TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

WEBOVÁ APLIKACE PRO SKLADOVOU LOGISTIKU

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Jan Kozánek**
Vedoucí práce: doc. RNDr. Pavel Satrapa, Ph.D.

Liberec 2015

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

WEB APPLICATION FOR WAREHOUSE LOGISTICS

Bachelor thesis

Study programme: B2646 – Information Technology
Study branch: 1802R007 – Information Technology

Author: **Jan Kozánek**
Supervisor: doc. RNDr. Pavel Satrapa, Ph.D.

Liberec 2015

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2014/2015**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jan Kozánek**
Osobní číslo:	**M12000149**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Webová aplikace pro skladovou logistiku**
Zadávající katedra:	**Ústav nových technologií a aplikované informatiky**

Zásady pro vypracování:

1. Seznamte se s problematikou skladové logistiky a proveďte rešerši existujících softwarových nástrojů.
2. Navrhněte webovou aplikaci pro logistiku několika lokálních skladů umožňující sledování a úpravy jejich aktuálního stavu.
3. Aplikace by měla podporovat autentizaci uživatelů a přidělování jejich oprávnění na základě rolí.
4. Aplikace by měla být koncepčně připravena na lokalizaci do různých jazyků.
5. Návrh implementujte a otestujte.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **30 - 60 stran**

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

Seznam odborné literatury:

[1] GILMORE, W. Velká kniha PHP 5 a MySQL: kompendium znalostí pro začátečníky i profesionály. Nové, 3. vyd. Překlad Jan Pokorný. Brno: Zoner Press, 2011, 736 s. Encyklopedie Zoner Press. ISBN 978-80-7413-163-9.
[2] ZAKAS, Nicholas C, Jeremy PCPEAK a Joe FAWCETT. Ajax: profesionálně. Vyd. 1. Překlad Jiří Koutný. Brno: Zoner Press, 2007, 668 s. ISBN 978-80-86815-77-0.
[3] DUCKETT, Jon. Javascript: interactive front-end web development. 1. vyd. Indianapolis, IN: John Wiley, 2013, 640 s. cm. ISBN 11-185-3164-7.
[4] TATROE, Kevin, Rasmus LERDORF a Peter MACINTYRE. Programming PHP. 3. vyd. Sebastopol, CA: O'Reilly Media, 2013, xxii, 514 s. ISBN 14-493-9277-6.

Vedoucí bakalářské práce: **doc. RNDr. Pavel Satrapa, Ph.D.**
Ústav nových technologií a aplikované informatiky

Datum zadání bakalářské práce: **20. října 2014**

Termín odevzdání bakalářské práce: **15. května 2015**

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

prof. Dr. Ing. Jiří Maryška, CSc.
vedoucí ústavu

V Liberci dne 20. října 2014

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 14. 5. 2015

Podpis: Kozanek

Poděkování

Rád bych poděkoval panu Doc. RNDr. Pavlovi Satrapovi, Ph.D. za pomoc při psaní bakalářské práce a za jeho rady.

Abstrakt

Tato bakalářská práce popisuje vývoj a ovládání interaktivní webové aplikace http://www.skladovalogistika.php5.cz pro logistiku několika lokálních skladů.

Aplikace umožňuje úpravu zboží na skladě, jejich přesouvání mezi sklady, vytváření objednávek, vydávání a přijímání zboží a generování příslušných dokladů.

Při programování byla použita softwarová architektura MVC, programovací jazyk PHP, jazyky HTML5, CSS3, Javascript a databáze MySQL.

Klíčová slova

Webová aplikace, MVC, HTML5, PHP, databáze MySQL, systém skladové logistiky

Abstract

This bachelor thesis describes a development and interface of an interactive web application http://www.skladovalogistika.php5.cz which is used for logistics of several local warehouses.

The application allows editing of stocked articles in the warehouse, their transfers between them, creating orders, giving and receiving of goods and generating corresponding receipts.

For programming, there was used an MVC software architecture, programming language PHP alongside with HTML5, CSS3, Javascript and MySQL database.

Keywords

Web application, MVC, HTML5, PHP, MySQL database, warehouse logistics system

Obsah

Seznam obrázků

Seznam zkratek a symbolů

MVC – Model, View, Controller (Softwarová architektura rozdělující datový model, řídící logiku a uživatelské rozhraní)
PHP – Hypertextový preprocesor (Skriptovací jazyk sloužící k programování dynamických webových stránek)
HTML – HyperText Markup Language (Značkovací jazyk)
CSS – Cascading Style Sheet (Kaskádový styl)
jQuery – Javascriptová knihovna
JSON – Javascript Object Notation (způsob zápisu dat)
SQL – Structured Query Language (standardizovaný dotazovací jazyk používaný v databázích)
PK – Primary key (primární klíč)
FK – Foreign key (cizí klíč)
AI – Auto Increment
Framework – Aplikační rámec (SW struktura, sloužící jako podpora při programování)
AJAX – Asynchronous Javascript and XML (asynchronní zpracování webových stránek)

1. Úvod

Cílem bakalářské práce bylo vytvořit interaktivní webovou aplikaci pro logistiku několika lokálních skladů a umožnit sledování a upravování jejich aktuálních stavů.

Uživatelé se přihlašují pod svými účty, na kterých jsou různé úrovně oprávnění. V závislosti na oprávnění mají přístup do různých částí systému. Mezi tyto role patří obchodníci a logistika.

Obchodník má možnost sledovat zásoby jednotlivých skladů a přidávat do systému objednávky. Logistika může měnit obsah jednotlivých skladů, žádat o přesun zboží mezi jednotlivými sklady a v poslední řadě se stará o výdej a příjem zboží z již zmiňovaných objednávek.

Všechny tyto části se nacházejí v jednoduchém a intuitivním uživatelském prostředí, které je koncepčně připraveno na lokalizaci do více jazyků. V základním stavu se nachází čeština a angličtina.

Toto téma jsem si vybral, protože mě programování webových aplikací zajímá. Zaměřoval jsem se na jednoduché uživatelské rozhraní bez jakýchkoliv rušivých elementů a na to, aby bylo pro uživatele co nejjednodušší na použití a nemusel se pro jeho obsluhování dlouho zaškolovat.

2. Aktuálně dostupné systémy

Podobných systémů skladové logistiky se na trhu vyskytuje několik. Ne ke každému je však jednoduché zjistit podrobnosti. U větších zahraničních firem je pro více podrobností potřeba kontaktovat obchodní oddělení, případně domluvit schůzku, na kterých teprve dojde k představení produktu a k domluvení ceny. Popisovány tedy budou systémy, ke kterým tyto informace byly dostupné. Jedná se o aplikace českých firem.

Asi nejznámějším systémem je Zápůjční sklad od pražské firmy ComSys Software [1]. Umožňuje kompletní evidenci odběratelů (adresy, místa určení, kontakty, dlužníky atd.). Dále evidenci materiálu, ve kterém lze vytvořit ceník, zahrnující prodejní ceny, skupiny zboží, procenta pronájmů, plochu a hmotnost bednění a to včetně fotodokumentace. V neposlední řadě také agendu nabídek a zakázek. U zakázek umožňuje jejich evidenci a tisk, vystavení výdejek a příjemek, případně také změnu sazeb. U agendy nabídek lze vystavit nabídkový list, včetně kalkulace a požadované zboží rezervovat. Cena za tento produkt je 19 500 Kč za licenci na 1 počítač. Při koupi licence na 2 počítače je poskytnuta sleva 20 %. Pro spuštění program vyžaduje instalaci doplňující runtime knihovny.

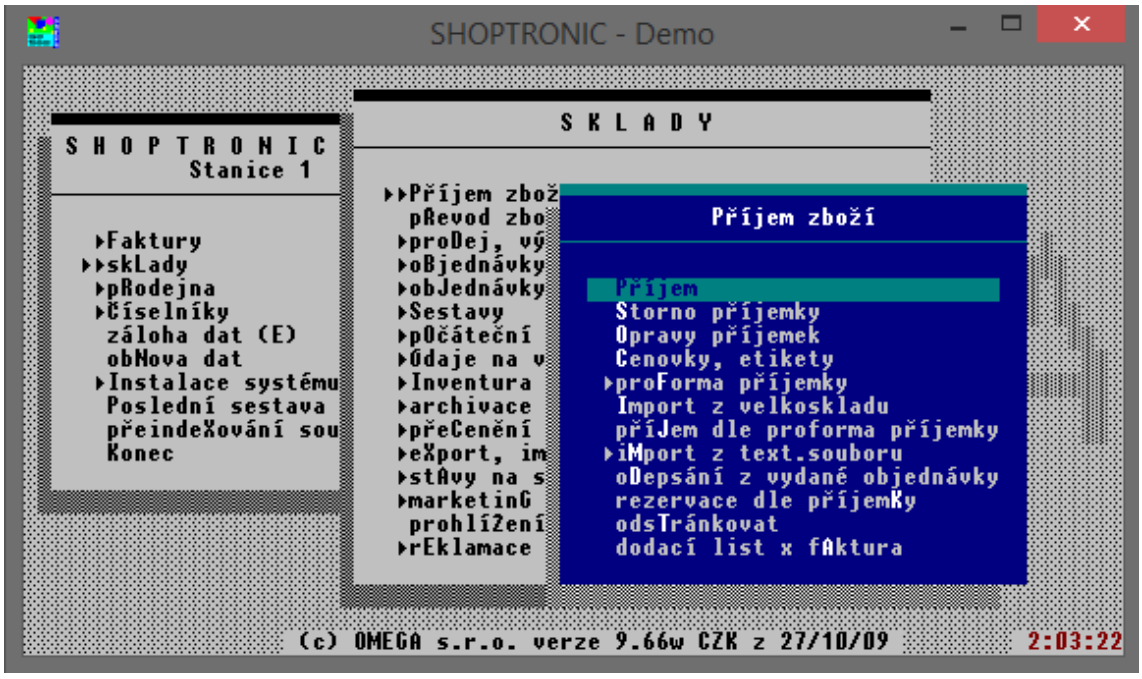
Dalším systémem je Shoptronic [2] od liberecké firmy Omega s.r.o. Tento systém již má víceuživatelské rozhraní. Jeho významnou vlastností je schopnost komunikovat se vzdálenými pracovišti (pobočky, dealeři), která jsou vybavena stejným typem softwaru. Tato pracoviště jsou spolu automaticky synchronizována v pravidelných intervalech, pomocí internetového připojení. Navíc obsahuje systém inventur a importu, či exportu. Shoptronic je nabízen v několika variantách. Pro maloobchod a pro velkoobchod. Cena za maloobchod činí 12 000 Kč a za velkoobchod 25 000 Kč. Dále je možné připlatit si zhruba 4 000 Kč a zákazník k softwaru dostane i servis od firmy Omega.

Posledním popisovaným softwarem bude skladový a fakturační software MAXim, od firmy MJ Soft, sídlící v Solnici [3]. Jedná se o plně síťový, modulově řešený program, který pracuje v prostředí Windows s databázovým serverem Firebird. Jeho největší výhodou je, že má integrovaný systém účetnictví. Tím je vhodný pro firmy, které samostatně nevedou účetnictví a zpracovává jim jej externí účetní. Celý systém pracuje v reálném čase, tzn., že každý uživatel v síti pracuje vždy s aktuálními daty. Obdobně jako software od firmy ComSys umožňuje tato aplikace přiřazení obrázků ke každému artiklu na skladě. Navíc obsahuje také evidenci a zpracování reklamací. Aplikace MAXim je nabízena ve 3 verzích (základní, rozšířená a max). Základní verze stojí 15 000 Kč, rozšířená 23 000 Kč a max stojí 29 000 Kč. Ceny jsou uváděny za 1 licenci. Za každou další licenci se platí polovina původní ceny.

2.1 Odůvodnění návrhu nového systému

Hlavní nevýhodou výše zmiňovaných systémů je jejich složitost. Jedná se o dlouho vyvíjené systémy. Jsou tedy vytvořeny pomocí dnes již zastaralých technologií a s postupem času na sebe „nabalily“ nové funkce, čímž se staly zbytečně složitými pro uživatele. Velmi často také obsahují funkce, které jsou pro danou firmu nepotřebné. Navíc nejsou jejich uživatelská prostředí pro uživa-

tele jednoduše pochopitelná. Například systém Shoptronic funguje v malém okně a vypadá jako stará aplikace pro systém DOS (viz obrázek 1). Má sice spoustu zajímavých a užitečných funkcí, ale velice špatně se používá.



Obrázek 1: Uživatelské rozhraní systému Shoptronic

To se ve svém systému snažím řešit. Systém by měl být pro uživatele jednoduchý, intuitivní, bez rušivých elementů a měl by vyžadovat minimální množství času potřebné k zaškolení. Nedosahuje tedy takové velikosti, jako některé dostupné programy, ale zaměřuje se především na základní funkce, které fungují jednoduše.

Další nepříjemností je cena. Ta se v tomto odvětví pohybuje okolo 20 000 Kč a zvyšuje se v závislosti na velikosti programu (Lite / Full aj.) a také v závislosti na značce firmy, která systém naprogramovala. Navíc je cena udávána pouze za jednu licenci. Když vezmeme v úvahu průměrnou firmu, kde se nachází zhruba 10 počítačů, tak počáteční náklady na zřízení takového systému budou 200 000 Kč. Vzhledem k tomu, že tento systém je dělán jako bakalářská práce, je poskytován zdarma. To je veliká výhoda oproti konkurenci. I kdyby se tomuto produktu měla v budoucnosti stanovit cena, bude jednotná za celý systém, nikoliv za licenci a to z toho důvodu, že je provozován na serveru, ke kterému lze přistupovat z jakéhokoliv internetového prohlížeče.

