `Sklad - Material` – určuje, jaký materiál je skladován v příslušném skladu. Sklad může skladovat jeden nebo více materiálů, materiál je skladován právě jedním skladem.
- vztah `Sklad - Uzivatel` – typu M:N, spojuje spojovací tabulka `Spravuje` a definuje, jaké sklady může spravovat příslušný uživatel. Každý uživatel spravuje jeden nebo více skladů, nemusí ale spravovat žádný. Oproti tomu každý sklad může spravovat jeden nebo více uživatelů, případně ho nemusí spravovat nikdo.
- vztah `Sklad - Prumer` – definuje, jaké průměrné ceny materiálů v daném období obsahuje daný sklad. Každý sklad může obsahovat jeden nebo několik průměrů a každý průměr materiálu se musí vztahovat právě k jednomu skladu.
- vztah `Sklad - Doklad` – říká, jaký materiál je možno skladovat v daném skladu. Materiál musí být skladován právě v jednom skladu, ale sklad musí skladovat jeden nebo více materiálů, případně nemusí skladovat žádný materiál.
- vztah `Dodavatel - Mesto` – slouží k identifikaci hlavního sídla dodavatele. I když by bylo vhodné, aby tato relace byla typu M:N (každý dodavatel může sídlit ve více městech), je podle požadavků zadavatele zjednodušena pouze na určení hlavního sídla dodavatele. V každém městě může tedy sídlit jeden nebo více dodavatelů, ale každý dodavatel musí sídlit právě v jednom městě.
- vztah `Dodavatel - Doklad` – udává, jaký dodavatel dodal materiál, který je obsažen na dokladu. Doklad obsahuje jednoho nebo žádného dodavatele, dodavatel je obsažen na jednom nebo více dokladech, případně není obsažen na žádném dokladu.

- vztah `Doklad - Uzivatel` – říká, jaký uživatel zaúčtoval příslušný doklad. Doklad musí být zaúčtován právě jedním uživatelem, ale uživatel může zaúčtovat jeden nebo více dokladů.
- vztah `Doklad - Polozka` – definuje, které položky obsahuje daný doklad. Položka musí patřit právě jednomu dokladu, doklad může obsahovat jednu či více položek, případně může být bez položek.
- vztah `Polozka - Material` – určuje, jaký materiál je uveden v položce dokladu. Materiál může být uveden na jedné nebo několika položkách, ale každá položka musí obsahovat právě jeden materiál.
- vztah `Material - Prumer` – určuje, k jakému materiálu se vztahuje daný průměr v období. Každý materiál může obsahovat jeden či více průměrů, případně žádný. Oproti tomu každý průměr se musí vztahovat právě k jednomu materiálu.

7 Implementace

V této kapitole si objasníme technologie a knihovny, které byly použity při vývoji této práce a následně uvedeme technické požadavky pro běh aplikace a databázového systému.

7.1 Použité technologie

Při vývoji této práce byly použity 3 technologie, mezi které patří **Microsoft .NET Framework**, **jazyk C#** a systém řízení báze dat **PostgreSQL**. Tento databázový systém byl podrobněji popsán v sekci 6.1.2 a proto ho zde již nebudeme dále popisovat, pouze si uvedeme, že v této práci byl použit tento databázový systém ve verzi 9.3. Zbylé dvě technologie si následně stručně charakterizujeme.

7.1.1 Microsoft .NET Framework

Microsoft .NET Framework je nejrozšířenější platforma pro operační systém Microsoft Windows. Tato platforma se skládá především z **běhového systému** (Common Language Runtime) a **základních knihoven tříd systému** (Basic Class Library). Common Language Runtime, zkráceně CLR, je společné běhové prostředí zajišťující běh programů přeložených z různých programovacích jazycích do CIL[1] [Vir(2002)]. Basic Class Library, zkráceně BCL, je rozsáhlá knihovna tříd, hodnotových typů a rozhraní. Umožňuje například práci se soubory, práci s komunikačními protokoly nebo podporu různých národních zvyklostí.

Dále pak tato platforma obsahuje **knihovny pro tvorbu grafického uživatelského rozhraní** a **knihovny pro tvorbu webových služeb**. Poslední část tohoto frameworku reprezentují **překladače jazyků kompatibilních s platformou .NET**. Mezi tyto jazyky patří například C#, Visual Basic .NET, C++/CLI, F# nebo jazyk J#.

[1] Common Intermediate Language (CIL) – je nízkoúrovňový programovací jazyk, podobný jazyku symbolických adres, který je jedním ze součástí umožňujících spolupráci jazyků kompatibilních s platformou .NET.

Platforma .NET Framework je dostupná v několika verzích. Poslední verzí této platformy je verze 4.5.1. V této práci byla použita verze 4.5, která se od nejnovější verze liší pouze několika detaily, které nemají na výsledek práce vliv.

7.1.2 Jazyk C#

Jazyk C# je vysokoúrovňový objektově orientovaný jazyk, který vyvinula firma Microsoft na základě jazyků C++ a Java. Tento jazyk je navržen pro použití s platformou .NET Framework a je považován jako hlavní jazyk této platformy. Podle [Nag(2009), s. 44] je tento jazyk samostatný a není součástí této platformy. Většina funkcí tohoto jazyka je umožněna právě za použití již zmíněné platformy. Současnou verzí je verze 5.0, která byla použita i v této práci. Mezi důležité vlastnosti tohoto jazyka při použití s platformou .NET patří **silná typová bezpečnost**, **řízený kód** (kontrola přetečení, správa paměti pomocí garbage collectoru, ...), **citlivost na velká a malá písmena**, **jednoduchá dědičnost**, **nullable typy** nebo možnost použití tzv. **property** místo klasických getrů a setrů.

7.2 Použité knihovny

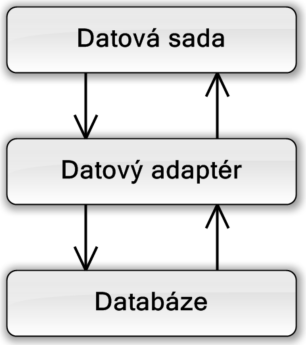
Při implementaci této práce byly použity knihovny **ADO.NET**, **iTextSharp** a knihovna **Npgsql**. Tyto knihovny přispěly nedílnou měrou ke zkvalitnění aplikace. V této části práce si tyto knihovny jednotlivě popíšeme.

7.2.1 ADO.NET

Knihovna ADO.NET (ActiveX Data Objects) představuje soubor komponent používaných pro přístup k datům či datovým službám v různých datových zdrojích a je součástí platformy Microsoft .NET Framework. Tato knihovna vznikla na základně nespokojenosti s technologií ADO, která je knihovnou komponent COM (Component Object Model) [Nag(2009), s. 936]. ADO.NET obsahuje především 4 důležité vlastnosti. Možnost **práce s daty z různých datových zdrojů**, **podpora vícevrstvých aplikací**, možnost **použití jak na online tak na offline aplikacích** a také **podpora práce s XML**. Právě

dvě poslední vlastnosti byly využity v této práci nejvíce.

Knihovna obsahuje třídu pro práci s daty pojmenovanou `DataSet`. Jedná se o datovou sadu, která obsahuje nejenom kolekci tabulek včetně jejich omezení, ale také relace mezi těmito tabulkami. Díky těmto datovým sadám je možné uchovávat data, modifikovat je a zpracovávat i bez připojení do databáze. Pro naplnění datových sad slouží třída `DataAdapter`, která se o připojení k databázi a stažení dat stará sama. Tato třída také umožňuje modifikace dat v datových sadách celkem pohodlně přenést do databáze. Schéma spolupráce těchto dvou tříd a databáze lze vidět na obrázku 7.1.



Obrázek 7.1: Schéma spolupráce datových sad, datových adaptérů a databáze [Aga(2009), s. 248]

Datové sady byly v práci použity zejména pro **zálohování** a **obnovování dat**, kdy bylo využito i výše zmíněné podpory práce s XML. Dále byly také využity pro **získávání dat z databáze**. Pro naplňování těchto sad ovšem nebyla využívána třída `DataAdapter`, ale třída `NpgsqlDataAdapter` z knihovny Npgsql. Tato třída implementuje rozhraní `IDataAdapter` a je uzpůsobena pro lepší komunikaci se zvoleným databázovým systémem.

7.2.2 Npgsql

Npgsql neboli .Net Data Provider for Postgresql je knihovna vyvinutá pro platformu Microsoft .NET Framework z důvodu zkvalitnění komunikace s databázovým systémem PostgreSQL. Tato knihovna je stejně jako PostgreSQL open source a je realizována v jazyce C#. Npgsql zajišťuje velké množství funkcí od samotného spojení s databází až po realizaci databázové transakce. V této práci byla použita verze 2.0.14.3, které dokáže komunikovat s PostgreSQL verze 7 a vyšší.

7.2.3 iTextSharp

iTextSharp je knihovna pro práci s PDF dokumenty, určená pro platformu Microsoft .NET Framework, která vznikla z knihovny iText. Tato knihovna je napsána v jazyce C# a je šířena pod licencí Affero General Public License. iTextSharp obsahuje mnoho funkcí pro práci s PDF dokumenty mezi které patří například generování a upravování těchto dokumentů, možnost rozdělovat a spojovat tyto dokumenty, automatické vyplňování PDF formulářů nebo přidávání digitálního podpisu do těchto souborů. V této práci je použita tato knihovna ve verzi 5.5.0 a je používána k **exportu** všech sestav do PDF souboru a také k jejich **tisku**.

7.3 Technické požadavky

Vytvořená aplikace není nijak hardwarově náročná a její minimální požadavky na funkčnost udávají spíše použité technologie. Tyto požadavky se vztahují především k platformě .NET Framework verze 4.5. Protože tato platforma je určena pouze pro systém Microsoft Windows, je aplikace schopna běhu pouze na tomto systému. Aplikace by měla splňovat bezproblémový běh za předpokladu, že na tomto systému bude nainstalována výše zmíněná verze frameworku, která je kompatibilní pouze se systém Microsoft Windows Vista SP2 a novějším. Mimo to by měla být aplikace schopna běhu jak na 32 bitové, tak na 64 bitové verzi tohoto systému.

Databázový systém také neklade žádné velké nároky na funkčnost. Jak bylo uvedeno v sekci 6.1.2, SŘBD PostgreSQL je schopný běhu na většině předních operačních systémů, mezi které patří většina unixových systémů včetně Linuxu a také systém Windows. Zároveň je schopný běhu jak na 32 bitových, tak na většině 64 bitových verzích těchto systémů. Můžeme tedy říci, že minimální požadavky na funkčnost databázového systému v podstatě odpovídají minimálním požadavkům těchto operačních systémů. V zásadě je, ale nutné předpokládat počet uživatelů, který bude k databázovému systému přistupovat a také možnou velikost databáze. Pro bezproblémový běh je pak vhodné tato kritéria vzít v úvahu a podle nich volit jak hardwarové, tak softwarové parametry serveru, na kterém databázový systém poběží.

8 Testování aplikace

V této kapitole si popíšeme jak probíhalo testování a jakých bylo dosaženo výsledků. Následně zhodnotíme stav aplikace a možnosti rozšíření do budoucna. Testování probíhalo ve dvou fázích. V první fázi byla testována správnost výpočtu při oceňování položek a ve druhé fázi byly ověřovány ostatní funkcionality systému.

8.1 Testování ekonomický operací

Cílem tohoto testování bylo ověření správného oceňování cen v daném období a také ověření správného příjmání a vydávání materiálu. Ověřování probíhalo na základě testovacích dat, které poskytl zadavatel. Protože si zadavatel nepřeje zveřejnění těchto dat, je v příloze D alespoň jejich ukázka. Testování probíhalo ve třech částech a každá část byla ověřována samostatně.

V první části byly nejprve napříjmány počáteční stavy tří druhů materiálů v období jednoho měsíce (příl. D.1). Následně byly napříjmány doklady k těmto třem druhům materiálů i za další období a to včetně různých kombinací s příjmem dopravy (příl. D.2). Výsledná kontrola množství a hodnoty materiálu na skladě ukázala prakticky totožné výsledky s testovacími daty.

Druhá část testování obnášela uzavření posledního období skladu a porovnání průměrných cen s testovacími daty. Průměrné ceny materiálů téměř odpovídaly průměrným cenám v testovacích datech. Tyto výsledky se lišily v jednotkách, maximálně desítkách haléřů a odchylka byla způsobena zaokrouhlovacími chybami.

Třetí část testování byla založena na výdejích materiálu. V této části bylo od každého materiálu vydáváno různé množství na několika dokladech (příl. D.2). Při kontrole výsledků se ukázalo, že množství materiálu je téměř totožné, ale hodnota vydaného materiálu se liší od testovacích dat v desítkách haléřů. I tato odchylka byla způsobena zaokrouhlovacími chybami, ovšem tyto chyby nebyly zapříčiněny jen zaokrouhlováním celkových cen při výdeji, ale větší měrou se na ní podílely zaokrouhlovací chyby při výpočtech průměrných cen. Z tohoto důvodu pak celková chyba lineárně narůstá s množstvím vydávaného materiálu.

8.2 Testování ostatních funkcionalit systému

Tato fáze testování byla založena na kontrole ostatních funkcionalit systému, mezi které patří reakce na ztrátu spojení s databázovým systémem a ověření správné funkčnosti při souběžném přístupu ke správě jednoho skladu. Reakce na ztrátu spojení s databázovým systémem byla testována při většině ekonomických operací, při přihlašování, výběru skladu a také u generování všech sestav. Aplikace při těchto testech byla stabilní a i při ekonomických operacích byla zachována konzistence dat v databázi, ale v grafickém uživatelském rozhraní vyskakovalo příliš mnoho chybových hlášení. Na základě tohoto testování byla tato hlášení nahrazena pouze jednou chybovou hláškou a ztrátě spojení.

Testování souběžného přístupu k jednomu skladu bylo testováno na dvou souběžně běžících procesech, které ve smyčce vkládaly a upravovaly data. Každý proces prováděl vkládání, úpravy a mazání dokladů. Výsledky tohoto testování ukázaly, že i při souběžném přístupu jsou data v databázovém systému konzistentní. Tento výsledek byl víceméně předpokládán z důvodu použití databázových transakcí u těchto operací.

8.3 Stav systému a možnosti rozšíření

V současné době byl systém předán zadavateli a byl uveden do testovacího provozu. Od uvedení do testovacího provozu zatím nebyly zaznamenány chyby a systém funguje bez problémů i při zatížení více uživateli. Reakce uživatelů na přívětivost aplikace a běh systému jsou ve většině případů pozitivní. Nový skladový systém je podle názorů uživatelů oproti stávajícímu stabilnější a komfortnější. Většina uživatelů výměnu starého systému za nový uvítala.

Aplikace obsahuje prozatím pouze uživatelskou část a je nutné informace o uživatelích a skladech přidávat přímo prostřednictvím databázového systému. Z tohoto důvodu bude vznikat administrátorské prostředí, kde bude možno provádět nejen tyto operace, ale i možnosti úprav a mazání ostatních informací v databázi. Zároveň tato část bude obsahovat možnosti zálohy celé databáze a bude umožněno alespoň částečně provádět údržbu databáze. Skladový systém by měl být do budoucna rozšířen o možnost proměnlivého způsobu průměrování a o možnost generování příjemek/výdejek, která byla uvedena v sekci 4.3.

9 Závěr

Tato bakalářská práce se zabývala skladovými systémy a hlavním cílem této práce bylo vytvořit nový skladový systém využívající vhodný relační databázový systém, který je přizpůsoben potřebám zadavatele. Pro dosažení tohoto cíle bylo stanoveno několik menších cílů.

Prvním z těchto cílů bylo analyzovat problematiku skladových systémů. Tento cíl spočíval především v analýze samotných zásob a jejich oceňování. Dále pak do tohoto bodu spadala analýza stávajícího systému a také analýza jiných systémů. Tento bod přinesl rozhled v oblasti skladové evidence a umožňoval přiblížit jak stávající skladovým systémem, tak různé další skladové programy.

Druhým cílem bylo navrhnout možná vylepšení stávajícího skladového systému. Na základě analýzy systémů bylo stanoveno sedm potřebných vlastností, které byly zahrnuty do nového systému a čtyři vlastnosti, které bylo vhodné začlenit do nového systému z důvodu zkvalitnění funkčnosti. Z těchto vhodných vlastností byla zahrnuta pouze možnost úpravy příjemek/výdejek. Možnost generování příjemek/výdejek bude zahrnuta až do administrátorské části systému a zbylé dvě možnosti byly na základě domluvy se zadavatelem zavrhnuty jako nepotřebné.

Do třetího cíle spadal samotný návrh nového systému včetně návrhu databáze. Při návrhu systému byla stanovena struktura aplikace a systém byl rozdělen do třívrstvé architektury. Na základě této struktury pak bylo navrženo jádro aplikace a grafické uživatelské rozhraní. Při návrhu databáze byl nejprve vybrán vhodný databázový systém a následně byl proveden samotný návrh relačního modelu databáze, přizpůsobeného požadavkům zadavatele.

Posledním cílem byla samotná implementace systému a následné otestování. Pro implementaci byly použity různé technologie a knihovny a následně stanoveny technické požadavky vytvořeného systému. Při testování byla ověřena správná funkčnost ekonomických operací a také ostatních funkcionalit systému. Na základě výsledků testování byly provedeny drobné korekce a následně byl stanoven stav systému včetně jeho možných rozšíření.

Na základě splnění těchto jednotlivých cílů byl splněn hlavní cíl a výstupem práce je nový skladový systém vzniklý pro účely zadavatele. Tento systém bude dále rozšířen o administrátorskou část, jak je uvedeno v sekci 8.3.

Seznam zkratek

.NET – „dotnet“ pochází z anglického slova network a představuje soubor technologií, které tvoří platformu (podrobněji popsáno v sekci 7.1.1).

ADO (ActiveX Data Objects) – je knihovna, která slouží pro manipulaci s databází pomocí rozhraní ODBC a je předchůdcem ADO.NET.

ADO.NET (ActiveX Data Objects .NET) – je soubor komponent používaných pro přístup k datům či datovým službám v různých datových zdrojích a je součástí platformy Microsoft .NET Framework (podrobněji popsáno v sekci 7.2.1).

AGPL (Affero General Public License) – je licence svobodného softwaru od společnosti Affero, která vznikla na základě licence GPL pro řešení podmínek ohledně užití softwaru přes síť.

BCL (Basic Class Library) – je rozsáhlá knihovna tříd, hodnotových typů a rozhraní v platformě Microsoft .NET Framework.

BSD (Berkeley Software Distribution) – je operační systém odvozený od Unixu distribuovaný Kalifornskou univerzitou v Berkeley.

CIL (Common Intermediate Language) – je nízkoúrovňový programovací jazyk, podobný jazyku symbolických adres, který je jedním ze součástí umožňujících spolupráci jazyků kompatibilních s platformou .NET.

CLR (Common Language Runtime) – je společné běhové prostředí zajišťující běh programů přeložených z různých programovacích jazycích do CIL.

COM (Component Object Model) – je standard, který definuje způsob, jakým spolu mohou softwarové komponenty komunikovat.

CSV (Comma-separated values) – v překladu hodnoty oddělené čárkami, je formát souboru pro tabulková data, kde každá hodnota na řádce je oddělena čárkou případně středníkem.

Demo (Demoverze) – volně dostupná verze komerčního software, která obsahuje pouze část funkcionality plné verze systému.

DOS (Disk Operating System) – rodina jednouživatelských a jednoúlohových operačních systémů určených pro procesory Intel 8088.

FIFO (First In First Out) – je v účetnictví metoda pro oceňování zásob při vyskladnění (podrobněji popsáno v sekci 2.3.2).

FK (Foreign Key) – neboli cizí klíč je v podstatě primární klíč jiné tabulky, který slouží k definici vztahů mezi tabulkami.

GUI (Graphical User Interface) – neboli grafické uživatelské rozhraní je rozhraní umožňující ovládat počítač pomocí grafických prvků.

IBM (International Business Machines Corporation) – je jednou z největších světových společností v oboru informačních technologií.

IT (Informační Technologie) – technické odvětví, zabývající se hardwarovou a softwarovou stránkou počítače.

LAMP (Linux, Apache, MySQL, PHP) – je softwarový balíček využívaný pro tvorbu webových aplikací.

LIFO (Last IN First Out) – je v účetnictví metoda pro oceňování zásob při vyskladnění, která je opačná než metoda FIFO.

MIT licence (Massachusetts Institute of Technology) – je svobodná licence, která vznikla v Massachusettském technologickém institutu v Cambridge.

MS-DOS (Microsoft Disk Operating System) – operační systém od firmy Microsoft, který byl jako první určen pro jednoduchou obsluhu osobních počítačů.

Npgsql (.Net Data Provider for Postgresql) – je knihovna vyvinutá pro platformu Microsoft .NET Framework z důvodu zkvalitnění komunikaci s databázovým systémem PostgreSQL (podrobněji popsáno v sekci 7.2.2).

OMG (Object Management Group) – je sdružení, které rozšiřuje a dohlíží na specifikaci jazyka UML.

OS/2 – je operační systém firmy IBM.

PDF (Portable Document Format) – je formát souboru od firmy Adobe, nezávislý na platformě, který byl vyvinut pro jednoduchou přenositelnost dokumentů.

PHP (PHP: Hypertext Preprocessor původně Personal Home Page) – je programovací jazyk, určený především k programování dynamických webových aplikací.

PK (Primary Key) – neboli primární klíč, je atribut případně několik atributů databázové tabulky, který jednoznačně identifikuje každý záznam.

PL/pgSQL (Procedural Language/PostgreSQL) – je procedurální programovací jazyk určený pro databázový systém PostgreSQL, který je podobný jazyku PL/SQL.

PL/SQL (Procedural Language/Structured Query Language) – je procedurální programovací jazyk od firmy Oracle,který je nadstavbou jazyka SQL a vznikl z programovacího jazyka Ada.

SP (Service pack) – je balík obsahující opravy, aktualizace nebo vylepšení software.

SQL (Structured Query Language) – je standardizovaný strukturovaný dotazovací jazyk používaný v databázových systémech.

SŘBD (Systém řízení báze dat) – počítačový systém zabezpečující definování struktury dat, možnost manipulace s daty, ochranu dat a také komunikaci systému s uživatelem.

UML (Unified Modeling Language) – je jednotný modelovací jazyk, který slouží k návrhu, specifikaci či vizualizaci softwarových systémů při jejich vývoji (podrobněji popsáno v sekci 5.3.1).

XLS – je binární formát od firmy Microsoft využívaný pro soubory vytvořené v aplikaci Microsoft Excel.

XML (Extensible Markup Language) – je obecný standard značkovacího jazyka.

Literatura

[Aga(2009)] AGARWAL, Vidya Vrat a James HUDDLESTON. *Databáze v C# 2008: průvodce programátora*. Vyd. 1. Brno: Computer Press, 2009, 424 s. ISBN 978-80-251-2309-6.

[Akt(2014)] Slovníček účetních pojmů: Aktivace. In: *Testy z účetnictví* [online]. 2014 [cit. 2014-04-14]. Dostupné z: `http://www.testyzucetnictvi.cz/slovnicek-ucetnich-pojmu.php?pojem=aktivace`

[Awi(2013)] A.W.I.S. SPRÁVA, SYSTÉMY S.R.O. *AWIS pokladní systém* [software]. 2013 [cit. 2014-04-14]. Dostupné z: `http://www.kasa-pokladna.cz/zkusebni-verze`

[Bán(2002)] BÁNOVSKÝ, Milan. *Skladík 5.5.5* [software]. 2002 [cit. 2014-04-14]. Dostupné z: `http://www.mbsw.cz/`

[Car(2010)] CARDOVÁ, Zdenka, Ing. Zásoby: (7. část). In: *Mzdová praxe* [online]. 1.12.2010 [cit. 2014-04-14]. Dostupné z: `http://www.mzdovapraxe.cz/archiv/dokument/doc-d28530v35681-zasoby-7-cast/`

[Čáp(2014)] ČÁPKA, David. 1. díl - Úvod do UML. In: *Devbook.cz: programátorská sociální síť* [online]. 2014 [cit. 2014-04-19]. Dostupné z: `http://www.devbook.cz/uml-uvod-historie-vyznam-a-diagramy`

[Kuč(2007)] KUČEROVÁ, Helena. Diagram tříd. *Vyšší odborná škola informačních služeb* [online]. 31. 3. 2007 [cit. 2014-04-19]. Dostupné z: `http://web.sks.cz/users/ku/pri/tridy.htm`

[Man(2011)] Třívrstvá architektura (Three-tier architecture). *ManagementMania.com* [online]. 2011 [cit. 2014-04-18].