Navíc pokud firma vyvíjející daný software nesídlí v České Republice, tak může i nastat problém s lokalizací. Zákazník zakoupí například španělský software a má několik možností. Jednou z nich je, že se zaměstnanci budou učit pracovat v softwaru s cizím jazykem, což práci spíše přidělá. Druhou možností je využít lokalizace od výrobce, která však bývá neúplná, případně i nesprávná. Přeloženy jsou například pouze části, které samotná firma využívá a zbytek je ponechán v originálním jazyce. Poslední možností je, že se samotná firma, kte-

rá systém koupila, postará o překlad, což však také z ekonomického hlediska není výhodné. I tento problém je v této práci řešen tím, že je systém od začátku koncepčně připraven na lokalizaci do více jazyků a vyžaduje tak méně času na překlad.

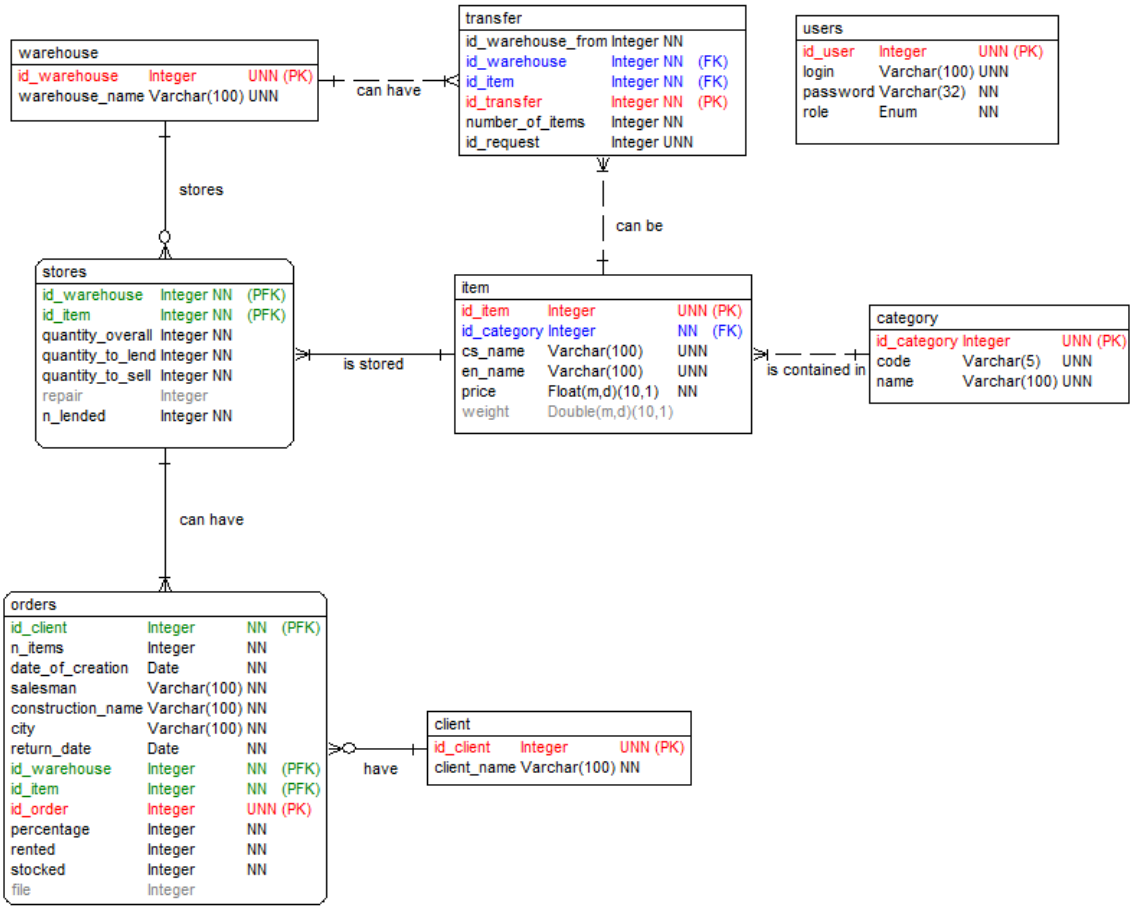
3. Databáze

3.1 Základní požadavky

Jak již bylo zmíněno, aplikace má umožňovat správu lokálních skladů. Musíme mít tedy nějakou tabulku s informacemi o jednotlivých skladech. Dále tabulku pro naskladněné artikly a pro jednotlivé kategorie, ve kterých se nachází. Navíc je nutné udržovat informace o konkrétních objednávkách. Budeme mít také tabulku s informacemi o zákazníkovi, spojenou s konkrétním skladem, ze kterého si artikly objednává a následně tabulku obsahující detailní informace o objednávce. Dále je potřeba uchovávat informace o případném přesunu zboží mezi sklady. Samostatnou tabulkou budou informace s přihlašovacími údaji jednotlivých uživatelů.

3.2 Databázový model

V programu CASE Studio 2 byl vytvořen relační model celé databáze. Ten popisuje základní požadavky na aplikaci kladené a zobrazuje vzájemné vztahy jednotlivých bloků. Každá entita (tabulka) obsahuje unikátní identifikátor, který je zároveň primárním klíčem pro jednoduchou identifikaci záznamu při dotazování.



Obrázek 2: Schéma relačního modelu

Tabulka *warehouse* obsahuje základní informace o jednotlivých skladech v systému. Obsahuje pouze identifikátor, kterým je celé číslo a následné jméno skladu.

Entita *item* obsahuje záznamy, popisující zboží, jež může být uloženo na sklad. Primárním klíčem je unikátní identifikátor. Entita dále obsahuje název zboží v češtině a angličtině (případně v dalších jazycích) a cenu spolu s váhou ve formátu double.

Každá položka se nachází v nějaké kategorii. Proto využíváme tabulku *category*, kde se nachází název kategorie ve Stringu a jejich unikátní kód v typu Varchar. Primárním klíčem je *id_category*.

Předešlé dvě entity *item* a *warehouse* jsou spojeny identifikační relací M:N, což znamená, že sklad může mít naskladněn jeden a více artiklů a stejně tak jeden předmět může mít naskladněn jeden nebo více skladů. Použitím M:N relace dojde k vytvoření nové, tzv. slabé entity. Ta se nazývá *stores* a jsou v ní uloženy údaje o stavu předmětu v daném skladu.

V této entitě dochází ke spojení primárních klíčů dvou spojovaných tabulek do nových *primárních cizích klíčů* (Primary Foreign Key). V této tabulce tedy nelze vyhledávat jednotlivé záznamy podle identických id, ale je potřeba dotazovat se pomocí tabulek, ve kterých se nachází primární klíče. Níže lze vidět ukázku SQL dotazu, který spojí tabulku *stores* a *item* podle jejich PK a vypíše všechny údaje ze *stores*, které mají požadované id skladu:

```
SELECT * FROM stores
JOIN item ON (item.id_item = stores.id_item)
WHERE id_warehouse = $id
```

Samotná entita *stores* tedy obsahuje dva primární cizí klíče (id předmětu a id skladu, ve kterém se nachází), dále *quantity_to_lend*, *quantity_to_sell, repair* a *n_lended*, což jsou údaje v datovém typu Integer obsahující informace o tom, kolik produktů je naskladněno a lze je dále prodat, půjčit, jsou v opravě anebo kolik jich je již zapůjčeno. Pro zobrazení celkového součtu jsou použity databázové objekty zvané pohledy (View). Ty poskytují data ve stejné podobě jako tabulka, ale na rozdíl od ní v ní nejsou uložena žádná data a obsahuje pouze předpis o tom, jakým způsobem mají být získána z ostatních tabulek. Ve výsledku se tedy jedná o jakousi virtuální tabulku.

Pohled *quantity2* obsahuje dotaz, ve kterém se nachází všechny výše zmiňované údaje o množství a poslední sloupec obsahuje jejich součet. Níže se nachází příkaz, kterým byl tento pohled vytvořen:

```
CREATE VIEW quantity2 as
SELECT stores.id_warehouse as id_warehouse, stores.id_item as
id_item, stores.quantity_to_lend as quantity_to_lend, sto-
res.quantity_to_sell as quantity_to_sell, stores.repair as
repair, stores.n_lended as n_lended, ((sto-
res.quantity_to_sell + stores.quantity_to_lend) + sto-
res.repair + stores.n_lended) as quantity_overall FROM stores
```

V databázi také uchováváme seznam klientů. Údaje o klientech se nachází v entitě *client.* Primárním klíčem je unikátní id *id_client* a jeho jméno je obsaženo ve Stringu *client_name*.

Jakýkoliv klient může mít zažádáno o jeden nebo více artiklů z jednoho skladu. Entita *klient* je tedy ve vztahu M:N k entitě *stores.* Tím dojde k vytvoření slabé entity *orders.*

Tabulka *orders* obsahuje 3 primární cizí klíče, kterými jsou id klienta, ke kterému se váže daná objednávka, id skladu, ze kterého mu budou artikly doručeny, a id artiklu, o který požádal. Primárním klíčem je číslo objednávky, které se vytvoří při zadání kontraktu do systému. Jedná se o unikátní číslo spojující celou objednávku pod jediné číslo. Mezi další údaje patří počet jednotlivých položek, datum vytvoření v datovém typu *Date*, jméno obchodníka, který daný obchod vytvořil, název stavby na kterou mají být artikly použity, název města a v případě, že je artikl pronajat, nikoliv prodán, tak procento z prodejní ceny a očekávané datum vrácení. Navíc uchováváme i hodnoty *rented* a *stocked*. Rented je počet artiklů, které má zákazník již zapůjčen a které tedy bude muset vrátit, a stocked je počet, kolik jich má být ještě zákazníkovi vydáno. Posledním údajem je *file*, což je celočíselná hodnota zobrazující poslední dostupné číslo souboru pro budoucí generování souborů.

Samozřejmě může dojít k situaci, kdy pro vytvoření objednávky nebude v jednom skladě dostatečný počet položek, ale v jiném ano. Uživatel má tedy možnost zažádat o přesun artiklů z jednoho skladu do druhého. Pro převody je použita entita *transfer.*

Ta spojuje tabulky *warehouse* a *item*. Zdědí tedy od nich dva cizí klíče, a to id skladu, do kterého chceme položky přesunout, a id jednotlivých položek. Také je uloženo id skladu, ze kterého chceme přesun provést, a počet jednotlivých položek. Celý převod je uložen pod jediným unikátním číslem, uloženým v *id_request*.

Poslední tabulka *users* obsahuje přihlašovací údaje jednotlivých uživatelů a jejich přidělenou roli. *Login* se nachází v datovém typu Varchar a *password* taktéž, ale v pevně dané délce 60 znaků. Obsahuje zašifrovanou podobu vytvořeného hesla pomocí algoritmu bcrypt, který je vestavěný v jazyce PHP.

Výše (viz obrázek 2) lze vidět popsané závislosti. Primární klíče jsou označeny červenou barvou, cizí klíče modrou barvou a cizí primární klíče zelenou barvou. Každý atribut má v prostředním sloupečku napsaný datový typ spolu s jeho délkou. Všechny atributy mají v posledním sloupečku nastaveno NN (Not Null), což znamená, že nesmí být ponechán prázdný. Pokud položka má nastaveno UNN, znamená to, že je unikátní a její hodnota se nesmí opakovat. Primární klíče mají rovněž atribut AI (Auto Increment), který zaručuje, že každý nový záznam bude mít svou hodnotu inkrementovanou o 1. Neinformativní relace je znázorněna přerušovanou čarou. Dále lze názorně vidět kardinalitu a parcialitu jednotlivých entit. Kardinalita určuje násobnost vztahu mezi dvěma entitami a parcialita jestli se entity ve vztahu musí nebo jen mohou nacházet.

3.3 MySQL

Pro ukládání dat bylo potřeba vybrat vhodný databázový systém. Zvolen byl systém MySQL, protože je multiplatformní, snadno implementovatelný, od počátku optimalizován pro rychlost, kompatibilní s jinými systémy a je především volně šiřitelný.

4. Jádro aplikace

Z důvodu jednoduchosti a přehlednosti kódu nebyl pro práci použit žádný dostupný PHP Framework. Místo toho byla architektura naprogramována ručně.
Úvodní stránka *index.php* slouží k propojení všech potřebných souborů pomocí funkce *require*:

```
require 'libs/Bootstrap.php';
require 'libs/Controller.php';
require 'libs/Model.php';
require 'libs/View.php';
require 'libs/database.php';
require 'libs/Session.php';
Session::init();
require 'config/paths.php';
require 'config/database.php';

$app = new Bootstrap();
```

4.1 Bootstrap.php

Soubor bootstrap.php se stará o správné parsování URL a přesměrování do controlleru. Je totiž žádoucí, aby URL byla čistá, přehledná, bez přebytečných znaků a od pohledu bylo poznat, na jaké stránce se nacházíme. O to se stará následující skript:

```
$url = isset($_GET['url']) ? $_GET['url'] : null;
$url = rtrim($url, '/');
$url = filter_var($url, FILTER_SANITIZE_URL);
$url = explode('/', $url);
```

Nejprve dojde ke zjištění, jestli se v adrese něco nachází nebo ne. Následně dojde k ořezání adresy podle lomítek a k jejímu rozdělení. Pomocí *filter_var se* docílí toho, aby se v adrese nenacházely nechtěné znaky (např. %), které by mohly být využity k napadení stránky. Výsledná adresa se rozloží do pole a podle délky pole dojde k zavolání controlleru s příslušným počtem parametrů.

Pokud je tedy například adresa ve tvaru „…/warehouse/show/9“, tak se načte v controlleru s názvem *warehouse* metoda *show*, která přebírá 1 parametr a tím je číslo 9. Lze tedy jednoduše posílat identifikátory, se kterými je potřeba pracovat. Stejným způsobem lze zavolat bezparametrovou metodu nebo i víceparametrovou metodu. Pokud bude URL adresa prázdná, dojde k přesměrování na úvodní stránku, tedy na index.

Spolu s načtením controlleru, dojde k načtení příslušného modelu. Je tedy nezbytné, aby se controller a model jmenovaly stejně.

Nakonec lze na kódu zobrazeném níže vidět, jak dojde k načtení správného controlleru. Z URL se vyjme název na nultém indexu, vloží se do pevně dané cesty, a pokud soubor na této adrese existuje, tak se vloží do stránky. Pokud ne, dojde k vyvolání chyby.

```
$file = 'controller/' . $url[0] . '.php';
    if (file_exists($file)) {
      require $file;
    } else {
      $this->error();
      return false;
    }
```

4.2 Controller.php

V souboru Controller.php se nachází jedna metoda s názvem *loadModel*, která přebírá jeden parametr. Tím je název souboru, který byl poslán z bootstrapu. Název souboru se vloží do předem zadané cesty a pro jednodušší rozlišení, jestli se jedná o controller nebo model, je za soubor přidána část „_model.php“. Pokud se na zadané cestě soubor nachází, tak se vloží do stránky, jinak dojde k zavolání chyby. Níže se nachází výše popsaná funkce:

```
public function loadModel($name) {
        $path = 'model/' .$name. '_model.php';

        if (file_exists($path)) {
             require 'model/' .$name. '_model.php';
             $modelName = $name . '_Model';
             $this->model = new $modelName();
        }
}
```

4.3 Model.php

V modelu se nachází pouze konstruktor, ve kterém se inicializuje přístup do databáze pomocí klíčového slova „db“. V jakémkoliv modelu poté stačí napsat *$this->db* a získáme tak přístup do databáze.

4.4 View.php

V souboru view.php se nachází jediná funkce s názvem *render*, přebírající jeden parametr, kterým je název stránky, kterou má tato funkce vykreslit.

Každá stránka v celém projektu, má stejnou hlavičku a patičku. Není tedy žádoucí, aby se na každé stránce tyto části opakovaly. Pokud by bylo potřeba upravit formát hlavičky, muselo by se tak učinit v každém souboru. Proto jsou hlavička a patička ve svých složkách jako samostatné soubory a pouze dochází k jejich připojování ke stránkám, které se mají vykreslit. Metodu lze vidět níže:

```
public function render($name) {
   require 'view/header.php';
   require 'view/' . $name . '.php';
   require 'view/footer.php';
}
```

4.5 Database.php

Tato třída se stará o připojení k databázi a používá k tomu ovladač PDO, od kterého dědí. Dlouhá léta bylo zvykem používat pro komunikaci prostou funkci *mysql_query [4][1].* Ta je však již zastaralá a od PHP verze 5.5 vypisuje chybu typu E_DEPRECATED. Tomu se dá vyhnout buď použitím novější funkce *mysql**i**_query* nebo právě ovladačem PDO, který byl v této práci použit.

PDO také ošetřuje spoustu bezpečnostních rizik, o která by se jinak programátor musel start sám. PDO totiž využívá předpřipravených dotazů. Samostatně posílá do databáze dotaz (prepare(INSERT INTO …)) a data. Data jsou chápána čistě jako data, nikoliv jako dotazy. Databáze se je tedy nesnaží analyzovat, jestli neobsahují nějakou instrukci. Nemůže tedy dojít k útoku typu SQL Injection, kdy místo dat, je poslána například instrukce na smazání databáze.

PDO umožňuje také používat PDO proměnné s dvojtečkou na začátku. U komplikovanějších dotazů to může zvýšit čitelnost a přehlednost kódu. Nejprve

tedy dojde k připravení SQL dotazu s prázdnými PDO proměnnými a až u samotného vykonání se propojí PDO proměnné s proměnnými v PHP pomocí asociativního pole.

V konstruktoru tohoto souboru dochází k připojení k databázi. Přihlašovací údaje jsou kvůli bezpečnosti nahrazeny konstantami a samotná definice konstant se nachází v samostatném souboru umístěném v adresáři *config/database.php.*

4.6 Session.php

Tento soubor obsahuje funkce pro jednodušší práci se *Session*. Abychom nemuseli používat volání $_SESSION, máme vytvořené jednoduché gettery, settery, konstruktor a destruktor. Metoda *set* přejímá dva parametry. První je jméno klíče, pod kterým bude session uložena, a druhý je údaj, který se v této session bude nacházet. Metoda *get* nám na zadaný klíč vrátí obsah session.

Sessions se v aplikaci využívají k hlídání přihlášeného uživatele. Jakmile se uživatel přihlásí, vytvoří se mu unikátní session id, které je aktivní dokud se neodhlásí nebo neopustí stránku. Pokud se uživatel nepřihlásí, nemá povolen přístup ke zbytku aplikace a rovněž nesmí přistupovat k údajům jiných uživatelů.

V poslední řadě jsou session používány k zobrazování informačních hlášení. Například když si nový uživatel úspěšně vytvoří nový účet, dojde k přesměrování a k vytvoření nové session zodpovídající za ono hlášení. Session se po vypsání textu okamžitě zničí, takže text se zobrazí pouze v momentě, kdy je potřeba, a není tedy nutné vytvářet kopii již existující stránky kvůli jednomu novému hlášení.

4.7 Config/paths.php

Tento soubor obsahuje definici konstanty URL, která je poté používána ve zbytku aplikace. Není tedy nutné při zadávání cest k souborům vypisovat vždy celou cestu, ale stačí zavolat konstantu pomocí klíčového slova *URL.*

4.8 Config/database.php

Zde se nachází již konkrétní údaje pro přihlášení do databáze. Údaje jsou uloženy v konstantách, které se používají v souboru *database.php*, který se připojuje k databázi.

5. Lokalizace

Aplikace je koncepčně připravena na lokalizaci do různých jazyků. V základním nastavení se nachází český a anglický jazyk. Požadavkem bylo vymyslet jednoduchý systém, který by byl jednoduše rozšiřitelný, a pro překládání nebyla potřeba znalost programovacích jazyků.

V úvahu byly brány dvě možnosti. Využití asociativního pole nebo bulit-in funkce gettext(). Za hlavní nevýhodu gettextu považuji jeho složitost a nutnost mít nainstalovaný samostatný program pro práci s **.po* (Portable Object) soubo-

ry. Naproti tomu při využití asociativního pole lze soubory rychle editovat přímo ve vývojovém prostředí a není potřeba instalace doplňků.

5.1 Inicializace

Ve složce *lang* se nachází soubor *init.php*, který obsahuje inicializaci překladů. Nejprve dojde k vytvoření pole, do kterého jsou vloženy názvy povolených jazyků.

Jazyk lze kdykoliv přepnout kliknutím na požadovaný jazyk v hlavičce stránky. Kliknutím se odešle *lang?=[nazev_jazyka],* ke kterému můžeme přistoupit pomocí funkce GET. Nejprve dojde k ověření, jestli se v proměnné *lang* něco nachází a pokud ano, tak zdali je to jeden z povolených jazyků. Pokud jsou obě podmínky splněny, vytvoří se session s názvem zvoleného jazyka, která je aktivní do té doby, než uživatel vybere jiný jazyk. Název jazyka je ukládán do session, aby na všech stránkách byl nastaven stejný jazyk.

Pokud v URL není nic nastaveno, což znamená, že je uživatel na stránce poprvé, tak se nastaví defaultně anglický jazyk.

Nakonec podle zvoleného jazyka dojde k vložení (include) správného jazykové balíčku. Níže se nachází ukázka kódu:

```
$allowed_lang = array('czech', 'english');

if (isset($_GET['lang']) === true
&& in_array($_GET['lang'], $allowed_lang) === true) {
   Session::set('lang', $_GET['lang']);
} else if (isset($_SESSION['lang']) === false) {
   Session::set('lang', 'english');
}
include 'lang/' . $_SESSION['lang'] . '.php';
```

5.2 Jazykový balíček

Samotné jazykové soubory se nachází ve složce *lang/lang*. Každý balíček je samostatný php soubor, ve kterém se nachází asociativní pole. Klíče tohoto pole jsou používány v pohledu v HTML kódu místo statických řetězců a hodnoty tohoto pole jsou přeložené výrazy, které se v aplikaci zobrazují:

```
$lang = array(
   'contracts' => 'Kontrakty',
   'warehouses' => 'Sklady',
    …
}
```

Pokud bychom chtěli tedy přistoupit k výrazu „Sklady“, použijeme v pohledu proměnnou $lang[‘warehouses‘].

6. Popis aplikace

Celý projekt byl napsán v jazyce HTML5 a byl na něj aplikován nejnovější kaskádový styl CSS3. V aplikaci jsou totiž hojně využívány grafické prvky, které ve starších verzích nejsou podporovány. HTML5 umožňuje používání moderních technologií jako například AJAX, obsahuje novou sémantiku zaměřující se hlavně na přehlednost kódu a přidává spoustu nových atributů do formulářů, které jsou rovněž v systému často používány.

6.1 Přihlašovací obrazovka

Přihlašovací obrazovka se skládá z hlavičky, ve které se nachází název aplikace, z patičky s informacemi o stránce a ze samotného těla. Dokud není uživatel přihlášen, jsou mu všechny další prvky skryty.

V těle dokumentu se nachází přihlašovací formulář. Formulář obsahuje pole pro jméno a heslo a potvrzovací tlačítko.

Formulář má nastavenou akci „index/login“ a data se přenášejí metodou *POST*. Po odeslání těchto údajů dojde k otevření controlleru s názvem *index* a k zavolání metody *login*. Tam pomocí již zmíněného volání POST přiřadím zadané údaje do proměnných. Z databáze si vezmeme hash uživatelova hesla a přiřadíme ho do proměnné. Jedná se o hash vypočítaný pomocí vestavěné funkce *password_hash*, nacházející se v PHP od verze *5.5.* Tato funkce využívá šifrovací algoritmus *bcrypt*, spolu s přidáním takzvané *soli*. Sůl je v šifrování několik náhodných bitů, které slouží jako doplňující vstup při použití jednosměrné funkce, čímž umožňuje, aby její výstup měl více možných variant. Stejné heslo tedy bude pokaždé jinak zašifrováno.

Dnes již není doporučené používat známé šifrovací algoritmy [12], mezi které patří například *MD5, sha256* a jiné, protože již nejsou bezpečné. Kdyby útočník nějakým způsobem získal přístup k databázi s těmito hashi, tak je dnes lze dešifrovat zpět do čitelné podoby. Výše zmiňované šifrovací algoritmy jsou navrhnuty, aby byly velice rychlé. Na moderních počítačích tak lze pomocí „brute force“ útoků tyto hashe velice jednoduše prolomit. Proto je doporučeno používat složitější hashovací algoritmy, spolu s kryptografickou solí. Mezi ně patří například již zmiňovaná vestavěná PHP funkce password_hash [13].

Pokud bychom chtěli heslo zpětně ověřit, využije se k tomu vestavěná funkce *password_verify*. Tato funkce přebírá dva parametry. Prvním je uživatelem zadané heslo a druhým je hash hesla uloženého v databázi. Pokud se heslo shoduje s hashem, vrátí funkce hodnotu *true*.

Pokud jsou přihlašovací údaje správně zadány, dojde k vytvoření nové session *loggedin*, jejímž obsahem je role, kterou daný uživatel má v systému.

V jiném případě dojde k navrácení na přihlašovací obrazovku a vypsání chyby.

Funkce starající se o přihlášení se nachází níže:

```
function login() {
    $login = $_POST['login_login'];
    $password = $_POST['login_password'];

    $check = $this->model->getPassword($login);
```

```
    $role = $check['role'];
    $hash = $check['password'];

    $over = password_verify($password, $hash);

     if ($over) {
       Session::set('loggedin', $role);
       header('location: ../index');
     } else {
       //chyba
      Session::set('login-error', true);
       header('location: ../index');
     }
 }
```

Funkce, která ověřuje uživatelův hash a roli je volána z modelu. Z kódu níže lze vidět, že se jedná o jednoduché volání, které vrací potřebné údaje. Nejprve si připravíme SQL dotaz pomocí funkce *prepare.* Samotný dotaz provedeme až funkcí *execute*, do které vložíme asociativní pole s proměnnými v PHP. Funkce vrátí pole výsledků, ve kterém se, pokud uživatel existuje, bude nacházet jeho role a zašifrované heslo.

```
public function getPassword($login) {
          $sql = $this->db->prepare("SELECT password, role FROM
users WHERE login = :login");
          $sql->execute(array(
                ':login' => $login
         ));
         return $sql->fetch(PDO::FETCH_ASSOC);
}
```

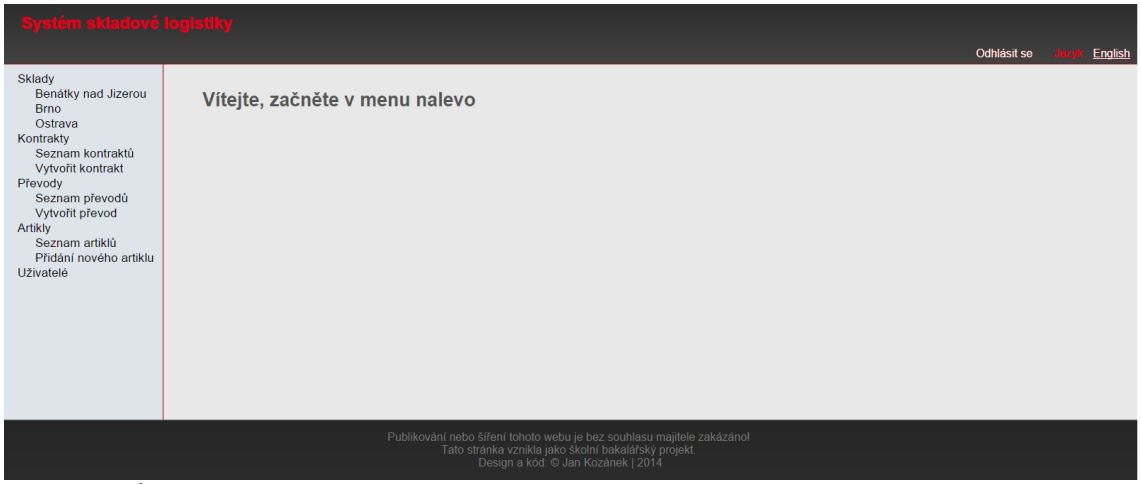
6.2 Hlavní stránka

Po přihlášení se uživateli zpřístupní menu nacházející se v levé části aplikace. Obsah menu se liší v závislosti na přidělené roli. Obchodník má možnost prohlédnout si stav všech skladů, vytvářet nové objednávky a prohlížet si stav aktivních objednávek. Logistika má obdobně jako obchodník přístup k aktivním kontraktům, nemůže ale přidávat nové objednávky. Navíc má také přístup k seznamu artiklů v databázi, může nové artikly naskladňovat na sklad a může provádět mezi jednotlivými sklady přesuny zboží. Administrátor má přístup k celému systému a navíc má přístup do administračního menu, kde může vytvářet a upravovat jednotlivé uživatele.