Dostupné z: `https://managementmania.com/cs/trivrstva-architektura-three-tier-architecture`

[Mic(2012)] Proč SQL Server. *Microsoft: Microsoft SQL Server* [online]. 2012 [cit. 2014-04-16]. Dostupné z: `http://www.microsoft.com/sqlserver/cs/cz/product-info/why-sql-server.aspx`

[Mya(2014)] About. *MySQL.com* [online]. 2014 [cit. 2014-04-16]. Dostupné z: `http://www.mysql.com/about/`

[Myp(2014)] Products. *MySQL.com* [online]. 2014 [cit. 2014-04-16]. Dostupné z: `http://www.mysql.com/products/`

[Nag(2009)] NAGEL, Christian, Bill EVJEN, Jay GLYNN, Karli WATSON a Morgan SKINNER. *C# 2008: programujeme profesionálně*. Vyd. 1. Brno: Computer Press, 2009, 1126 s. ISBN 978-80-251-2401-7.

[Omg(2014)] Unified Modeling Language. *OMG.org* [online]. 2014 [cit. 2014-04-19]. Dostupné z: `http://www.omg.org/spec/UML/`

[Poh(2014)] About: History. *PostgreSQL.org* [online]. 2014 [cit. 2014-04-16]. Dostupné z: `http://www.postgresql.org/about/history/`

[Poi(2014)] Documentation: 5.8. Inheritance. *PostgreSQL.org* [online]. 2014 [cit. 2014-04-16]. Dostupné z: `http://www.postgresql.org/docs/9.3/static/ddl-inherit.html`

[Ryn(2010)] RYNEŠ, Petr. *Podvojné účetnictví a účetní závěrka 2010* 10. vyd. Olomouc: ANAG, 2010, s. 122. ISBN 978-80-7263-580-1.

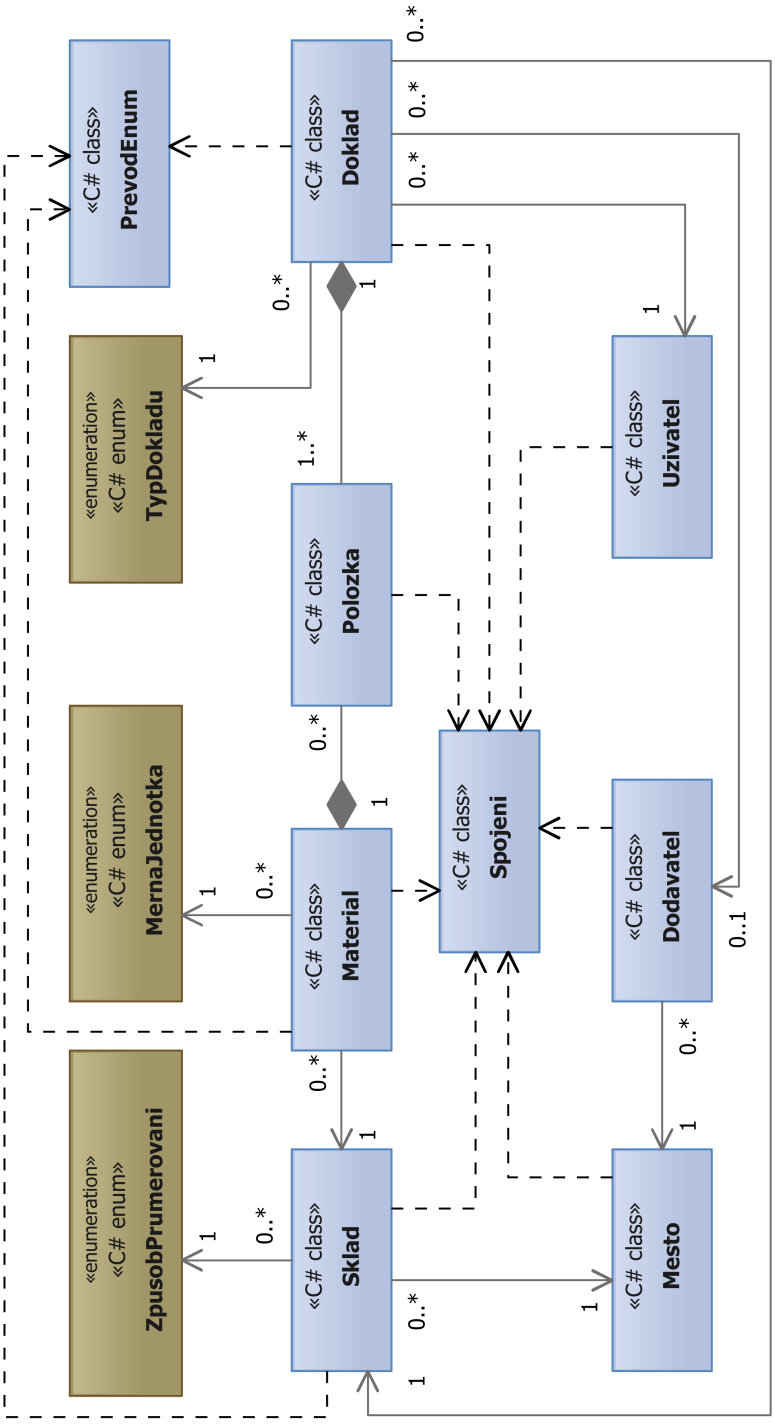
[Vir(2002)] VIRIUS, Miroslav. *C# pro zelenáče*. Praha: Neocortex, 2002, s. 27-28. ISBN 80-86330-11-7.

[You(2012)] The History of SQL Server. In: *Youtube* [online video]. 2012 [cit. 2014-04-16]. Dostupné z: `https://www.youtube.com/watch?v=fSN2ihUkSCk`

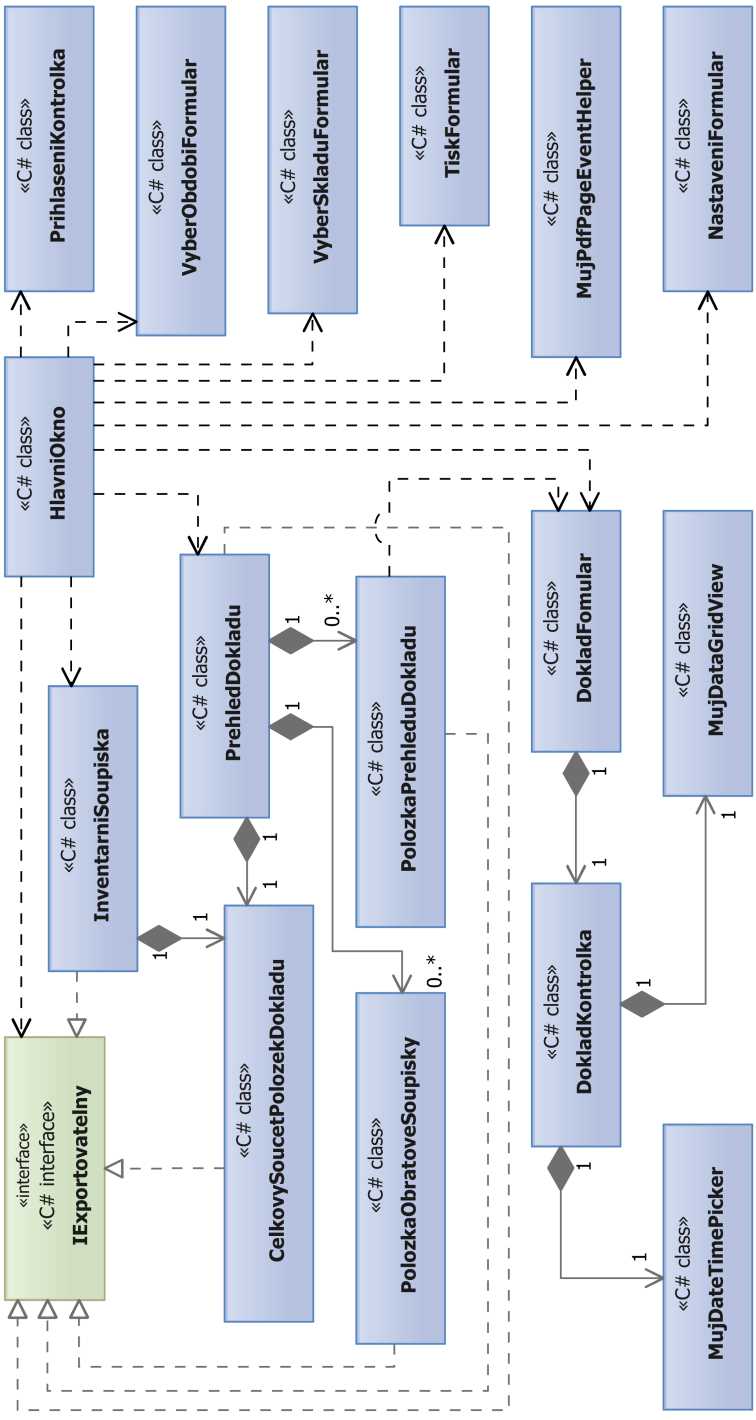
[Zás(2014)] Slovníček účetních pojmů: Zásoby. In: *Testy z účetnictví* [online]. 2014 [cit. 2014-04-14]. Dostupné z: `http://www.testyzucetnictvi.cz/slovnicek-ucetnich-pojmu.php?pojem=zasoby`

Seznam příloh

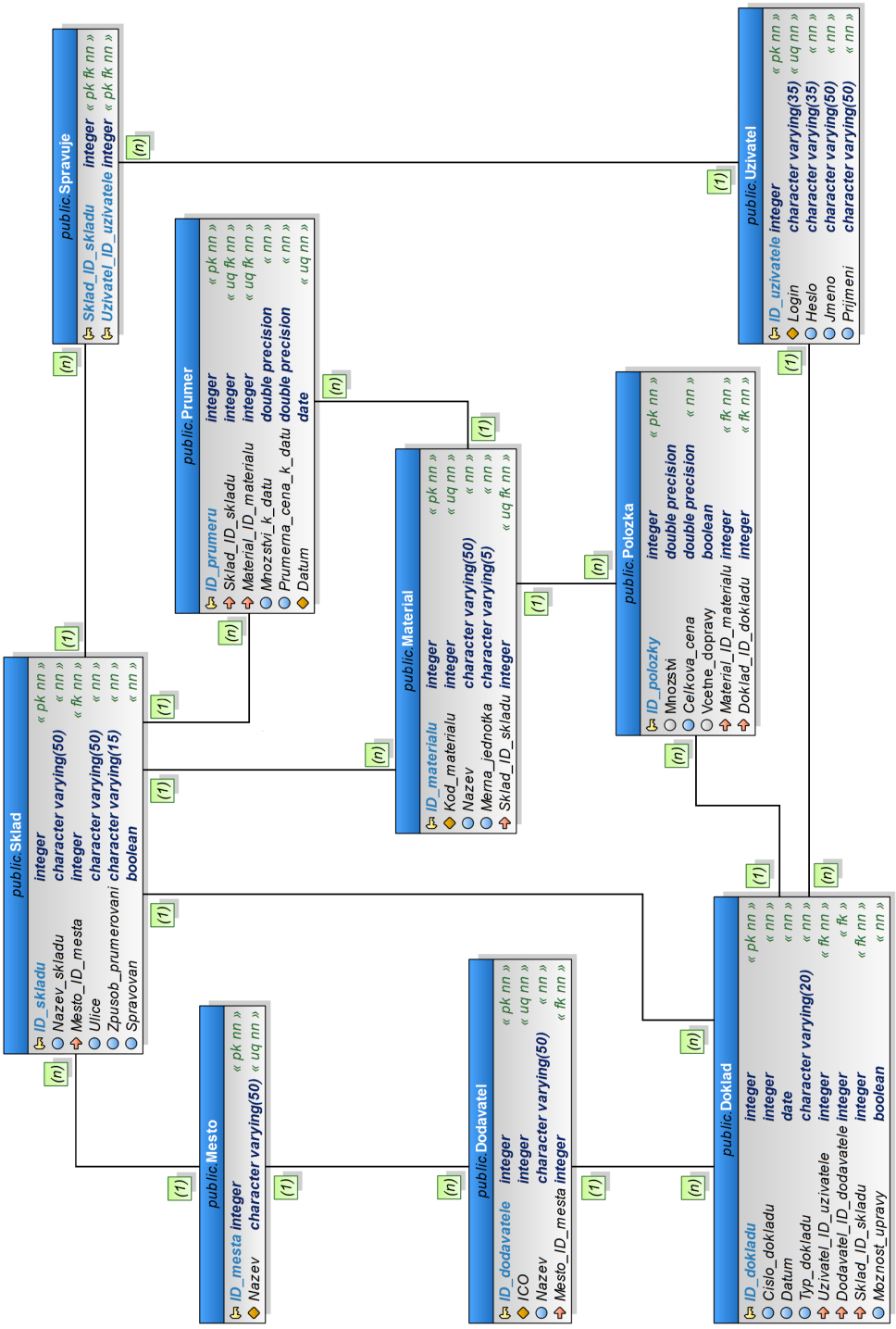
A UML: Diagram tříd – Jadro



B UML: Diagram tříd – GUI



C Relační model databáze



D Ukázka testovacích dat

D.1 Únor

Číslo dokladu: 1220003 Typ: Příjem Dodavatel: 1 – Silmat, Příbram

Kód	Název	MJ	Množství	Jednotková cena	Doprava	Cena celkem
104	asfalt	t	32,065	11 185,97	Ano	358 678,20
199	písek	t	213,710	140,00	Ano	29 919,42
540	nafta motorová	l	1 026,000	28,77	Ano	29 514,75
Celkem						418 112,37 Kč

Číslo dokladu: 1220004 Typ: Příjem Dodavatel: –

Kód	Název	MJ	Množství	Jednotková cena	Doprava	Cena celkem
193	kamenivo 2/5	t	815,230	277,20	Ano	225 982,78
Celkem						225 982,78 Kč

D.2 Březen

Číslo dokladu: 1220034 Typ: Příjem Dodavatel: 2 – BGH, České Budějovice

Kód	Název	MJ	Množství	Jednotková cena	Doprava	Cena celkem
104	asfalt	t	54,010	11 625,00	Ano	627 866,25
Celkem						627 866,25 Kč

Číslo dokladu: 1220049 Typ: Příjem Dodavatel: 3 – Bögl a Krýsl, Dobřany

Kód	Název	MJ	Množství	Jednotková cena	Doprava	Cena celkem
193	kamenivo 2/5	t	68,270	187,92	Ano	12 829,12
Celkem						12 829,12 Kč

Číslo dokladu: 1220052 Typ: Příjem Dodavatel: 3 – Bögl a Krýsl, Dobřany

Kód	Název	MJ	Množství	Jednotková cena	Doprava	Cena celkem
193	kamenivo 2/5	t	34,930	188,00	Ne	6 566,84
Celkem						6 566,84 Kč

Číslo dokladu: 1220060 Typ: Příjem dopravy Dodavatel: –

Kód	Název	MJ	Množství	Jednotková cena	Doprava	Cena celkem
193	kamenivo 2/5	t	–	–	–	7 737,75
Celkem						7 737,75 Kč

Číslo dokladu: 1220006 Typ: Příjem Dodavatel: 4 – Berdych, Plzeň

Kód	Název	MJ	Množství	Jednotková cena	Doprava	Cena celkem
199	písek	t	271,560	139,00	Ano	37 746,84
Celkem						37 746,84 Kč

Číslo dokladu: 1 Typ: Výdej do spotřeby

Kód	Název	MJ	Množství	Jednotková cena	Doprava	Cena celkem
104	asfalt	t	39,940	–	–	457 770,39
193	kamenivo 2/5	t	76,730	–	–	21 147,25
Celkem						478 917,64 Kč

Číslo dokladu: 2 Typ: Výdej do spotřeby

Kód	Název	MJ	Množství	Jednotková cena	Doprava	Cena celkem
199	písek	t	114,070	–	–	15 905,92
540	nafta motorová	l	858,000	–	–	24 684,66
Celkem						40 590,58 Kč

E Uživatelská dokumentace

E.1 Překlad a spuštění aplikace

Pro správné přeložení a bezproblémový běh aplikace je **nutné** mít nainstalovaný **Microsoft .NET Framework 4.5**. Pro překlad aplikace slouží skript `build.csproj` v adresáři `src` a samotný překlad se provádí programem `MSBuild.exe`. Tento program pro překlad je součástí .NET Frameworku a to jak v 32 bitové, tak v 64 bitové verzi. Výchozí cesta k program `MSBuild.exe` je:

- pro 32 bitovou verzi – `C:\Windows\Microsoft.NET\Framework\v4.0.30319`
- pro 64 bitovou verzi – `C:\Windows\Microsoft.NET\Framework64\v4.0.30319`

Tuto cestu je **nutné** mít přidanou v systémové proměnné **PATH** pro jednodušší překlad programu a verze tohoto programu se volí podle verze systému (64 bitový systém = 64 bitová verze apod.). Samotný překlad programu se pak provádí pomocí příkazu:

```
msbuild build.csproj
```

Pokud překlad proběhl úspěšně, výsledná spustitelná verze aplikace se nachází v adresáři o úroveň výše a to konkrétně v adresáři `bin`. Samotná aplikace se pak spouští souborem `Sklad.exe`.

E.2 Instalace databázového systému

E.2.1 Linux – Debian

Instalace databázového systému pro operační systém Linux v distribuci Debian je možná buď pomocí balíčkovacího systému **apt** nebo pomocí **grafického instalátoru** přiloženého ve složce `PostgreSQL\Linux`. V případě použití balíčkovacího systému postupujte podle návodu na adrese `https:`

`//wiki.postgresql.org/wiki/Apt`. Následně je ještě vhodné pro snazší manipulaci s databázovým systémem nainstalovat program **PgAdmin3**. Instalace tohoto programu se provádí příkazem:

```
sudo apt-get install pgadmin3
```

V případě použití grafického instalátoru je nutné vybrat instalační soubor z již výše zmíněné složky vzhledem k 32 bitové nebo 64 bitové verzi operačního systému. Instalace se pak zahájí příkazem

```
sudo ./postgresql-9.3.4-3-linux-<verze>.run
```

kde `<verze>` je typ operačního systému. Tedy `x64` pro 64 bitový operační systém a `x86` pro 32 bitový operační systém. Dále se postupuje podle pokynů instalátoru. Následně je opět vhodné doinstalovat program **PgAdmin3**.

E.2.2 Windows

Databázový systém PostgreSQL pro operační systém Windows lze nainstalovat pouze pomocí **grafického instalátoru** přiloženého ve složce `PostgreSQL\Windows`. Pro instalaci stačí spustit soubor

```
postgresql-9.3.4-3-<verze>.exe
```

kde `<verze>` je typ operačního systému. Tedy `x64` pro 64 bitový operační systém a `x86` pro 32 bitový operační systém. Dále se postupuje podle pokynů instalátoru. Následně je vhodné doinstalovat program **PgAdmin3** pro snazší manipulaci s databázovým systémem. Tento program je možné nainstalovat rovnou jako součást instalace databázového systému nebo je přiložen jeho instalátor ve složce `PostgreSQL\Windows\PgAdmin3`. Kde stačí spustit soubor

```
pgadmin3.msi
```

E.2.3 Vytvoření databáze pomocí psql

Jedním ze způsobů vytvoření databáze je použití programu **psql**. Ke snadnému vytvoření databáze slouží SQL skript `databaze.sql` přiložený ve složce `Databaze`. Vytvoření databáze prostřednictvím **psql** se na Linuxu a ve Win-

dows drobně liší. V Linuxu stačí v terminálu použít příkaz

```
psql -h <host> -p <port> -U <login> -W -f <skript>
```

kde `<host>` je hostname resp. IP adresa serveru, v našem případě *localhost*, `<port>` je port, na kterém databázový server běží (implicitní port je *5432*), `<login>` je uživatelské jméno pro přístup k serveru, defaultně *postgres* a část `<skript>` reprezentuje SQL skript, tedy *databaze.sql*. Pak už se jen stačí řídit případnými pokyny.

Po vytvoření databáze je ještě nutné vytvořit uživatelské funkce. Pro vytvoření uživatelských funkcí je určen skript s názvem `funkce.sql`, který je přiložen ve stejné složce jako `databaze.sql`. Pro vytvoření uživatelských funkcí slouží příkaz

```
psql -h <host> -p <port> -U <login> -W -d <nazev_databaze> -f
<skript>
```

kde jediným rozdílem oproti předchozímu příkazu je `-d <nazev_databaze>`. Tato část udává název databáze, v našem případě *sklad*. Dále je pak nutné v části `<skript>` místo *databaze.sql* použít skript *funkce.sql*. V případě použití testovacích dat ze souboru `testovaci_data.sql`, bude použit stejný příkaz, pouze bude tento skript použit v části `<skript>`.

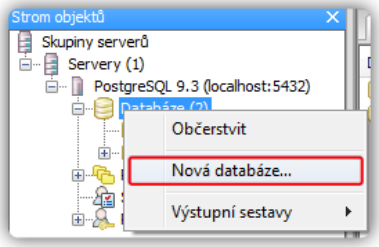
Rozdíl oproti Linuxu je ten, že ve Windows je nutné nejprve najít složku s nainstalovaným **PostgreSQL** (pokud není nastavená systémová proměnná **PATH**), kde ve složce `bin` je program `psql.exe`. Příkaz pro vytvoření databáze je téměř totožný s Linuxovým příkazem pouze místo `psql` je použito `psql.exe`.

Příklad všech tří výše uvedených příkladů pro běžné použití vypadá následně:

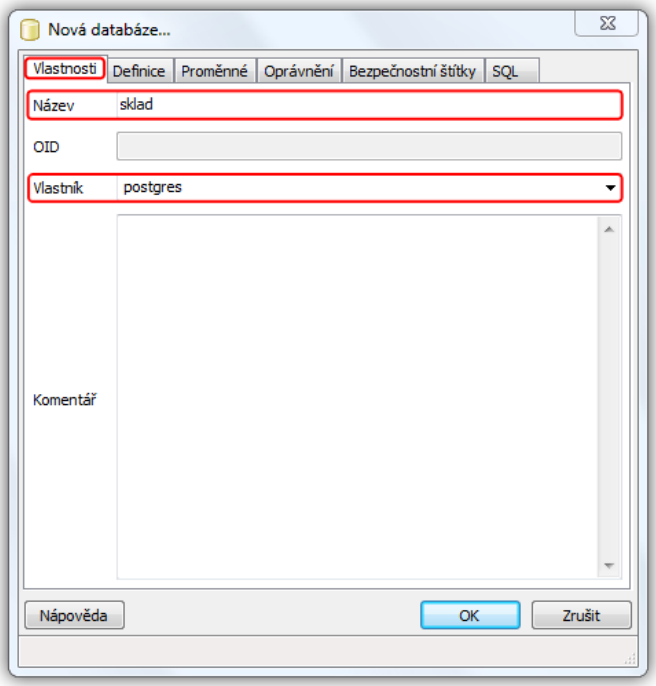
```
psql -h localhost -p 5432 -U postgres -W -f databaze.sql

psql -h localhost -p 5432 -U postgres -W -d sklad -f funkce.sql

psql -h localhost -p 5432 -U postgres -W -d sklad -f testovaci-
_data.sql
```