Na menu byl aplikován jednoduchý Javascriptový skript zobrazený níže, který při kliknutí na hlavní *ul* prvek menu plynule otevře. Například při kliknutí na „Sklady“ dojde k otevření menu, ve kterém se nachází všechny sklady v systému, ze kterých si uživatel již může konkrétně vybrat. Bylo jen potřeba ošetřit, aby efekt neprobublával i do dalších elementů.

```
$(function() {
        $('#MainMenu > li').click(function(e) {
            var $el = $('ul',this);
            $el.stop(true, true).slideToggle(250);
            return false;
        });
        $('#MainMenu > li > ul > li').click(function(e) {
            e.stopPropagation();
        });
});
```

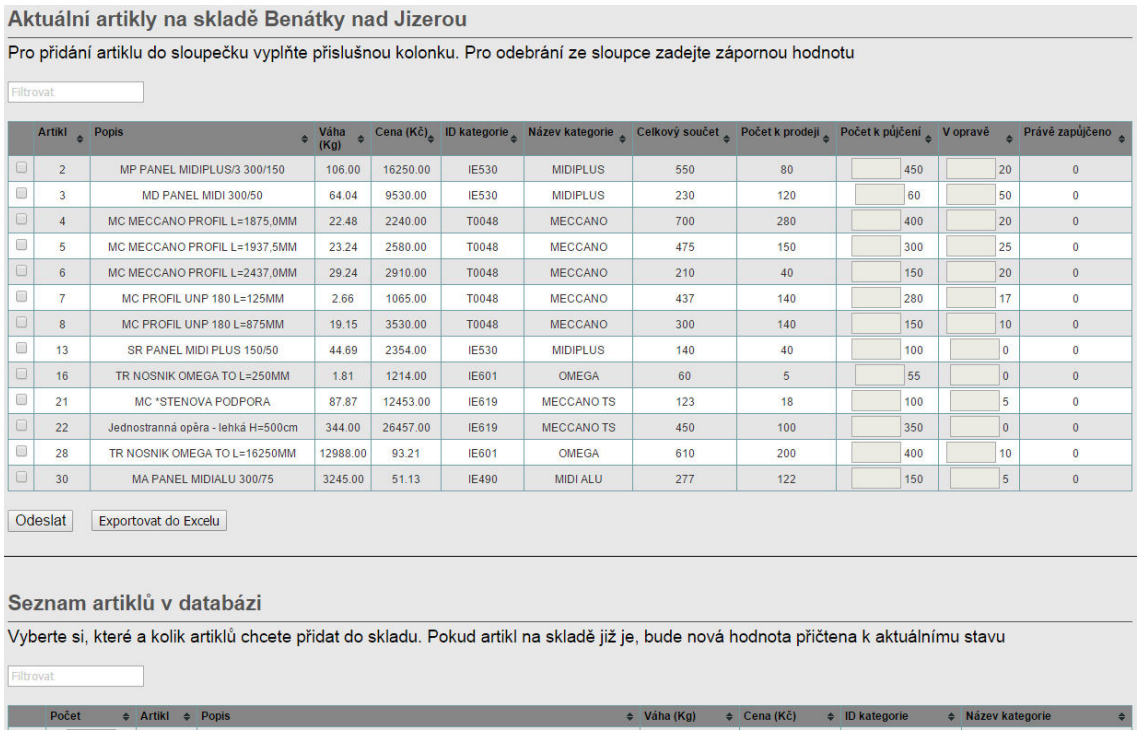
V pravém horním rohu se nachází tlačítko na odhlášení. Po jeho rozkliknutí dojde ke zrušení session *loggedin*, čímž uživatel přijde o možnost otevřít jakoukoliv další stránku, kromě přihlašovací obrazovky. Napravo od tohoto tlačítka se navíc nachází možnost přepínání jazyka.



Obrázek 3: Úvodní stránka po přihlášení

6.3 Sklady

Po vybrání konkrétního skladu se otevře stránka s přehledem jednotlivých artiklů na skladě. Stránka je rozdělena do dvou bloků, nacházejících se pod sebou. V prvním se nachází interaktivní tabulka s přehledem již zmiňovaných artiklů. V druhém bloku se taktéž nachází tabulka, která však umožňuje přidávání artiklů z databáze systému na sklad.



Obrázek 4: Hlavní stránka skladů

6.3.1 Filtrování obsahu

V prvním bloku se nachází název skladu, ve kterém se aktuálně nacházíme. Pod ním je pole <input>, které umožňuje jednoduché filtrování výsledků tabulky. Využívá k tomu *AJAX light table filter [6]*, takže se výsledky zobrazí okamžitě bez potřeby znovu načítat stránku.

```
  <input type="search" class="light-table-filter" data-
table="itemall" data-table-columns="1,2,3,4,5,6,7,8,9,10,11"
placeholder="Filtrovat">
```

Input využívá elementu *HTMLElement.dataset,* který posílá data do javascriptu, který se stará o třídění tabulky. Data-table-columns například obsahuje informace o tom, podle kterých sloupců lze tabulku filtrovat a data-table název tabulky, na kterou se javascript bude vztahovat.

Tento způsob filtrování byl aplikován na všechny tabulky v celém systému. Níže lze vidět část kódu starající se o filtrování:

```
function _onInputEvent(e) {
  _input = e.target;
  var tables = document.getElementsByClassName
              (_input.getAttribute('data-table'));
  var columns = (_input.getAttribute('data-table-columns')
                || '').split(',');
  Arr.forEach.call(tables, function (table) {
     Arr.forEach.call(table.tBodies, function (tbody) {
```

```
        Arr.forEach.call(tbody.rows, function (row) {
            _filter(row, columns);
        });
    });
  });
}

function _filter(row, columns) {
  var text, val = _input.value.toLowerCase();
  if (columns.length) {
     columns.forEach(function (index) {
       text += ' ' + row.cells[index].textContent.toLowerCase();
     });
  } else {
     text = row.textContent.toLowerCase();
  }
row.style.display = text.indexOf(val) === -1 ? 'none' : 'table-
row';
}
```

Funkce nejprve najde požadovanou tabulku pomocí metody *getElementsBy-ClassName* a následně si rozdělí její sloupečky a uloží do proměnné. Rozdělí samozřejmě pouze ty sloupečky, které byly označeny pro filtrování v data elementu. Poté tabulku postupně prochází a filtruje její obsah.

6.3.2 Třídění obsahu

Pro jednoduchou práci s tabulkami bylo implementováno i jednoduché třídění tabulek podle hlavičky sloupce. Použila se k tomu volně dostupná knihovna s názvem *tablesorter.js [5].*

Tablesorter je dostupný pod dvojitou licencí. Jedná se o licence MIT a GPL. Knihovna ke správnému fungování vyžaduje ještě přidání knihovny jQuery. Tyto knihovny jsou připojeny v hlavičce HTML:

```
    <script src="<?php echo URL; ?>styles/js/jquery-1.11.0.js"
type="text/javascript"></script>
     <script src="<?php echo URL; ?>sty-
les/js/jquery.tablesorter.js"></script>
     <script src="<?php echo URL; ?>styles/js/jquery-
ui.js"></script>
```

Všechny sloupce, které lze třídit, mají v hlavičce malou šipku, která znázorňuje směr třídění. Ve výchozí pozici se zobrazují dvě šipky, pokud třídíme obsah od největšího po nejmenší, tak ukazuje šipka směrem dolů a pokud od nejmenšího po největší, tak nahoru. Kliknutím na příslušnou hlavičku tabulky, dojde k uspořádání dat. Navíc lze také třídit tabulku podle více sloupců najednou. Při podržení klávesy SHIFT a kliknutím na další hlavičku sloupce dojde k setřídění podle obou sloupců najednou.

To, které sloupce budou tříditelné a které ne, lze provést pomocí jednoduchého javascriptu:

```
$(document).ready(function() {
    $("#itemall_trans").tablesorter({
    headers: {
        3: {sorter: false},
        4: {sorter: false},
        5: {sorter: false},
        6: {sorter: false}
    }
    });
});
```

Na element s daným ID je aplikována funkce *tablesorter*, která má uvnitř řečeno, pro která čísla sloupců se vypne možnost třídění. V příkladu výše jsou tedy sloupce číslo 3, 4, 5 a 6 vypnuty. Třídění se nehodí používat u tabulek, ve kterých jsou některé buňky sloučeny, protože začne docházet ke grafickým problémům.

6.3.3 Export tabulek do Excelu

Může nastat situace, že bude potřeba obsah nějaké tabulky odeslat někomu, kdo nemá do systému vůbec přístup. Na všechny tabulky v systému je tedy aplikován Javascriptový skript umožňující export tabulky do formátu .xls.

Pod každou tabulkou se nachází tlačítko, které má jako *onclick* funkci nastavený převáděcí skript a jako parametr je vždy uveden název tabulky, kterou chceme převést.

Na začátku skriptu dojde k vytvoření souboru s předvyplněnými XML značkami, díky kterým bude soubor identifikován jako Excelovská tabulka.

Výstupem je další funkce, která se již stará o vyjmutí tabulky z dokumentu a její vložení do souboru. Nejprve dojde k záloze celé tabulky. Vzhledem k tomu, že některé tabulky mají speciálně upravený formát (rowspan, skryté řádky apod.), bylo potřeba tabulku projet řádek po řádku a nastavit všechny hodnoty rowspan na 1 a skryté buňky znovu zviditelnit. Poté již může být tabulka vložena do .xls souboru. Nakonec tabulku vrátím do původního stylu, v jakém se nacházela před exportem, a výsledný soubor je uživateli odeslán ke stáhnutí. Soubor si může libovolně pojmenovat a uložit kamkoliv do počítače. Níže se nachází výstupní funkce celého skriptu:

```
function(table, name) {
        var backup = table;
        $('#'+table+' tr').each(function(){
          $(this).find('td').each(function(){
              $(this).attr("rowspan", 1);
              $(this).show();
          })
          $(this).find('th').each(function(){
```

```
                $(this).attr("rowspan", 1);
                $(this).show();
            })
          })
          if (!table.nodeType) {
            table = document.getElementById(table);
          }
          var ctx = {worksheet: name || 'Worksheet', table: ta-
ble.innerHTML}
          window.location.href = uri + base64(format(template,
ctx));
          MergeCommonRows($('#'+backup));
          if (backup == "itemall_cont2") {
            $("#"+backup+" td:first-child").hide()
          $("#"+backup+" th:first-child").hide()
          }
  }
```

6.3.4 Práce s tabulkami

Jak již bylo zmíněno, v prvním bloku stránky se nachází tabulka s aktuálním přehledem zásob na skladě. Tabulka obsahuje id artiklu, název v požadovaném jazyce, cenu a váhu zboží a v jaké kategorii se nachází.

Posledních pět sloupců ukazuje, jak lze s jednotlivými položkami manipulovat. Lze vidět, kolik položek lze prodat zákazníkovi, kolik z nich jich lze zapůjčit, případně kolik jich je v opravně. Pokud je položka zapůjčena, tak se zobrazí ve sloupci „zapůjčeno“. Pomocí databázového pohledu je také zobrazen kompletní součet všech těchto sloupců.

O zobrazení dat z databáze se stará funkce *show*(*$id*)*,* nacházející se v controlleru. Funkce má jeden parametr, kterým je identifikátor skladu, jenž je zasílán do modelu, kde se používá jako podmínka pro zobrazení hodnot pouze daného skladu. Z modelu si tedy necháváme zaslat název skladu a jeho obsah. Do pohledu se posílá v poli pomocí následujícího příkazu:

```
$this->view->items = $this->model->getAllItems($id);
```

V pohledu máme následně přístup k proměnné *items*, kterou procházíme cyklem *foreach* a dynamicky tak vytváříme tabulku. Tabulka má staticky danou hlavičku a pouze její obsah je ve zmiňovaném cyklu, přičemž každou iterací dojde k přidání nového řádku tabulky.