E.2.4 Vytvoření databáze pomocí PgAdmin3

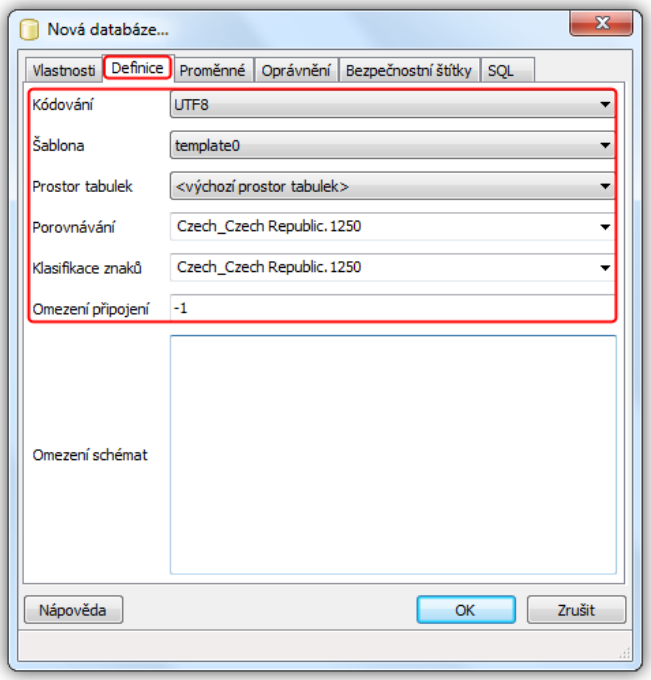
Dalším ze způsobu vytvoření databáze je použití programu **PgAdmin3**. V tomto programu nejprve musíme vytvořit databázi. Ve stromu objektů klikneme pravým tlačítkem myši na položku **Databáze**, jak je vidět na obrázku E.1. Následně do pole **Název databáze** napíšeme `Sklad` a položku **Vlastník** změníme na `postgres`. Výsledek je vidět na obrázku E.2. Následně v záložce **Definice** vyplníme všechna pole jako na obrázku E.3 a potvrdíme.



Obrázek E.1: Vytvoření databáze



Obrázek E.2: Vyplnění záložky **Vlastnosti**



Obrázek E.3: Vyplnění záložky **Definice**

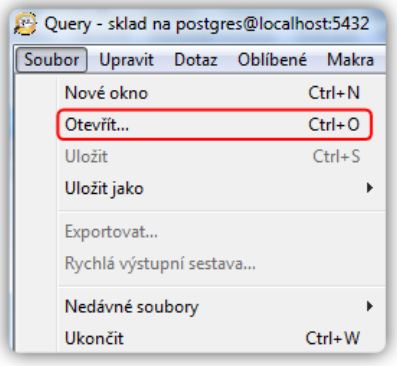
V případě, že nechcete provádět toto nastavení, je možné v okně vytváření databáze do záložky **SQL** po odškrtnutí položky **Jen pro čtení** zkopírovat následující kód a potvrdit.

```
CREATE DATABASE sklad
  WITH ENCODING='UTF8'
       OWNER=postgres
       TEMPLATE=template0
       LC_COLLATE='Czech_Czech Republic.1250'
       LC_CTYPE='Czech_Czech Republic.1250'
       CONNECTION LIMIT=-1
       TABLESPACE=pg_default;
```

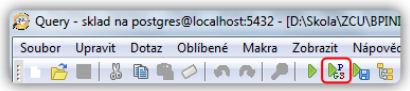
Nyní je ještě nutné vytvořit samotnou strukturu databáze. To provedeme kliknutím na tlačítko SQL zobrazeném na obrázku E.4.

Obrázek E.4: Tlačítko **SQL**

Po zobrazení okna konzole klikneme do hlavního menu na položku **Soubor** a zde vybereme **Otevřít**. Tento postup lze vidět na obrázku E.5. Následně vybereme SQL skript `databaze_pgadmin.sql`, který je přiložený ve složce `Databaze`. Pak už jen stačí kliknout na tlačítko **Spustit skript** vyobrazené na obrázku E.6.



Obrázek E.5: Otevření skriptu



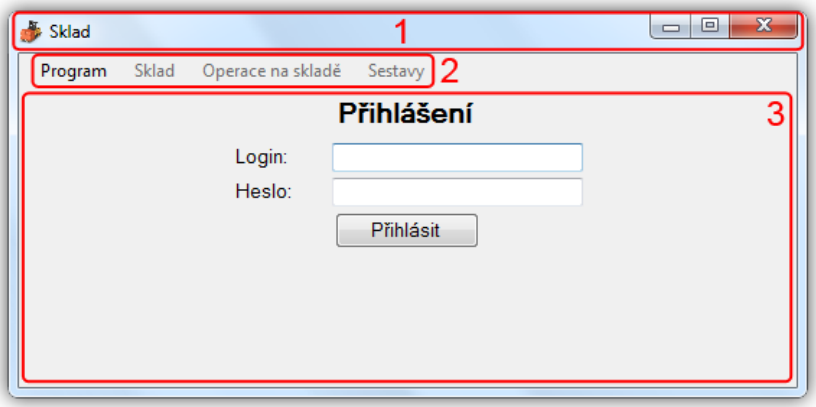
Obrázek E.6: Spuštění skriptu

Obdobným způsobem je pak ještě nutné spustit skript `funkce.sql` a v případě použití testovacích dat také `testovaci_data.sql`.

E.3 Popis hlavního okna programu

Hlavní okno (obrázek E.7) se skládá ze 3 částí, konkrétně ze **záhlaví okna** (1), **menu** (2) a **hlavního panelu** (3). Záhlaví okna obsahuje především titulek s nadpisem, který se mění podle přihlášeného uživatele a vybraného skladu a také tlačítka pro minimalizaci, maximalizaci a zavření okna. Menu

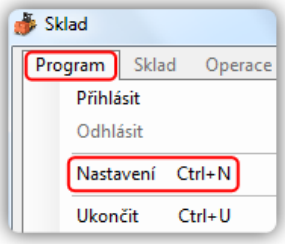
programu reprezentuje hlavní ovládací prvek programu. Prostřednictvím tohoto menu se ovládá celý program. Hlavní panel programu slouží především pro zobrazování sestav a také pro přihlášení.



Obrázek E.7: Hlavní okno

E.4 Nastavení a kontrola spojení

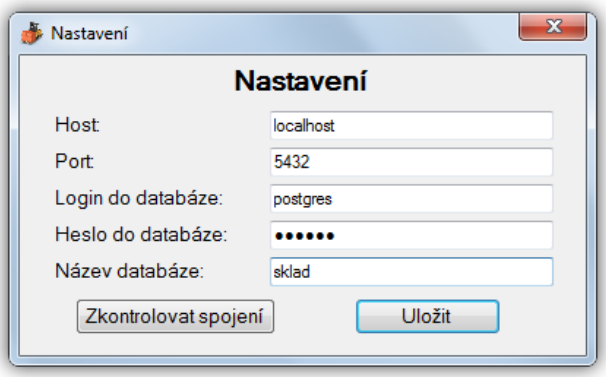
Nastavení spojení se provede kliknutím na položku menu **Program** a následně se vybere položka **Nastavení** (obrázek E.8). Případně je možné použít klávesovou zkratku *Ctrl + N*.



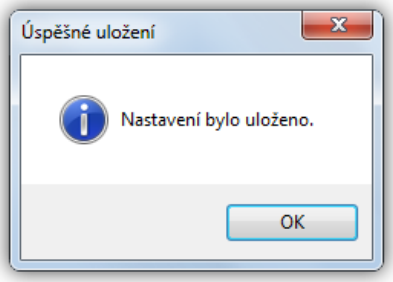
Obrázek E.8: Položka menu **Nastavení**

V okně **Nastavení** pak vyplníme všechny položky jako na obrázku E.9, pouze změníme položku **Host** na hostname databázového serveru. Před uložením nastavení můžeme zkontrolovat správné vyplnění kliknutím na tlačítko **Zkontrolovat spojení**. Příslušné dialogové okno ohlásí, zda-li byla kontrola úspěšná či neúspěšná. Pro uložení klikneme na tlačítko **Uložit**, zobrazí se

dialogové okno (obrázek E.10), které potvrdíme a zavřeme okno **Nastavení** křížkem v pravém horním rohu.



Obrázek E.9: Vyplněné okno **Nastavení**



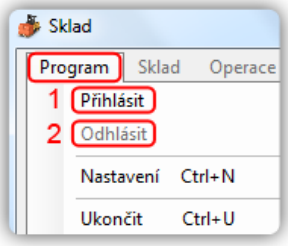
Obrázek E.10: Úspěšné uložení nastavení

E.5 Přihlášení a odhlášení uživatele

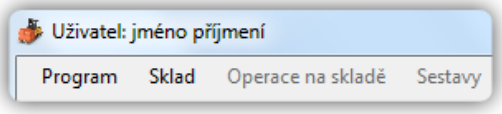
Přihlášení uživatele se provádí kliknutím na položku menu **Program** a následně se vybere položka **Přihlásit** (1). Tato akce je znázorněna na obrázku E.11. Po této akci se v hlavním panelu programu zobrazí formulář pro přihlášení, kde je nutné vyplnit položky **Login** a **Heslo** a následně kliknout na tlačítko **Přihlásit**. Úspěšné přihlášení se projeví změnou titulku v záhlaví hlavního okna programu na jméno a příjmení přihlášeného uživatele (obrázek E.12). Pro přihlášení uživatele pod testovacími daty se do pole **Login** napíše „`login`“ a do pole **Heslo** napíše „`heslo`“ stejně jako na obrázku E.13.

Odhlášení uživatele se provede kliknutím na položku menu **Program**

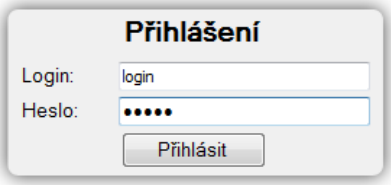
a následně se vybere položka **Odhlásit** (2). Tuto činnost můžeme vidět na obrázku E.11.



Obrázek E.11: Položky menu **Přihlásit** a **Odhlásit**



Obrázek E.12: Změna záhlaví po přihlášení

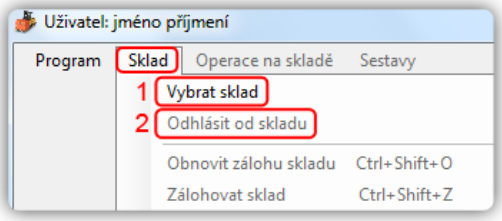


Obrázek E.13: Ukázka přihlášení

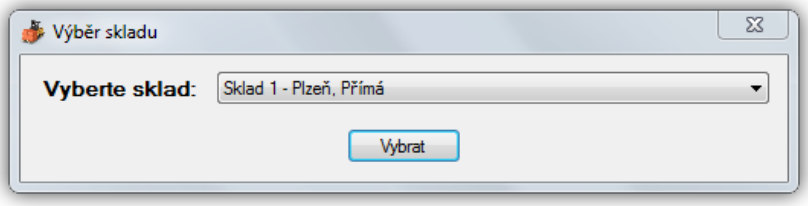
E.6 Výběr a odhlášení od skladu

Výběr skladu se provede kliknutím na položku menu **Sklad** a následně výběrem položky **Vybrat sklad** (1). Tato činnost je znázorněna na obrázku E.14. Po této akci se zobrazí dialogové okno pro výběr skladu (obrázek E.15), kde z nabídky skladů vybereme sklad, který chceme spravovat a klikneme na tlačítko **Vybrat**. Úspěšné vybrání skladu se projeví změnou záhlaví hlavního okna a to konkrétně přidáním informací o vybraném skladu (obrázek E.16).

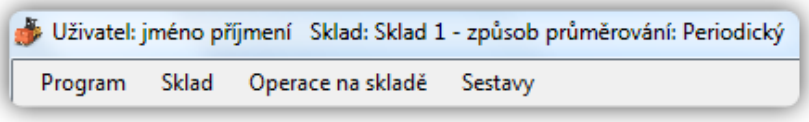
Odhlášení od skladu se provede kliknutím na položku menu **Sklad** a následně výběrem položky **Odhlásit od skladu** (2). Tato činnost je znázorněna na obrázku E.14.



Obrázek E.14: Položky menu **Vybrat sklad** a **Odhlásit od skladu**



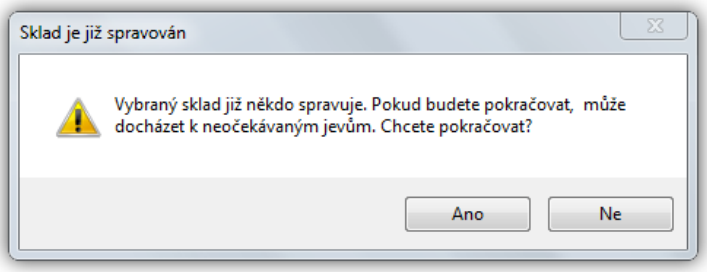
Obrázek E.15: Dialogové okno pro výběr skladu



Obrázek E.16: Změna záhlaví po výběru skladu

Pokud se při výběru skladu zobrazí dialogové okno jako na obrázku E.17 znamená to, že vybraný sklad aktuálně někdo spravuje a pokud budete pokračovat, může docházet k neočekávaným jevům. V tomto případě je **vhodné nepokračovat** a počkat než bude daný sklad volný, případně zjistit kdo daný sklad aktuálně spravuje.

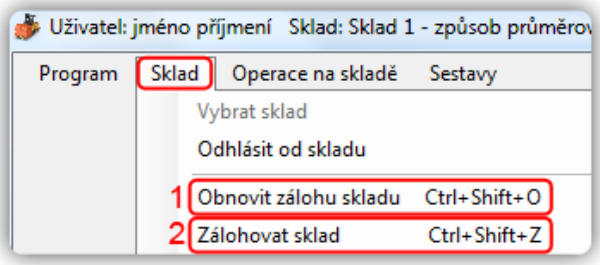
Pozn.: Může se stát, že při předchozím užívání programu nastala neočekávaná událost např. pád databázového serveru a v tomto důsledku nedošlo k odhlášení uživatele od skladu. Program pak mylně signalizuje, že sklad je využíván někým jiným. V tomto případě je vhodné kontaktovat administrátora serveru, aby chybu opravil.



Obrázek E.17: Upozornění spravovaného skladu

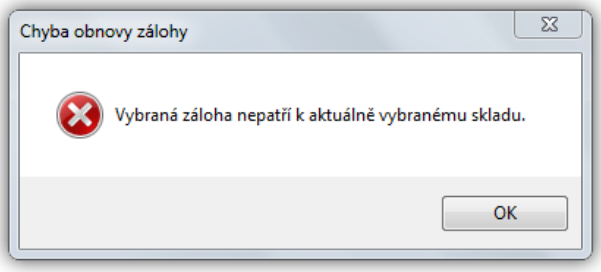
E.7 Zálohování a obnovení zálohy skladu

Zálohování skladu se prování kliknutím na položku menu **Sklad** a následně na **Zálohovat sklad** (2). Tato činnost je vidět na obrázku E.18. Případně je možné použít klávesovou zkratku *Ctrl + Shift + Z*. Následně se zobrazí dialogové okno, kde vybereme umístění zálohy, název zálohy a potvrdíme. Úspěšné či neúspěšné provedení zálohy ohlásí dialogové okno.



Obrázek E.18: Položky menu **Zálohovat sklad** a **Obnovit zálohu skladu**

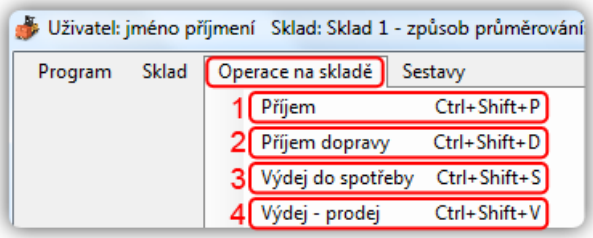
Obnovení zálohy skladu se provádí kliknutí na položku menu **Sklad** a následně na **Obnovit zálohu skladu** (1). Tato činnost je znázorněna na obrázku E.18. Případně je možné použít klávesovou zkratku *Ctrl + Shift + O*. Následně se zobrazí dialogové okno, kde vybereme zálohu a potvrdíme. V případě vybrání zálohy, která nepatří k aktuálně vybranému skladu se zobrazí dialogové okno jako na obrázku E.19. Úspěšné či neúspěšné obnovení zálohy se ohlásí příslušným dialogovým oknem. V případě **neúspěšného** obnovení zálohy, zůstanou data skladu **stejná** jako před započetím této akce.



Obrázek E.19: Špatně vybraná záloha

E.8 Operace na skladě

Operace na skladě se provádí kliknutím na položku menu **Operace na skladě** a následným výběrem jedné z položek podle požadované operace. Tyto položky jsou vidět na obrázku E.20, kde jsou vidět i příslušné klávesové zkratky.

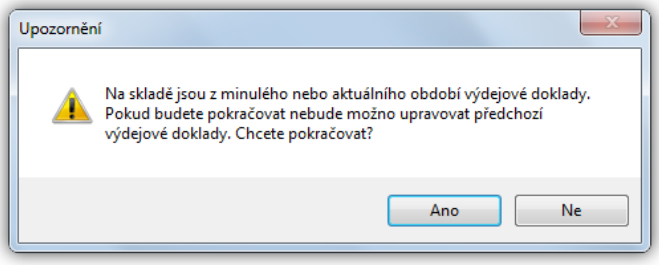


Obrázek E.20: Položky menu v části **Operace na skladě**

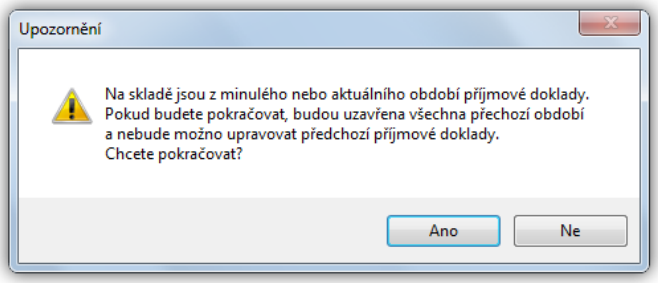
Při výběru položky (1) nebo (2) se zobrazí upozornění znázorněné na obrázku E.21, které říká, že výdajové doklady přijaté v předchozím období budou uzavřeny. Dříve než budete pokračovat, je **doporučeno** zkontrolovat výdajové doklady z minulého období, případně provést zálohu skladu, protože po potvrzení tohoto upozornění již nebude možné tyto doklady upravovat ani mazat.

Při výběru položky (3) nebo (4) se zobrazí upozornění znázorněné na obrázku E.22, které říká, že příjmové doklady přijaté v aktuálním období budou uzavřeny a budou vypočítány periodické průměry za aktuální období. Dříve než budete pokračovat, je **doporučeno** zkontrolovat příjmové doklady z aktuálního období, případně provést zálohu skladu, protože po potvrzení tohoto upozornění již nebude možné tyto doklady upravovat, mazat ani přidá-

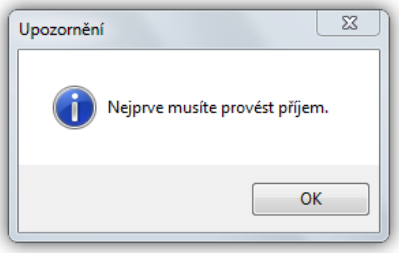
vat další v aktuální období. Při výběru těchto položek se také může zobrazit informační dialog (obrázek E.23), který informuje o tom, že na skladě není přijatý žádný materiál a je nutné nejprve nějaký materiál přijmout.



Obrázek E.21: Upozornění uzavření období při příjmu



Obrázek E.22: Upozornění uzavření období při výdeji



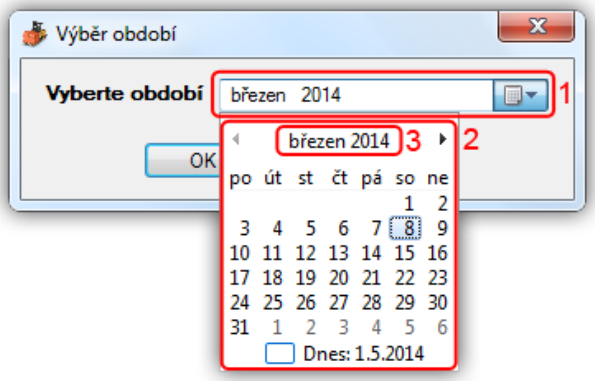
Obrázek E.23: Upozornění na prázdný sklad

Po těchto akcích vždy následuje výběr období a po něm už samotný formulář pro danou skladovou operaci. Zde je nutné ještě podotknout, že formuláře pro operace na skladě je možno posouvat na další položku klávesou *Enter*, pro urychlení práce.

E.8.1 Výběr období

Dialogové okno pro výběr období je vidět na obrázku E.24. Samotný výběr období se provádí kliknutím do pole pro výběr data (1), po kterém se zobrazí kalendář (2). V tomto kalendáři je vidět název měsíce společně s rokem (3) a jednotlivé dny v tomto měsíci.

Pro výběr aktuálně zvoleného měsíce stačí kliknout na jakýkoliv den v tomto měsíci. Pro změnu měsíce je nutné kliknout na prvek (3), následně se místo dnů zobrazí měsíce, kde pro požadovaný měsíc vybereme příslušnou položku a na ni klikneme. Poté opět zvolíme jakýkoliv den v tomto měsíci. Pro výběr roku postupujeme obdobně, pouze při části výběru měsíce, klikneme na prvek (2) ještě jednou. Vybrané období je pak vidět v prvku (1). Po výběru období už pouze stačí kliknout na tlačítko **Ok**.



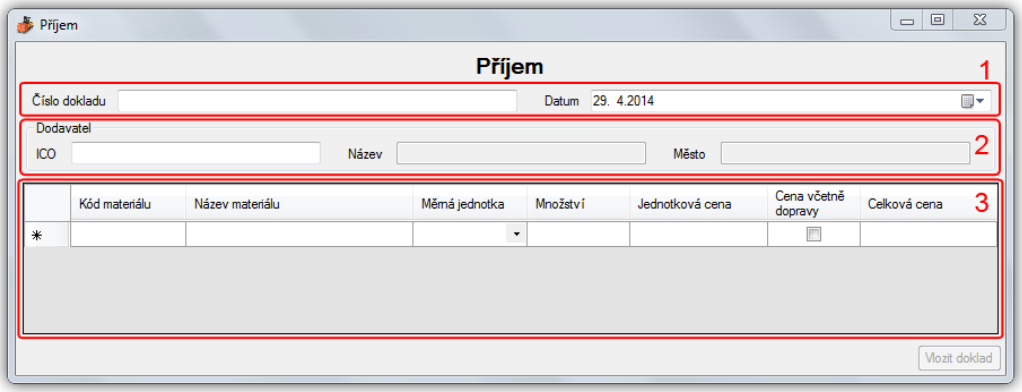
Obrázek E.24: Formulář **Výběr období**

E.8.2 Příjem a Příjem dopravy

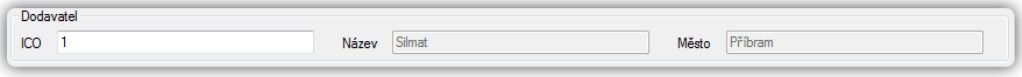
Formulář pro příjem je vidět na obrázku E.25. Při příjmu je vždy nejprve důležité vyplnit hlavičku dokladu (1) a to zejména položku **Číslo dokladu** a následně položku **Datum**. Datum je možné buď napsat nebo vybrat obdobným způsobem jako při výběru období, ale je možné nastavit pouze datum z aktuálně vybraného období.

Část **Dodavatel** (2) se řídí podle položky **ICO**. Pokud je tato položka prázdná, považuje se daný doklad jako doklad bez dodavatele. Pokud je zadáno identifikační číslo dodavatele, který již je v databázi, je doplněn auto-

maticky (obrázek E.26). V opačném případě je nutné vyplnit kromě položky **ICO** i položky **Název** a **Město**. Oba formuláře jak **Příjem**, tak **Příjem dopravy** mají prvek (1) a (2) stejný a liší se až v prvku (3).