Pro manipulaci s hodnotami obsahuje tabulka první sloupec se zaškrtávacím polem (checkbox). Pokud chce uživatel měnit hodnoty, musí nejprve položku zaškrtnout, aby bylo zřejmé, že chce upravovat právě tuto.

```
  <input type="checkbox" name="checked[]" value="<?= $va-
lue['id_item']?>" onclick="enableName(this, 'quantity2<?= $va-
lue['id_item']?>');" />
```

Každý checkbox má své unikátní id, které je stejné jako id artiklu. Zaškrtnutím pole se spustí metoda *enableName*, která jako parametr přebírá id checkboxu, které bylo zaškrtnuto.

Ve výchozí poloze jsou totiž editační pole pro úpravu počtu zboží zašedlá, aby do nich nemohl uživatel nic zapisovat. Kód, který povoluje a zakazuje používání polí je zobrazen níže:

```
function enableName(ctrl, txtId) {
   if (ctrl.checked)
       $('.' + txtId).removeAttr('disabled');
   else
       $('.' + txtId).attr('disabled', 'disabled');
}
```

Jedná se o jednoduché ověření. Pokud je checkbox s daným id zaškrtnutý, tak se následujícím polím odebere atribut *disabled*. V opačném případě se jim tento atribut přiřadí.

Následně pokud chce uživatel zvýšit počet artiklů, které chce půjčit, tak do příslušného sloupce vyplní hodnotu, o kterou se pole inkrementuje. Pokud má tedy v poli 250 položek a chce jich 10 přidat, jednoduše do pole napíše hodnotu 10 a klikne na tlačítko odeslat. Stejným způsobem funguje i odebírání, akorát s tím rozdílem, že zadá zápornou hodnotu.

Artiklů lze samozřejmě upravovat libovolné množství najednou. Z tohoto důvodu je název checkboxu nastaven jako pole.

```
name="checked[]"
```

Do controlleru tedy přijdou data uspořádána v poli a dojde k jejich postupnému zpracování. Nejprve si ověřím, zdali pole skutečně existuje a data byla odeslána. Pokud ne, dojde k přesměrování zpět do pohledu. Vzhledem k tomu, že uživatel má možnost vyplnit dvě kolonky, tak se může stát, že jednu nechá prázdnou, buď z důvodu, že jí nepotřebuje vyplnit, nebo na ní pouze zapomněl. Z tohoto důvodu si každé pole projdu a prázdné hodnoty nahradím nulami, abych stále mohl posílat dotazy do databáze.

Poté projedeme pomocí for cyklu celé pole zaškrtnutých hodnot, čímž v každé iteraci získáme id artiklu, který chce uživatel přidat, kolik jich chce zapůjčit a kolik jich chce poslat do opravny. Tyto hodnoty jsou zaslány do modelu, kde pomocí UPDATE dotazu dojde k inkrementaci zadaných hodnot. Díky tomu nevadí, pokud pole obsahuje 0, protože se zkrátka hodnota nezvýší. Nakonec je potřeba celkovou zadanou hodnotu odečíst ze zdroje, o což se stará funkce *substractFromToSell*. Níže se nachází ukázka kódu:

```
foreach ($_POST['n_to_lend'] as $i => $value) {
   if ($value === "") $_POST['n_to_lend'][$i] = 0;
}

foreach ($_POST['repair'] as $i => $value) {
   if ($value === "") $_POST['repair'][$i] = 0;
```

```
    }

    for ($i=0; $i < sizeof($_POST['checked']); $i++) {
       $id_article = $_POST['checked'][$i];
       $quantity_to_lend = $_POST['n_to_lend'][$i];
       $repair = $_POST['repair'][$i];
       $this->model->editItemsInWarehouse($id_warehouse,
$id_article, $quantity_to_lend, $repair);

       $sum = $quantity_to_lend + $repair;
       $this->model->substractFromToSell($id_warehouse,
$id_article, $sum);
    }

    //Dotaz volaný v modelu
    UPDATE stores
    SET quantity_to_lend = quantity_to_lend + :quantity_to_lend,
    repair = repair + :repair
    WHERE id_warehouse = :id_warehouse AND id_item = :id_article
```

V druhém bloku stránky, se nachází seznam všech artiklů, které jsou v databázi. K této části má přístup pouze uživatel s oprávněním logistiky. Tato tabulka funguje na podobném principu jako předchozí, ale slouží k nákupu zboží na sklad. Uživatel opět zaškrtne, kterou položku chce přidat na sklad, a zadá její počet. Pokud položku na skladě již má, dojde k jejímu navýšení, a zadaný počet kusů. V opačném případě se vytvoří úplně nový záznam, který se vloží do databáze.

Do controlleru tedy přijdou data uspořádána v poli a dojde k jejich postupnému zpracování. Nejprve se ověří, zdali pole skutečně existuje a data byla odeslána. Pokud ne, dojde k přesměrování zpět do pohledu. V opačném případě pole projdeme a pomocí funkce *array_filter* dojde k odstranění prázdných míst. Ta se v poli objeví, protože se odešlou i nezaškrtnuté artikly jako prázdné hodnoty. Následně se v poli resetují klíče, aby pole začínalo standardně od nuly. Tím, že uživatel může zaškrtnout položky v libovolném pořadí, dochází k zamíchání indexů. Dále projdeme cyklem pole zaškrtnutých hodnot o délce přijatého pole, přičemž v každém cyklu získáme id položky a její počet. Tyto informace jsou již zaslány do modelu, spolu s id skladu, ve kterém se uživatel nachází a dojde k inkrementaci záznamu v databázi. Níže se nachází ukázka kódu:

```
    if (isset($_POST['quantity']) && isset($_POST['checked'])) {
       $quantity = array_filter($_POST['quantity']);
       $quantity = array_values($quantity);
       $checked = $_POST['checked'];

     for ($i=0; $i < sizeof($checked); $i++) {
```

```
        $new_checked = $checked[$i];
        $new_quantity = $quantity[$i];
        $this->model->setItemsToWarehouse($id_warehouse,
$new_quantity, $new_checked);
    }
```

6.4 Artikly

6.4.1 Vytvoření artiklu

Do databáze systému je možné přidávat nové artikly, které lze poté přidávat na jednotlivé sklady. Tato možnost se nachází v menu v sekci „Artikly“ a v položce „Přidání nového artiklu“.

Na nové stránce se vykreslí formulář, do kterého již uživatel vyplní potřebné údaje. Jedná se o český a anglický název, prodejní cenu zboží, jeho váhu a v jaké kategorii produktů se nachází. Formulář má díky technologii HTML5 přesně dané typy, které lze do polí zadávat. Nelze tedy například do váhy nebo ceny zadávat znaky. Stejně tak je potřeba nenechat žádné pole prázdné, o což se stará atribut *required*.

Formulář po dokončení volá v controlleru *additem* metodu *add,* která se již stará o přidání nového artiklu do databáze. Je důležité ověřit, jestli se již zadaný artikl v databázi nenachází. Proto je potřeba podle jména zboží projet databázi a zeptat se, jestli se tam již takový záznam nenachází. Pokud dostaneme zpět prázdné pole, můžeme artikl vytvořit. V opačném případě se vytvoří nová session s id *artikl_existuje* a nastaví se jí hodnota na *true*.

Nakonec se při přesměrování zpět do pohledu ověřuje podmínka, jestli session s tímto id existuje a zdali je nastavena na true. Pokud je, tak se vypíše hláška velkým červeným písmem, že došlo k chybě, protože se uživatel snaží přidat produkt, který již v databázi existuje. Po vypsání této chyby dojde k okamžitému zničení Session, aby se toto hlášení nezobrazovalo stále dokola.

6.4.2 Seznam artiklů

Přidané artikly lze zobrazit v pod-menu „Seznam artiklů“. Na této stránce se nachází prostá tabulka, zobrazující všechny artikly nacházející se v databázi. Tabulku lze samozřejmě filtrovat a podle potřeby ji lze rovněž exportovat do Excelu. Jedná se o čistě informativní tabulku a nelze s ní tedy provádět více interakcí.

6.5 Převody

Pro jednoduchou správu zboží v celé republice má logistika možnost přesouvat zboží mezi jednotlivými lokálními sklady. Může totiž nastat situace, kdy zákazník zažádá o určitý typ zboží, který v daném skladu není naskladněn nebo ho má nedostatečný počet kusů a v jiném skladu v republice je k dispozici.

V tomto případě může uživatel s oprávněním logistiky zažádat o převod. Může tak učinit kliknutím na menu „Převodů“ a následně na položku „Vytvořit převod“. Dojde tak k otevření stránky s formulářem.

V první části formuláře se nachází výběrové rolovací pole s názvy všech skladů v systému. Uživatel má možnost vybrat si, ze kterého skladu chce provést přesun. Následně si vybere, do kterého skladu má položky přesunout. Aby uživatel nemohl vybrat dva stejné sklady, používám skript, který po vybrání první hodnoty znemožní vybrat druhou stejnou:

```
var select1 = select = document.getElementById( 'select1' );
var select2 = select = document.getElementById( 'select2' );

select1.onchange = function() {
    preventDupes.call(this, select2, this.selectedIndex );
};

select2.onchange = function() {
    preventDupes.call(this, select1, this.selectedIndex );
};

function preventDupes( select, index ) {
    var options = select.options,
        len = options.length;
    while( len-- ) {
        options[ len ].disabled = false;
    }
    select.options[ index ].disabled = true;
    if( index === select.selectedIndex ) {
        alert('You\'ve already selected the item "' + se-
lect.options[index].text + '".\n\nPlease choose another.');
        this.selectedIndex = 0;
    }
}
```

Nejprve pomocí *getElementByID* dojde k vybrání obou dvou inputů a na každý se použije *onchange* funkce. K jejímu zavolání tak dojde vždy, když uživatel změní svůj výběr. Následně se uvnitř funkce zavolá vlastní funkce *preventDupes*, která pošle jako první parametr informace o tom, z jakého selectu došlo k odeslání, v druhém parametru na jaký select chce funkci aplikovat a jako poslední parametr index vybrané hodnoty.

Výsledná funkce *preventDupes* projede celý *select* cyklem a všude zruší atribut disabled, aby nedošlo k zablokování více hodnot. Následně se vybere volba na požadovaném indexu a atribut *disabled* se nastaví na *true*.

6.5.1 jQuery autocomplete

Po vybrání obou skladů může začít uživatel zadávat prvky, které chce přesunout. Artikly se zadávají podle jejich ID a následně počet, kolik jich chce uživatel

přesunout. Aby uživatel nemusel spoléhat pouze na zadání správného ID, tak je použit *jQuery autocomplete [8]*, který po zadání hodnoty zobrazí pomocí AJAXU všechny artikly odpovídající zadanému id. Pokud tedy zadá například číslo 1, tak se mu zobrazí všechny výsledky, které obsahují číslo 1, nikoliv pouze začínající tímto číslem.

Na element *input* je aplikování následující skript:

```
$('input.auto').each(function() {
    $(this).autocomplete({
        source: "contract/auto/",
        minLength: 1
    });
});
```

Hodnota *minLength* určuje při kolika znacích, má dojít k automatickému doplňování. Vzhledem k tomu, že uživatel zadává krátká ID artiklů, tak je tato hodnota nastavena na 1.

Source odkazuje na místo, odkud bude skript brát data pro doplňování. V tomto případě jsem skript odkázal do controlleru s názvem *contract* a na metodu *auto*.

K hodnotám, které se asynchronně posílají, můžeme přistoupit pomocí globální metody _GET[‘term’]. V ní se nachází vždy nejaktuálnější hledaný výraz. Tento výraz je poslán jako parametr funkci *autocomplete*, která je poslána do modelu. V modelu dochází k prostému SELECT dotazu, avšak navíc s použitím operátoru LIKE. Ten se používá v kombinaci s operátorem WHERE, k bližší specifikaci hledaného výrazu. U výrazu lze použít znak (%), který zastupuje jakýkoliv znak. Lze tedy procento zapsat za hledaný výraz, čímž dojde k tomu, že výraz může být nahrazen libovolným řetězcem. V našem případě jsem procento využil před i za hledaným výrazem, z důvodu krátkých hledaných hodnot. Výsledky se ještě setřídí podle id vzestupně, a aby nedocházelo k odesílání velkého množství dat, tak množství výsledků omezím na 10:

```
public function autocomplete($search) {
        $sql = $this->db->prepare("SELECT id_item, cs_name, en_name FROM item WHERE id_item LIKE '%$search%' order by id_item asc limit 0,10");
        $sql->execute();
        return $sql->fetchAll(PDO::FETCH_ASSOC);
}
```

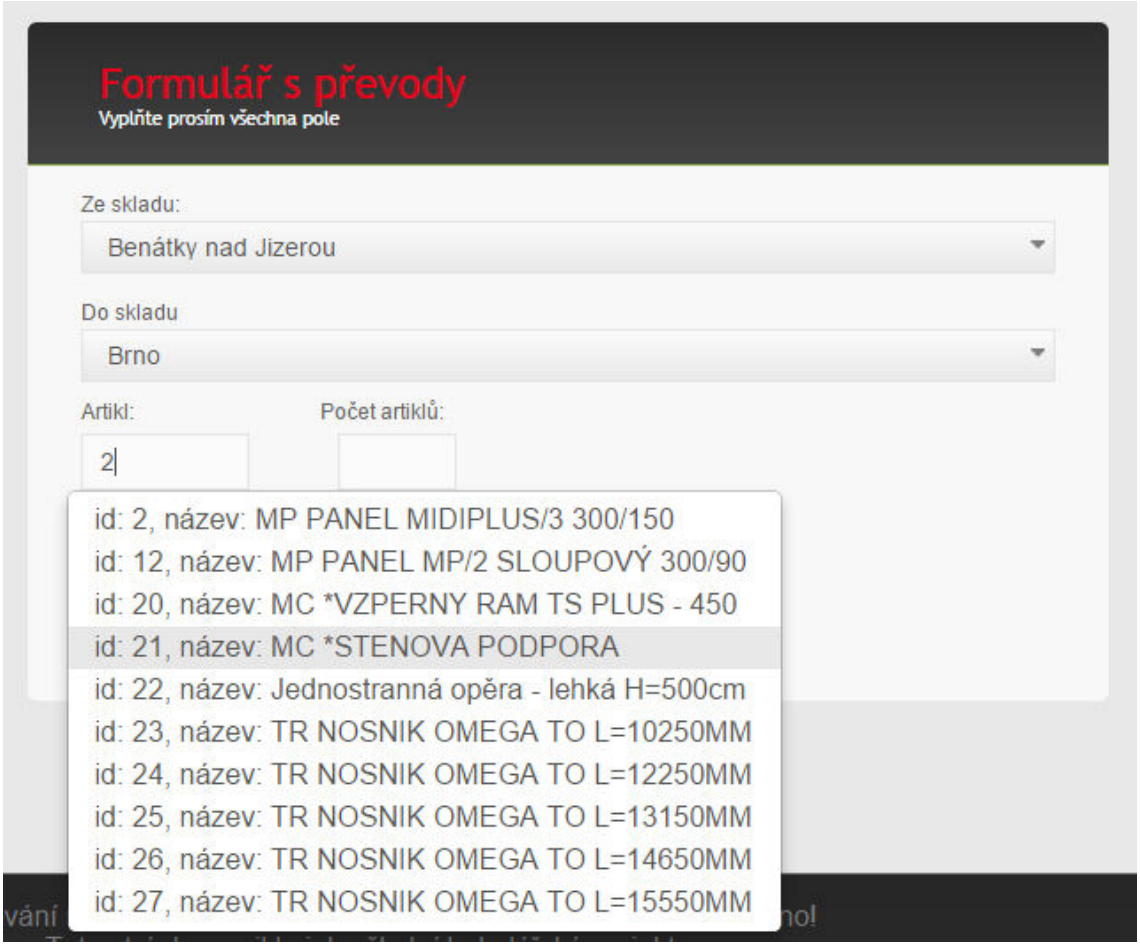
Jelikož autocomplete nedokáže pracovat čistě s výsledky databázového dotazu, je potřeba převést tyto hodnoty do JSON pole. Autocomplete však ještě pro korektní zobrazení vyžaduje použití pole, kde klíčem bude hodnota *‚value‘.* Bylo tedy potřeba ručně vytvořit nové pole, kde klíčem bude právě value a za ním až konkrétní výsledek.

Pro zobrazení popisku je třeba rovněž vytvořit klíč s názvem *‚label‘* a do něj přidat popisky. Pro zobrazení správného jazyka je použita Session, která zjistí použitý jazyk a podle toho zobrazí výsledek ve správném jazyku.

Takto dochází k dynamickému vytváření pole. Na konci každé iterace je použita funkce *array_push,* která přidá nový výsledek na konec jako u zásobníku. Na výsledné pole je použita funkce *json_encode*, která PHP pole přepíše do pole, které již lze používat v Javascriptu. Výsledek celého autocomplete se vypíše pomocí funkce *print_r*. Ukázka zmiňovaného kódu se nachází níže:

```
  function auto($id) {
      $neco = $this->model->autocomplete($_GET['term']);
      $array = array();
      $array_j = array();

      foreach ($neco as $i => $value) {
        $array['value'] = $neco[$i]['id_item'];
        if (Session::get('lang') == 'czech') {
          $array['label'] = 'id: '.$neco[$i]['id_item'].', ná-
zev: '.$neco[$i]['cs_name'];
        } else if (Session::get('lang') == 'english') {
          $array['label'] = 'id: '.$neco[$i]['id_item'].', name:
'.$neco[$i]['en_name'];
        }
        array_push($array_j, $array);
      }
      $data = json_encode($array_j);
      print_r($data);
   }
```



Obrázek 5: jQuery autocomplete

6.5.2 Dynamické vytváření polí formuláře

Uživatel má samozřejmě možnost přidat do převodu více než jeden artikl najednou. K tomu jsou ve formuláři dvě tlačítka „+“ (přidání) a „-“ (odebrání). Kliknutím na „+“ dojde k přidání dvou nových kolonek pro zadání ID a počtu kusů. Kliknutím na „-“ dojde k odstranění naposledy přidaného prvku.

Nové položky se přidávají do prázdného bloku * *, a řadí se pod sebe. Přidávání a odebírání položek obstarává Javascript. Pokud uživatel klikne na tlačítko s pevně daným identifikátorem, zavolá se výše zmiňovaný element div a pomocí funkce *append* se dovnitř nakonec připíše nové vstupní pole. Jedná se o input totožný s tím, který se nachází na začátku při načtení dokumentu. Navíc ještě musíme tomuto inputu přiřadit novou autocomplete funkci, protože ve výchozím stavu reaguje autocomplete pouze na první input. Následně dojde k připojení i inputu zaznamenávající počet kusů.

Obdobná funkce se nachází i u tlačítka pro odebrání prvků. Jednoduše dojde k nalezení prvku s pevně daným *class* jménem a zavolá se na něj funkce remove. Aby však nedošlo ke smazání i polí, která uživatel nepřidal tlačítkem k přidání, používá se selektor *:last*. Vzhledem k tomu, že mají všechny inputy stejný název, tak je opět použito pole, do kterého se všechny výsledky uloží pro pozdější zpracování. Výše popisovaný kód lze vidět níže:

```
  $(document).ready( function() {
      $("#add_item").click( function() {
          $("#next_items").append('<input   class="new_id"   type="text"  name="id_item[]"  placeholder="Article  ID"  required style="width: 15%;"/>');
          $("#next_items .new_id:last").autocomplete({
              source: "contract/auto",
              minLength: 1
          });
          $("#next_items").append('<input class="new_number" type="number"  name="quan_item[]"  min="1"  required  style="margin-left: 50px;"/><br class="new_br">');
      });

      $("#remove_item").click( function() {
          $(".new_id:last").remove();
          $(".new_number:last").remove();
          $(".new_br:last").remove();
      });
  });
```

Po potvrzení formuláře jsou výsledky poslány do controlleru, kde dojde k jejich zpracování. Nejprve dojde k ověření, zdali uživatel správně zadal názvy skladů. Poté dojde k vytvoření unikátního ID pro celý převod. Z databáze si zjistíme nevyšší id. Pokud id neexistuje, dojde k jeho vytvoření, jinak k inkrementaci. Následně opět for cyklem dojde k projetí odeslaného pole výsledků a v každé iteraci k přidání záznamu do databáze. V tuto chvíli ještě k samotnému převodu nedojde. V podstatě dojde k vytvoření žádosti, kterou je potřeba ještě schválit.

6.6 Seznam převodů

V této části převodů má uživatel možnost spravovat veškeré požadavky o převody. Tato sekce je dostupná pouze logistickému oddělení. Na stránce se nachází tabulka s přehledem zažádaných převodů. Tabulku lze již klasicky profiltrovat a její obsah exportovat do Excelu.

V tabulce je nejprve zobrazené číslo převodu, následně z jakého skladu chceme přesun provést. Následuje cílový sklad a seznam položek (id, název, počet). V poslední kolonce jsou dvě ikonky. Zelená fajfka převod potvrdí a červený křížek ho zamítne.

Aby tabulka byla v uživatelsky přijatelnější podobě a neměli jsme v ní pod sebou neustále opakující se hodnoty jako je číslo převodu, či název skladu, je použit skript, který buňky se stejným textem sloučí do jedné. Vzhledem k tomu, že je tabulka vytvářena dynamicky v závislosti na výsledku databázového dotazu, tak nelze přímo říci, kterým buňkám nastavit atribut *rowspan*. Z tohoto důvodu bylo potřeba vytvořit funkci, která si s tímto problémem poradí až po načtení stránky.

Nejprve v cyklu iterujeme skrz každý sloupec tabulky. Hodnota *rowspan* je na začátku nastavena na číslo 1. Následně pomocí funkce *find* projdeme n-tého potomka tabulky a u každého ověřujeme, jestli se tento řádek již nachází v poli. Pokud ne, rowspan se inkrementuje o 1 a celému řádku se přidá *class* jméno „hidden“. V opačném případě dojde k uchování *break* hodnot pouze pro první sloupec a nastavení *rowspan* na 1. Když takto projdeme celou tabulku, tak se nakonec na všechny prvky, obsahující třídu „hidden“ zavolá metoda *hide()*, čímž dojde k jejich schování. Výše popisovaný algoritmus lze vidět níže:

```
   function MergeCommonRows(table) {
       var firstColumnBrakes = [];
       for(var i=1; i<=table.find('th').length; i++){
          var previous = null, cellToExtend = null, rowspan = 1;
          table.find("td:nth-child(" + i +
")").each(function(index, e){
                var jthis = $(this), content = jthis.text();
                if (previous == content && content !== ""
   && $.inArray(index, firstColumnBrakes) === -1) {
                    jthis.addClass('hidden');
                    cellToExtend.attr("rowspan", (rowspan = row-
span+1));
                }else{
                    if(i === 1) firstColumnBrakes.push(index);
                    rowspan = 1;
                    previous = content;
                    cellToExtend = jthis;
                }
            });
       }
       $('td.hidden').hide();
    }
```

6.6.1 Interakce s převody

Pokud se uživatel rozhodne, že převod nepotvrdí, kliknutím na červený křížek v posledním sloupci dojde k jeho smazání. V případě, že převod chce schválit, je potřeba ověřit, zdali je převod korektní.

Aby se nemusely do controlleru odesílat všechny hodnoty pomocí POST metody, tak dojde k odeslání pouze id převodu. Následně si zjistíme z databáze veškeré údaje týkající se tohoto identifikátoru.

For cyklem se poté projede celé pole výsledků a u každého artiklu se ptáme modelu, jestli daný artikl v konkrétním skladu je a v jakém počtu.

Pokud je výsledek prázdný, znamená to, že na skladě tato položka není a nelze tedy provést tento převod. Dojde k vytvoření *session* s názvem „transferError“ a přesměrování zpět na pohled převodů, kde díky aktivní session dojde k vypsání chyby, že se uživatel snažil přesunout položku, která není momentálně na skladu. Stejná situace nastane v případě, že se uživatel bude snažit přesunout větší počet položek, než který je momentálně naskladněn.

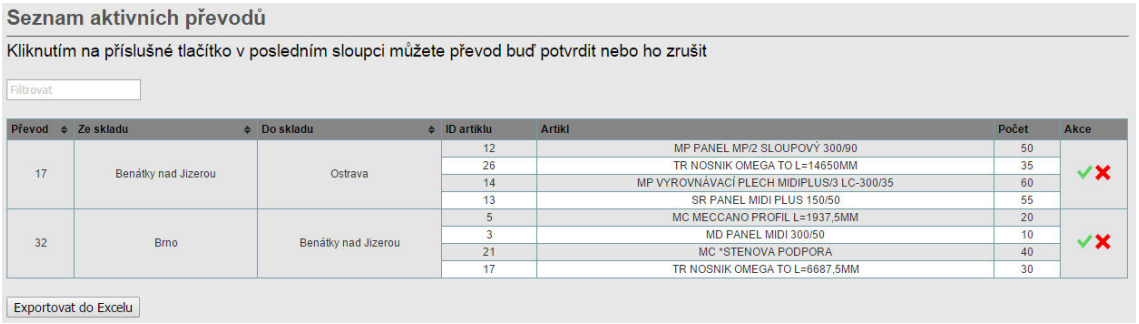
V opačném případě je potřeba položku odečíst z původního skladu a přičíst jí v novém skladu. Nejprve tedy zjistíme, zdali položka, kterou chceme převést do nového skladu již v novém skladě je nebo ne. Pokud ano, tak hodnotu inkrementujeme pomocí UPDATE dotazu, jinak vložíme zcela novou pomocí INSERT dotazu.

Nakonec je potřeba položku z původního skladu dekrementovat a celou žádost o převod smazat. Následně ještě dojde k vytvoření session, která uživateli při návratu do pohledu ohlásí úspěšný převod a dojde k přesměrování zpět do seznamu převodů. Celý tento postup lze vidět na kódu níže:

```
  function accept($id) {
      $results = $this->model->getTransferByID($id);
      for ($i=0; $i < sizeof($results); $i++) {
        $fromWarehouseID = $results[$i]['id_warehouse_from'];
        $toWarehouseID = $results[$i]['id_warehouse'];
        $item_ID = $results[$i]['id_item'];
        $quantity = $results[$i]['number_of_items'];
        $stocked = $this->model-
>getItemAndQuantityById($fromWarehouseID, $item_ID);
        if (empty($stocked)) {
          Session::set('transferError', true);
          header('location: ../../listtransfers');
          exit;
        } else {
          $already_exists = $this->model-
>itemExistsInDestination($toWarehouseID, $item_ID);

          $old_warehouse = $stocked[0]['quantity_to_sell'];
          if ($old_warehouse < $quantity) {
            Session::set('transferError', true);
            header('location: ../../listtransfers');
            exit;
          }
          if (empty($already_exists)) {
            $this->model-
>transferCreateNewArticleInWareHouse($toWarehouseID, $item_ID,
$quantity);
          } else {
            $this->model-
>transferAddToNewWarehouse($toWarehouseID, $item_ID, $quantity);
          }
          $this->model-
>transferDeductFromOldWarehouse($fromWarehouseID, $item_ID, $qu-
antity);
          $this->model->deleteSelectedTransfers($id);
        }
      }
      Session::set('transferSuccess', true);
```

```
        header('location: ../../listtransfers');
    }
```



Obrázek 6: Tabulka převodů

6.7 Kontrakty

Uživatelé s oprávněním obchodníků mají možnost do systému přidávat objednávky. Tato funkce logistice není přístupná. Po rozkliknutí menu „Kontrakty“ a pod-menu „Vytvořit kontrakt“ se uživateli zobrazí formulář na vyplnění kontraktu.