Obrázek E.25: Formulář pro příjmy

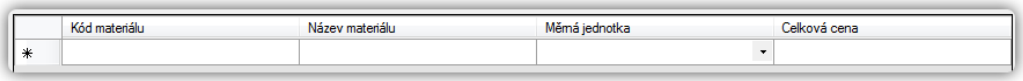


Obrázek E.26: Ukázka doplnění dodavatele

Prvek (3) reprezentuje položky samotného dokladu. Pro správné vyplnění dokladu je nutné u každé položky vždy vyplnit všechny buňky. Při zadávání kódu materiálu se v případě jeho existence automaticky doplní buňky **Název materiálu** a **Měrná jednotka** a tato dvě pole následně nejdou editovat. V opačném případě je nutné tato dvě pole vyplnit. Oba dva typy dokladů mají tuto část prvku (3) stejnou.

U formuláře **Příjem** jsou dále buňky **Množství**, **Jednotková cena**, **Cena včetně dopravy** a **Celková cena**. Po zadání množství a celkové ceny se jednotková cena vypočítává automaticky a pro úplné vyplnění položky dokladu už pouze stačí zatrhnout, zda-li je daná cena včetně dopravy nebo ne (buňka **Cena včetně dopravy**).

U formuláře **Příjem dopravy** je dále pouze buňka **Celková cena** a jak vypadá prvek (3) pro tento formulář lze vidět na obrázku E.27. Pro možnost vyplnění této buňky je nutné nejprve vyplnit buňku **Kód materiálu** existujícím materiálem.



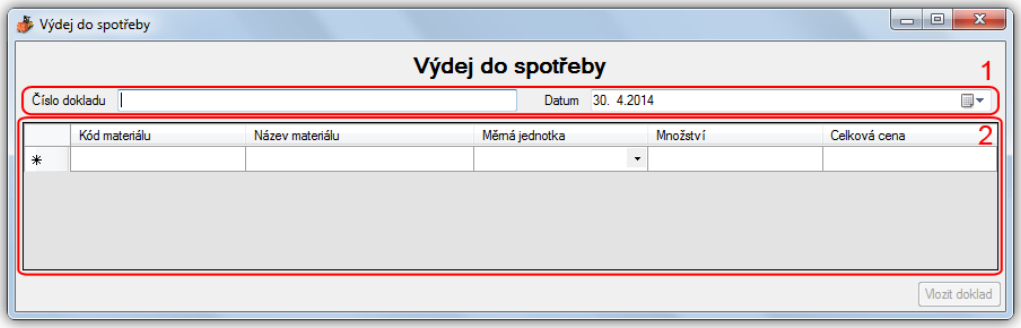
Obrázek E.27: Prvek (3) pro formulář **Příjem dopravy**

Po vyplnění všech potřebných položek už pouze stačí kliknout na tlačítko **Vložit**. Případné chybové hlášky jsou vysvětleny v sekci E.8.4.

E.8.3 Výdej do spotřeby a Výdej – prodej

Obě dvě skladové operace mají podobný formulář a tento formulář je znázorněn na obrázku E.28. Stejně jako u příjmů, je vždy nejprve důležité vyplnit hlavičku dokladu (1). U těchto skladových operací není část dodavatele a proto stačí vyplnit položku **Číslo dokladu** a také **Datum**. Pro položku **Datum** platí stejná pravidla jako u příjmů.

V části (2), která reprezentuje položky dokladu, je nejprve nutné zadat kód existujícího materiálu, následně se automaticky doplní buňky **Název materiálu**, **Měrná jednotka** a zpřístupní se buňka **Množství**. Do této buňky stačí zadat množství materiálu a celková cena se doplní programově na základě výpočtu váženého aritmetického průměru. Po vyplnění všech položek dokladu už stačí pouze kliknout na tlačítko **Vložit doklad**. Případné chybové hlášky jsou vysvětleny v sekci E.8.4.



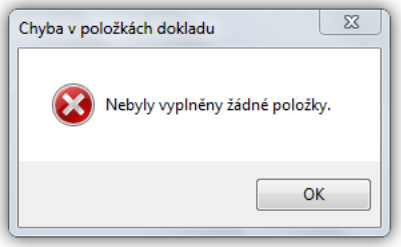
Obrázek E.28: Formulář pro výdeje

E.8.4 Chybové hlášky

- *Chyba v hlavičce dokladu* – Tato chyba se projevuje vždy v hlavičkách dokladů a je signalizována vykřičníkem vedle položky, kde chyba vznikla. Po najetí myší na tento vykřičník bude daná chyba upřesněna. Příklad této chyby je vidět na obrázku E.29.

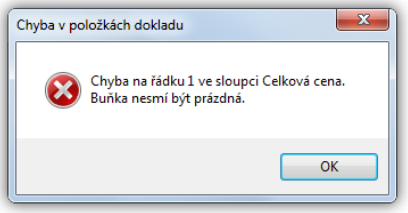
Obrázek E.29: Chyba v hlavičce dokladu

- *Nevyplněné položky* – Tato chyba vzniká v případě vkládání dokladu, u kterého nebyla vyplněna ani jedna položka. Chybová hláška je znázorněna na obrázku E.30.



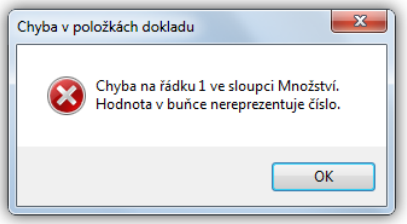
Obrázek E.30: Nevyplněné položky

- *Prázdná buňka* – Tato chyba může vzniknout téměř ve všech buňkách všech formulářů a vzniká v případě nevyplnění některé z buněk položky dokladu. Jak vypadá chybová hláška této chyby je vidět na obrázku E.31.



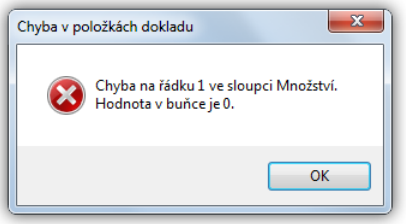
Obrázek E.31: Prázdná buňka

- *Chyba čísla* – Tato chyba vzniká především v buňkách **Celková cena** nebo **Množství** a je způsobena zadáním čísla ve špatném formátu. Chybová hláška vypadá jako na obrázku E.32.



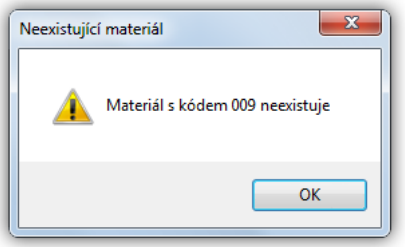
Obrázek E.32: Chyba čísla

- *Chyba nula* – Tato chyba vzniká především v buňkách **Celková cena** nebo **Množství** a je způsobena zadáním čísla nula. Chybová hláška vypadá jako na obrázku E.33.



Obrázek E.33: Chyba nula

- *Neexistující materiál* – Tato chyba vzniká ve formulářích **Příjem dopravy**, **Výdej do spotřeby** a **Výdej – prodej**. Je způsobena zadáním kódu materiálu, který není evidován. Chybová hláška vypadá jako na obrázku E.34.

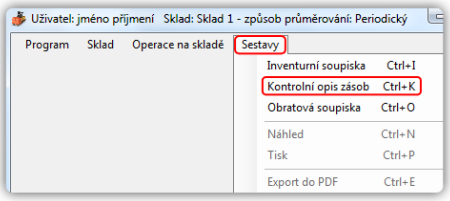


Obrázek E.34: Neexistující materiál

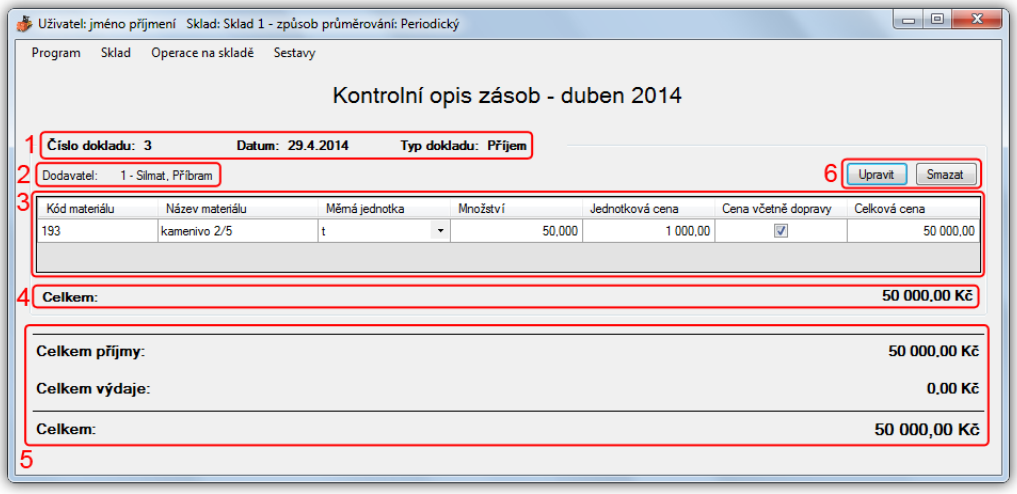
U chyb *Prázdná buňka*, *Chyba nula* a *Chyba čísla* je možné identifikovat buňku chyby podle chybové hlášky, kde je vždy uvedeno číslo řádku a název sloupce.

E.9 Kontrolní opis zásob

Kontrolní opis zásob je možné zobrazit kliknutím na položku menu **Sestavy** a následně na položku **Kontrolní opis zásob**, jak je znázorněno na obrázku E.35. Případně je možné použít klávesovou zkratku *Ctrl + K*. Po provedení této činnosti je nutné provést výběr období obdobně, jak je popsáno v sekci E.8.1. V případě, že ve vybraném období nejsou žádné doklady, zobrazí se dialogové okno s příslušnou hláškou. V opačném případě se zobrazí v hlavním panelu hlavního okna kontrolní opis zásob, podobně jako na obrázku E.36.



Obrázek E.35: Položka menu **Kontrolní opis zásob**



Obrázek E.36: Formulář **Kontrolní opis zásob**

Položky formuláře kontrolního opisu zásob:

1. informace o dokladu
2. informace o dodavateli
3. jednotlivé položky dokladu
4. celkový součet dokladu
5. celkový součet všech dokladů v kontrolním opise zásob za období
6. tlačítka pro možnost úpravy nebo smazání dokladu

Tlačítka pro možnost úpravy nebo smazání je možné použít pouze v případě, že dané období ještě nebylo uzavřeno.

E.9.1 Úprava a smazání dokladu

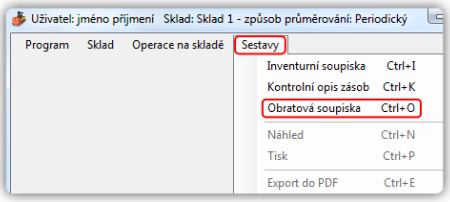
Z kontrolního opisu zásob je možno pomocí tlačítek **Upravit** a **Smazat** editovat či smazat daný doklad. V případě smazání stačí kliknout na tlačítko **Smazat**, zobrazí se dialogové okno o smazání dokladu a po potvrzení bude vybraný doklad smazán.

Úpravu je možné zahájit kliknutím na tlačítko **Upravit**. Následně se zobrazí okno podle typu dokladu stejné jako pro daný typ skladové operace. Toto okno je ovšem oproti formuláři pro skladovou operaci nastaveno tak, aby bylo možno editovat pouze sloupce **Množství**, **Celková cena** nebo **Cena včetně dopravy**. V případě provedení úpravy některé položky dokladu v těchto sloupcích stačí pro potvrzení kliknout na tlačítko **Upravit** v tomto formuláři. Následná změna se projeví i v kontrolním opise zásob.