K úspěšnému vytvoření je potřeba vyplnit jméno klienta, pro kterého je zakázka tvořena, jméno prodejce, který obchod provedl, jméno stavby a město, do kterého materiál bude odeslán, následně z roletkového menu název skladu, ze kterého bude zboží odesláno, a poté už jen jednotlivé kusy zboží. Pokud si zákazník zboží pouze pronajímá, tak se vyplní i očekávané datum návratu. Obchodník má také možnost zadat u každého artiklu, zdali si ho zákazník bude pronajímat nebo celý kupovat a v případě pronájmu určit cenu procentem z prodejní hodnoty.

Zadávání artiklů je opět umožněno dynamicky. To znamená, že uživatel má možnost tlačítkem „+“ přidávat další pole, do kterých bude zadávat další kusy. Ty se přidají do prázdného elementu div obdobně jako u již zmiňovaných převodů, tedy Javascriptem, který do divu pomocí funkce *append* přidá část HTML kódu.

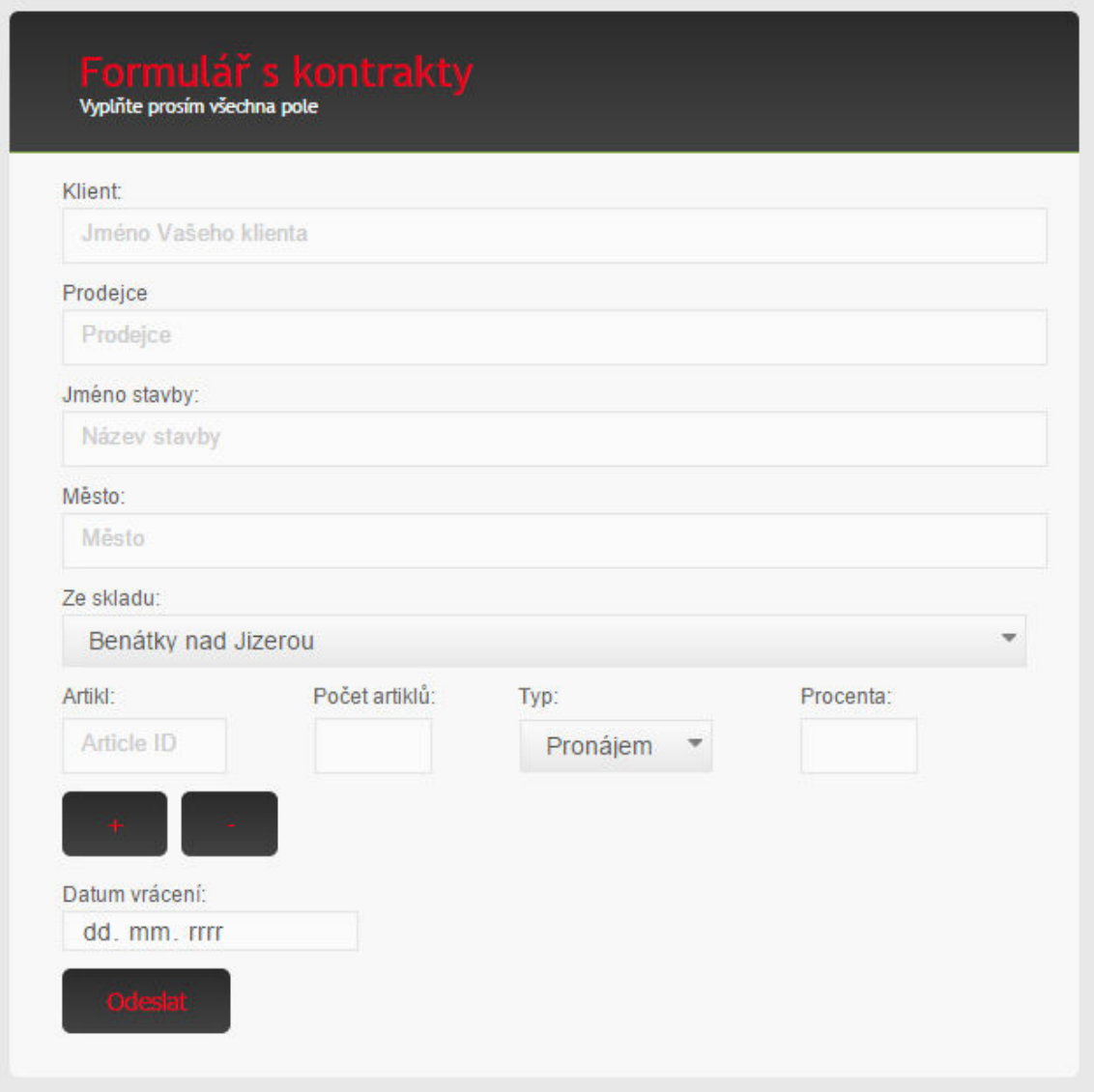
Kód artiklu opět umožňuje automatické doplňování pomocí *jQuery autocomplete* a stejně tak při přidávání nových polí do formuláře je potřeba tuto funkci inputu přiřadit. O vyplňování tohoto formuláře se zpravidla stará vnitřní obchodník, který dostane informace o tom, které konkrétní artikly zákazník vyžaduje. Má tedy již k dispozici seznam identifikátorů jednotlivých artiklů a nemusí zadávat zdlouhavé názvy, nýbrž pouze jejich identifikátory.

Informace vyplněné ve formuláři jsou odeslány do controlleru, kde k nim máme přes _POST přístup. Nejprve je potřeba zjistit, zdali zadaný zákazník je už v systému. Pokud je, tak si vezmeme jeho id, které používáme dále, jinak mu vytvoříme v databázi záznam.

Jelikož pole procent není povinné z důvodu, že při prodeji položky není potřeba zadávat procento z prodejní ceny, tak si celé pole procent projedeme cyklem a prázdné hodnoty nahradíme číslem 100. Tím se nám celé pole vyplní a lze s ním dále pracovat v databázi. Každá položka totiž bude mít daná procenta. Pronajaté artikly určité procento z prodeje a položky určené pro prodej budou mít nastaveny 100 %.

Dále by měla být možnost již existující kontakt rozšířit. To znamená, že pokud obchodník zadá jméno klienta, název stavby a město stejné jako je již v systému, tak si z databáze vezmeme id onoho kontraktu a dále pracujeme s ním, čímž dojde k jeho rozšíření. V opačném případě pracujeme s novým unikátním identifikátorem.

Nakonec projedeme cyklem pole přidaných artiklů a pro každý z nich zavolám z modelu metodu, pro přidání nové objednávky. Jakmile proiterujeme celé pole, dojde k přesměrování na seznam kontraktů. V tomto kroku stále nedochází k odečtení zboží ze skladu. Zatím došlo pouze k vytvoření objednávky. Její vydání lze provést v seznamu kontraktů.



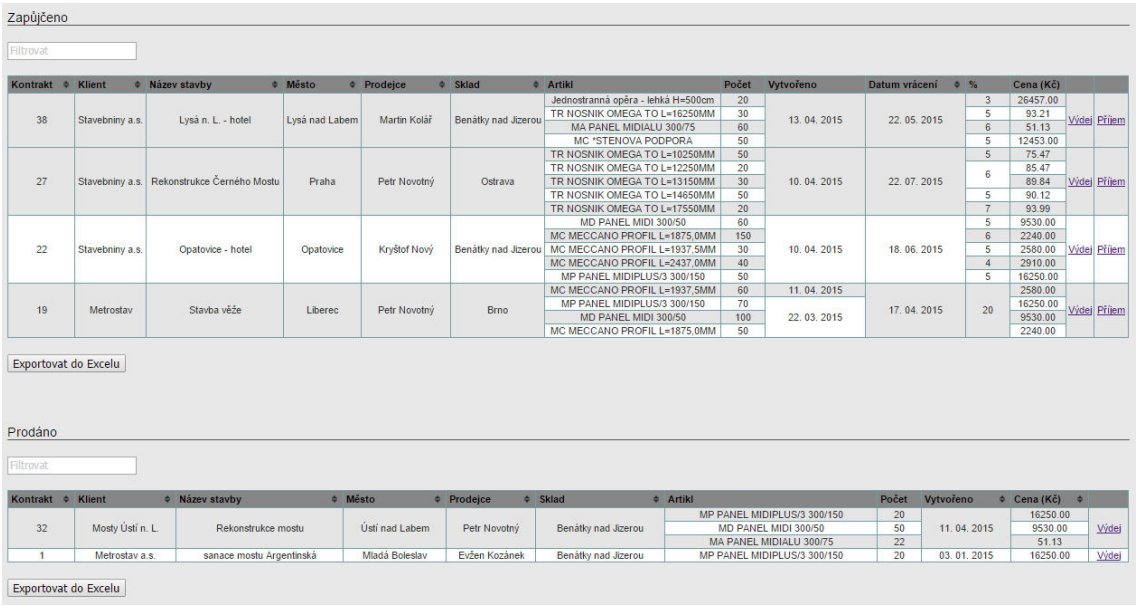
Obrázek 7: Formulář s kontrakty

6.7.1 Seznam kontraktů

Stránka se seznamem kontraktů obsahuje dvě tabulky. První tabulka obsahuje kontrakty, ve kterých si zákazník zboží pronajímá. Tabulka pod ní naopak zboží, které se má prodat.

Obě tabulky mají možnost filtrování obsahu pomocí AJAXu, třídění pomocí tableSortu, exportování do Excelu a pro lepší čitelnost jsou buňky spojené pomocí *merge* funkce.

Tabulky obsahují všechny informace, které byly obchodníkem při vytváření zadány. V posledních dvou sloupcích je možnost rozkliknout si podrobnosti kontraktu a to buď příjem, nebo výdej. Uživatelé s oprávněním obchodníků nemají k výdejkám a příjemkám přístup.


Zapůjčeno

Filtrovat

Kontrakt	Klient	Název stavby	Město	Prodejce	Sklad	Artikl	Počet	Vytvořeno	Datum vrácení	%	Cena (Kč)		
38	Stavebniny a.s.	Lysá n. L. - hotel	Lysá nad Labem	Martin Kolář	Benátky nad Jizerou	Jednostranná opěra - lehká H=500cm	20	13. 04. 2015	22. 05. 2015	3	26457.00	Výdej	Příjem
						TR NOSNIK OMEGA TO L=16250MM	30			5	93.21		
						MA PANEL MIDIALU 300/75	60			6	51.13		
						MC *STENOVA PODPORA	50			5	12453.00		
27	Stavebniny a.s.	Rekonstrukce Černého Mostu	Praha	Petr Novotný	Ostrava	TR NOSNIK OMEGA TO L=10250MM	50	10. 04. 2015	22. 07. 2015	5	75.47	Výdej	Příjem
						TR NOSNIK OMEGA TO L=12250MM	20			6	85.47		
						TR NOSNIK OMEGA TO L=13150MM	30				89.84		
						TR NOSNIK OMEGA TO L=14650MM	50			5	90.12		
						TR NOSNIK OMEGA TO L=17550MM	20			7	93.99		
22	Stavebniny a.s.	Opatovice - hotel	Opatovice	Kryštof Nový	Benátky nad Jizerou	MD PANEL MIDI 300/50	60	10. 04. 2015	18. 06. 2015	5	9530.00	Výdej	Příjem
						MC MECCANO PROFIL L=1875,0MM	150			6	2240.00		
						MC MECCANO PROFIL L=1937,5MM	30			5	2580.00		
						MC MECCANO PROFIL L=2437,0MM	40			4	2910.00		
						MP PANEL MIDIPLUS/3 300/150	50			5	16250.00		
19	Metrostav	Stavba věže	Liberec	Petr Novotný	Brno	MC MECCANO PROFIL L=1937,5MM	60	11. 04. 2015	17. 04. 2015	20	2580.00	Výdej	Příjem
						MP PANEL MIDIPLUS/3 300/150	70	22. 03. 2015			16250.00		
						MD PANEL MIDI 300/50	100				9530.00		
						MC MECCANO PROFIL L=1875,0MM	50				2240.00		

Exportovat do Excelu

Prodáno

Filtrovat

Kontrakt	Klient	Název stavby	Město	Prodejce	Sklad	Artikl	Počet	Vytvořeno	Cena (Kč)	
32	Mosty Ústí n. L.	Rekonstrukce mostu	Ústí nad Labem	Petr Novotný	Benátky nad Jizerou	MP PANEL MIDIPLUS/3 300/150	20	11. 04. 2015	16250.00	Výdej
						MD PANEL MIDI 300/50	50		9530.00	
						MA PANEL MIDIALU 300/75	22		51.13	
1	Metrostav a.s.	sanace mostu Argentinská	Mladá Boleslav	Evžen Kozánek	Benátky nad Jizerou	MP PANEL MIDIPLUS/3 300/150	20	03. 01. 2015	16250.00	Výdej

Exportovat do Excelu

Obrázek 8: Seznam kontraktů

6.7.2 Výdej a příjem zboží

V této části kontraktu lze již zboží expedovat k zákazníkovi. Na této stránce se nachází tabulka s přehledem celého kontraktu, spolu s informacemi o tom, kolik zboží zbývá zákazníkovi dodat, případně kolik ho má zákazník vrátit.