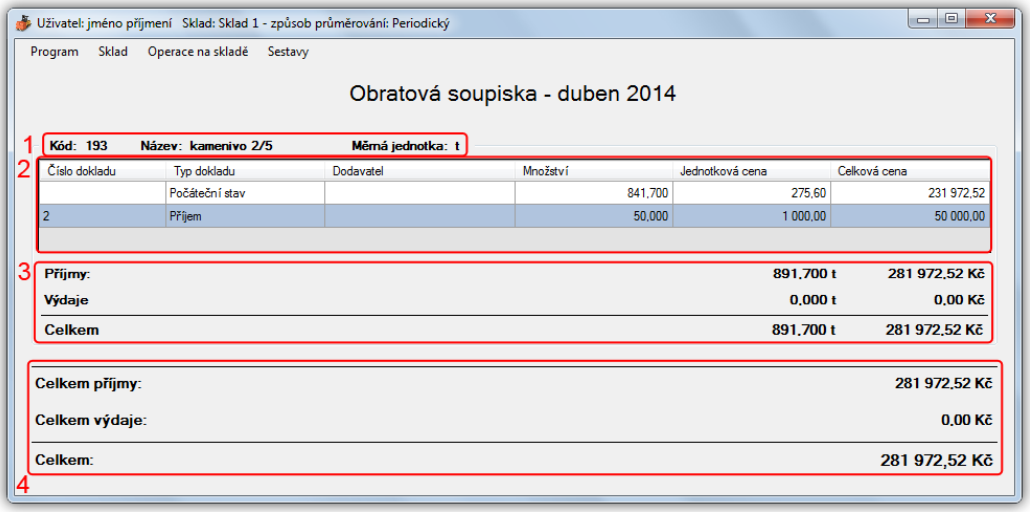
E.10 Obratová soupiska

Obratovou soupisku je možné zobrazit kliknutím na položku menu **Sestavy** a následně na položku **Obratová soupiska**, jak je možno vidět na obrázku E.37. Případně je možné použít klávesovou zkratku *Ctrl + O*. Následně je nutné provést výběr období obdobně, jak je popsáno v sekci E.8.1.

V případě, že ve vybraném období nejsou žádné pohyby materiálu, zobrazí se dialogové okno s příslušnou hláškou. V opačném případě se zobrazí v hlavním panelu hlavního okna formulář obratové soupisky, podobně jak je znázorněno na obrázku E.38.



Obrázek E.37: Položka menu **Obratová soupiska**



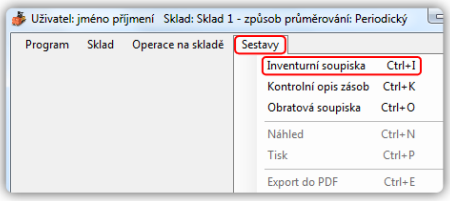
Obrázek E.38: Formulář **Obratová soupiska**

Položky formuláře opisu zásob:

1. informace o materiálu
2. pohyby materiálu v daném období
3. součty cen a množství pohybů materiálu v daném období
4. celkový součet všech pohybů materiálů v daném období

E.11 Inventurní soupiska

Inventurní soupisku je možné zobrazit kliknutím na položku menu **Sestavy** a následně na položku **Inventurní soupiska**, jak je vyobrazeno na obrázku E.39. Případně je možné použít klávesovou zkratku *Ctrl + I*. Dále je nutné provést výběr období obdobně, jak je popsáno v sekci E.8.1. V případě, že ještě nebyl naskladněn žádný materiál, zobrazí se dialogové okno s příslušnou hláškou. V opačném případě se zobrazí v hlavním panelu hlavního okna formulář inventurní soupisky, jak je možno vidět na obrázku E.40.



Obrázek E.39: Položka menu **Inventurní soupiska**

Obrázek E.40: Formulář **Inventurní soupiska**

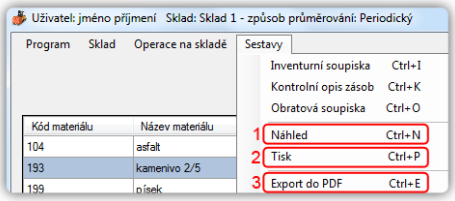
Položky formuláře opisu zásob:

1. informace o současném stavu materiálů na skladě v daném období
2. celková hodnota materiálů na skladě v daném období

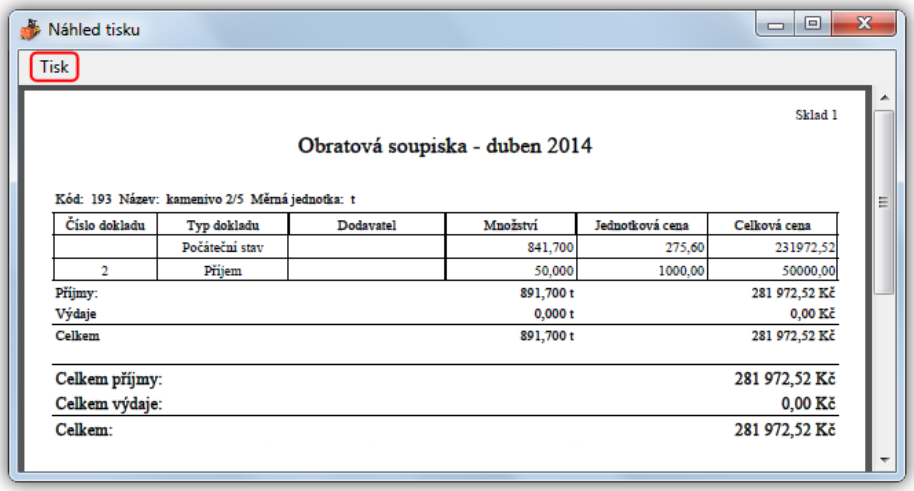
E.12 Export, tisk a náhled tisku

Aby bylo možné provést některou z těchto operací je nejprve **nutné** zobrazit **Inventurní soupisku**, **Obratovou soupisku** nebo **Kontrolní opis zásob**. Následně je možné provést jednu z těchto operací, jak je znázorněno na obrázku E.41, včetně jejich klávesových zkratek. V případě položky **Export do PDF** (3) se zobrazí dialogové okno, kde stačí vybrat umístění, napsat název výsledného souboru a potvrdit. Výsledný soubor se následně ihned zobrazí.

V případě položky **Tisk** (2) se zobrazí dialogové okno tisku, kde se provedou příslušná nastavení a potvrzením se výsledná sestava odešle k tisku na vybranou tiskárnu. Tiskový náhled je možné zobrazit kliknutím na položku **Náhled** (1). Po této akci se zobrazí okno s tiskovým náhledem. Z tohoto okna je možno daný dokument také vytisknout a to kliknutím na položku menu **Tisk**, jak je znázorněno na obrázku E.42.



Obrázek E.41: Položky menu **Náhled**, **Tisk** a **Export do PDF**

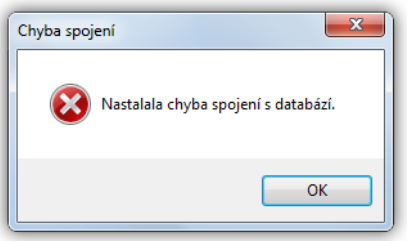


Obrázek E.42: Okno **Náhled**

Pozn.: U tiskového náhledu mohou některé čáry tabulek být opticky zkresleny. Toto zkreslení je způsobeno různým přibližování a oddalováním náhledu a přepočítáváním jednotlivých čar vůči rozlišení monitoru.

E.13 Ztráta spojení s databází

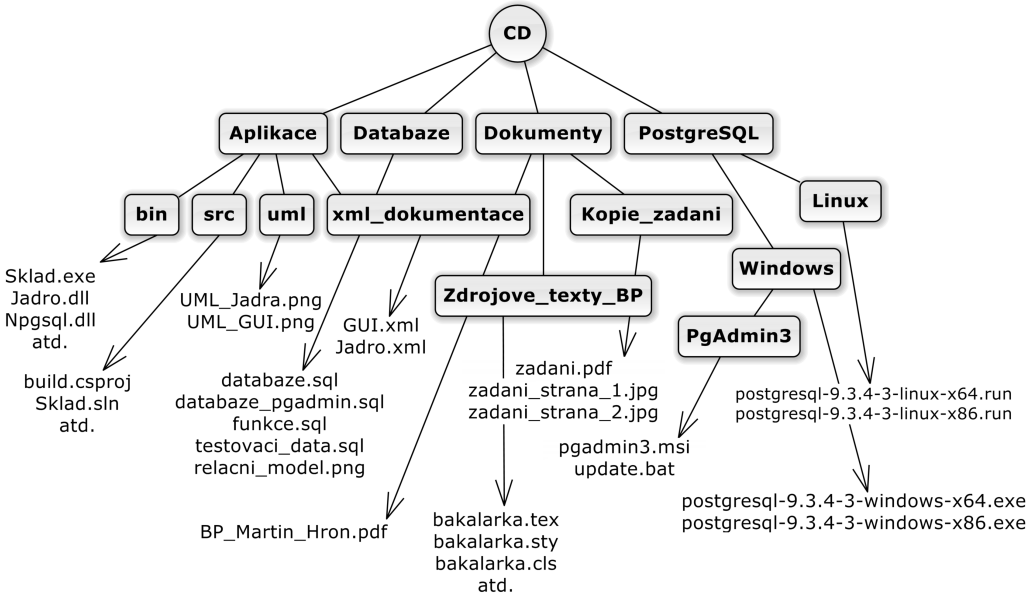
Pokud se zobrazí dialogové okno jako na obrázku E.43, znamená to, že došlo ke ztrátě spojení s databázovým systémem. V tomto případě je **možné nějakou dobu počkat a poté zkusit danou operaci znovu**. Jestliže ani poté nedošlo k opětovnému spojení s databázovým systémem, je vhodné kontaktovat administrátora serveru a oznámit tuto skutečnost.



Obrázek E.43: Ztráta spojení s databází

F Obsah CD

F.1 Struktura obsahu CD



F.2 Popis obsahu CD

Aplikace

- *bin* – složka obsahující spustitelnou verzi aplikace (`Sklad.exe` – hlavní soubor pro spuštění aplikace)
- *src* – složka obsahující zdrojové kódy aplikace (`build.csproj` – hlavní soubor pro překlad aplikace)
- *uml* – složka obsahující UML modely diagramů tříd obou částí programu
- *xml_dokumentace* – složka obsahující xml soubory s dokumentačními komentáři zdrojových kódů

Databaze

- `databaze.sql` – skript pro vytvoření databáze pomocí programu psql
- `databaze_pgadmin.sql` – skript pro vytvoření databáze pomocí programu PgAdmin3
- `funkce.sql` – skript obsahující uživatelské funkce databáze včetně komentářů
- `testovaci_data.sql` – skript s ukázkou testovacích dat
- `relacni_model.png` – obrázek relačního modelu databáze

Dokumenty

- *Kopie_zadani* – složka obsahující kopii zadání bakalářské práce
  - `zadani.pdf` – pdf dokument s kopií zadání bakalářské práce
  - `zadani_strana_1.jpg` – obrázek první strany kopie bakalářské práce
  - `zadani_strana_2.jpg` – obrázek druhé strany kopie bakalářské práce
- *Zdrojovy_text_BP*
  - *obrazky* – složka s obrázky bakalářské práce
  - *modely* – složka s modely bakalářské práce
  - *uzivatelska_dokumentace* – složka s obrázky uživatelské dokumentace
  - `bakalarka.tex` – hlavní zdrojový soubor bakalářské práce
- `BP_Martin_Hron.pdf` – pdf dokument s výslednou bakalářskou prací včetně příloh

PostgreSQL

- *Linux* – složka s instalačními soubory PostgreSQL pro Linux
  - `postgresql-9.3.4-3-linux-x64.run` – instalační soubor PostgreSQL pro 64 bitový systém
  - `postgresql-9.3.4-3-linux-x86.run` – instalační soubor PostgreSQL pro 32 bitový systém
- *Windows* – složka s instalačními soubory PostgreSQL pro Windows
  - `postgresql-9.3.4-3-windows-x64.exe` – instalační soubor PostgreSQL pro 64 bitový systém
  - `postgresql-9.3.4-3-windows-x64.exe` – instalační soubor PostgreSQL pro 32 bitový systém
  - *PgAdmin3* – složka s instalačním souborem programu PgAdmin3