U výdeje obsahuje první sloupec tabulky pole s předvyplněnými hodnotami o tom, kolik artiklů zbývá zákazníkovi dodat. Běžně se stává, že zákazník si nemůže všechno zboží odvézt okamžitě, proto je potřeba případně kolonku přepsat hodnotou o tom, kolik zboží si zákazník tedy odvezl. Z tohoto důvodu se v databázi uchovávají sloupce hlídající celkový počet artiklů v kontraktu, kolik artiklů zbývá dodat a kolik artiklů již zákazník má. Celá objednávka tedy nemusí být vydána najednou, ale postupně.

S tím také souvisí generování dokladů o výdeji, či příjmu. Po každém výdeji nebo přijmu je nutné vygenerovat doklad potvrzující předání zboží. Doklady jsou ukládány na server a po jeho vytvoření dojde také pod objednávkou k vytvoření odkazu ke stažení. Vzhledem k tomu, že si zákazník zboží může odvézt po částech, je potřeba pro každé předání mít uložený doklad o převzetí.

Tabulka výdeje i příjmu je zabalena v tagu *<form>*, který má *action* nastavený na controller s názvem *details* a v závislosti na tom, jestli se jedná o výdej nebo příjem, se zavolá stejnojmenná metoda *vydej_order* nebo *prijem_order*. Kliknutím na tlačítko „Odeslat“ dojde k potvrzení formuláře a odeslání dat metodou *post* do controlleru.

V controlleru nejprve ověřím, zdali uživatel všechna pole vyplnil a žádná nezůstala prázdná. Pokud ne, tak dojde k vytvoření session s identifikátorem *emptyValuesError* a k přesměrování zpět na přehled objednávky, kde se zobrazí varování s červeným textem, informující uživatele o problému.

Následně *for* cyklem dojde k projetí celého pole odeslaných dat. V každé iteraci cyklu dojde v databázové entitě *orders* k označení zvolených artiklů jako zapůjčených a v dané objednávce k odečtení celkového počtu kusů, které se mají ještě dodat. Dále je potřeba zjistit, z jakého skladu dojde k odeslání artiklů. Pomocí id objednávky dojde k SQL dotazu, ze kterého si uložíme id položky, id skladu a typ výdeje (prodej / pronájem). Pokud artikl pronajímáme, v příslušném skladu dojde k přesunutí dostupných produktů do zapůjčených. V opačném případě dojde k pouhému odečtení položky ze skladu. Pokud se zboží přijímá od zákazníka, tak dojde k navrácení zboží ze sloupce „zapůjčeno“ do sloupce „K pronájmu“. Tím vyřízení objednávky končí a je potřeba už pouze vytvořit doklad.

Pro vytváření PDF dokumentů byla použita PHP knihovna FPDF [7]. F v této zkratce znamená Free, což znamená, že knihovnu lze použít a libovolně modifikovat bez jakýchkoliv omezení. FPDF umožňuje formátování stránek, práci s hlavičkami a patičkami, vkládání obrázků, barev, tabulek atd. Knihovna není závislá na žádných dalších modulech a funguje v PHP ve verzi 4 a vyšší. Ve své práci jsem použil modifikovanou verzi této knihovny, zvanou tFPDF, která přidává podporu UTF-8 kódování.

Pro použití bylo potřeba pouze na stránce index.php zadat cestu k této třídě. Jelikož má tato knihovna problém s používáním českých fontů a při jejím použití zobrazuje na stránce varování, bylo potřeba po dobu využívání tFPDF vypnout hlášení o chybách, protože poté nastával problém s odesíláním požadavků o přesměrování do hlavičky:

```
error_reporting(E_ALL);
ini_set("display_errors", 0);
```

Následně dojde k vytvoření instance, na kterou se poté volají jednotlivé vykreslovací funkce. Nejprve se vytvoří prázdná stránka, dojde k nastavení fontu na *DeJaVuSansCondensed* o velikost číslo 12. Vkládání prvků do stránky se dělá obdobně jako u tabulek, čili volí se, do jakých buněk o dané velikosti chceme prvek vložit. Do první buňky se vloží unikátní číslo vyřizovaného kontraktu, do druhé název stavby, a do třetí jméno klienta.

Poté je potřeba vložit do dokumentu tabulku s přehledem předaného zboží. Pro správné zobrazení si z databáze vypíši tabulku přesně ve tvaru, v jakém chci tabulku také vykreslit v PDF dokumentu. Tabulka je ve své podstatě dvoj-

rozměrné pole, které postupně naplňujeme hodnotami. Nejdříve si vytvoříme hlavičku tabulky. Do každé části vkládáme asociované pole, kde klíčem jsou identifikátory FPDF, jako je *text* pro nadpis, který bude v hlavičce, *width, height* pro šířku a výšku, *align* pro zarovnání textu, *font_name* pro název použitého písma, *font_size* pro jeho velikost a případné atributy určující například barvu textu, či pozadí buňky.

Tělo tabulky je již dynamicky generováno podle počtu údajů, které nám vrátil dotaz z databáze. Vyplňování těla je syntakticky stejné jako u hlavičky s tím rozdílem, že místo statického textu se používají proměnné, do kterých jsme uložili výsledky z databáze. V tabulce je v prvním sloupci id položky, v druhém počet kusů, poté název produktu a jeho váha. Poslední sloupec se liší v závislosti na tom, jestli vykreslujeme výdejku nebo příjemku. Příjemka má poslední dva sloupečky prázdné, výdejka v nich zobrazuje cenu zboží, do které je při pronájmu již započítáno i procento prodejní ceny a v posledním je i vynásobeno počtem kusů. Na konci těla této tabulky je poslední buňka, ve které se nachází celková cena za výdej. Celková cena je v každé iteraci připočítávána do celkové sumy. Nakonec se toto dvojrozměrné pole odešle do funkce *WriteTable(array)*, které se postará o vykreslení tabulky. Níže lze vidět právě popsaný postup pro vytvoření těla tabulky nacházející se uvnitř cyklu (Tento postup se opakuje podle počtu buněk v těle):

```
   $col = array();
   $col[] = array('text' => 'Kod poloz.', 'width' => '15',
'height' => '5', 'align' => 'C', 'font_name' => 'DejaVu',
'font_size' => '8', 'font_style' => '', 'fillcolor' =>
'135,206,250', 'textcolor' => '0,0,0', 'drawcolor' => '0,0,0',
'linewidth' => '0.4', 'linearea' => 'LTBR');
   …
   $col[] = array('text' => 'Mnozstvi', 'width' => '15',
'height' => '5', 'align' => 'C', 'font_name' => 'DejaVu',
'font_size' => '8', 'font_style' => '', 'fillcolor' =>
'135,206,250', 'textcolor' => '0,0,0', 'drawcolor' => '0,0,0',
'linewidth' => '0.4', 'linearea' => 'LTBR');
   $columns[] = $col;
```

Pod tabulkou jsou ještě vykreslena pole s aktuálním datem a prázdným polem pro razítko skladu a podpis klienta.

Výsledný dokument je potřeba uložit na server. Souboru je potřeba vytvořit název, který je unikátní, aby nedocházelo ke kolizím, je potřeba rozlišit, jestli se jedná o výdejku nebo příjemku, a také brát na vědomí, že těchto souborů můžeme vytvořit větší množství v závislosti na tom, kolikrát si zákazník pro objednávku přijel, než došlo ke kompletnímu vydání. Pro výdejku se tedy soubor ukládá ve formátu *vydej[č. kontraktu]_[č. souboru].pdf.* U příjemek se liší pouze první část řetězce na *prijem*. Nejprve dojde k získání posledního volného čísla souboru z databáze. To se vloží do řetězce s názvem souboru, poté se inkrementuje a pošle zpět do databáze. Číslo souboru je tak připraveno na příští použití. K nahrání dokumentu na server se použije funkce *Output*. Ta jako první parametr přejímá název souboru a jako druhý způsob odeslání souboru.

V našem případě je druhý parametr *F*, což značí odeslání souboru s pevně daným jménem na server do složky *receipt*:

```
$num = $this->model->getFileNum($id_contract);

$filename="vydej".$id_contract."_".$num['file'].".pdf";
$this->model->setFileNum($id_contract);

$pdf->Output("receipt/".$filename,'F');
```

Nakonec dojde k přesměrování uživatele zpět do pohledu detailů dané objednávky. Pod tabulkou je potřeba vypsat seznam odkazů na právě nahrané soubory. Je potřeba však vypsat buď pouze výdejky, nebo příjemky v závislosti na tom, v jaké části objednávky se nacházíme. K tomu byla použita funkce *glob()*, která vyhledává soubory podle nějakého vzoru. V tomto případě je vzor celý název souboru, kde pouze číslo souboru je nahrazeno hvězdičkou, což značí, že se místo ní může nacházet jakýkoliv znak. Odkaz na stáhnutí je v tagu <a href>, s atributem *download,* čímž se prohlížeči řekne, že se nemá přesměrovávat, ale soubor stáhnout:

```
<?php foreach
(glob("receipt/vydej".$this-
>contractInfo[0]['id_contract']."*.pdf") as $key => $value):
?>
  <div class="vydejky">
     <a href="../../<?=$value?>" downlo-
ad="vydejka.pdf">Stáhnout výdejku č. <?=$key?></a>
  </div>
<?php endforeach; ?>
```

Obrázek 9: Výdej kontraktu

7. Testování aplikace

Aplikace byla celá psána a testována na lokálním počítači (*localhost*). S testováním funkčnosti samotných webových stránek by nebyl žádný problém, nicméně pro dynamické vykreslování obsahu je využit programovací jazyk PHP. PHP skripty jsou prováděny na straně serveru a k uživateli je přenášen až výsledek jejich činnosti [14]. A stejně tak pomocí PHP přistupujeme k databázi, takže bylo potřeba mít na lokálním počítači nainstalován nějaký WAMP balíček.

WAMP je zkratkou pro Windows, Apache, MySQL a PHP, což je jak název vypovídá, balíček určený pro systém Windows, obsahující softwarový webový server Apache pro práci s PHP, databázový systém MySQL a samotný PHP balíček.

Pro práci bylo vybráno prostředí s názvem EasyPHP, které výše zmiňovaný balíček implementuje v nejnovějších verzích a má jednoduché ovládání. Pro testování aplikace jej pak stačí zapnout a ono nastartuje potřebné servery, ke kterým přes adresu „127.0.0.1“ získáme přístup k naší webové aplikaci.

Pro přístup k databázi má EasyPHP implementováno prostředí s názvem *phpMyAdmin*. Jedná se o nástroj napsaný v jazyce PHP, umožňující jednoduchou správu obsahu databáze MySQL prostřednictvím webového rozhraní. Umožňuje vytvářet/rušit databáze, vytvářet/upravovat/rušit tabulky, provádět SQL příkazy a spravovat klíče [15].

Na ostrý provoz byl použit server www.php5.cz. Jedná se o freehosting a je určen primárně pro programátory a vývojáře, kteří si chtějí své projekty v ostrém provozu vyzkoušet. Přístup k serveru je získán prostřednictvím FTP a hlavní výhodou je, že využívá stejné technologie jako EasyPHP. Není tedy žádný problém s přechodem z localhostu na freehosting. Navíc jsou PHP a MySQL pravidelně aktualizovány.

Jediný technický problém nastal u FPDF modulu, který generuje výdejky a příjemky. Při generování PDF dokumentu měl server problém s hledáním cesty k fontu *DejaVuSansCondensed.ttf*, i když byl správně nahrán. Bylo tedy potřeba vlastní font odstranit a použít vestavěný font *Arial*. Poté již proběhlo vše bez problému.

V původním návrhu aplikace generování výdejek a příjemek zcela chybělo a na její absenci bylo poukázáno, až při prvním testování. Zboží se tedy odečítalo ze skladu v momentě, kdy došlo k vytvoření objednávky. To byla samozřejmě chyba, protože došlo k vynechání kroku, kdy si zákazník musí pro zboží teprve přijet. Aplikace se tedy rozšířila o možnost rozkliknout příslušnou objednávku a zboží buď vydat, nebo přijmout.

Nejvíce změn při testování nastalo v uživatelském prostředí. Jedním požadavkem například bylo, aby co nejvíce vstupních polí ve formulářích bylo předem vyplněno. Například při vydávání zboží se do vstupních polí zadává, kolik zboží bylo zákazníkovi vydáno. V původním návrhu se všechna čísla psala ručně. Nyní jsou předvyplněná údaji z databáze a pouze pokud čísla nesouhlasí, dojde k jejich změně. Díky tomu je menší šance, že uživatel při vydávání zboží udělá chybu.

Další změna proběhla u formuláře na převod artiklů mezi sklady. Ten obsahuje dvě roletkové menu. V prvním se vybírá, z jakého skladu si přejeme převod zahájit a v druhém kam ho chceme přesunout. Původně šlo vybrat v obou polích stejný sklad, což se samozřejmě nesmí stát. Proto byl tento formulář do-

plněn o skript, který hlídá, co bylo v jednom poli vybráno a v druhém tu samou položku znemožní vybrat.

Aplikace byla interně testována pár dobrovolníky, kteří obdobné aplikace využívají a ve stavebním průmyslu pracují.

8. Závěr

Úspěšně byla vytvořena aplikace, která umožňuje správu produktů jednotlivých skladů. Do skladů lze libovolně přidávat nové artikly, upravovat jejich stav a přesouvat je mezi sklady. Lze také vytvářet objednávky a při jejich následném výdeji nebo příjmu dochází k ukládání příslušných dokladů. Jedním kliknutím lze aplikaci okamžitě přepínat mezi dalšími jazyky a případná další lokalizace nevyžaduje žádné další znalosti programování nebo žádná specifická editační prostředí.

Uživatel má v závislosti na oprávnění přístup pouze k určitým částem systému, takže nemůže měnit něco, na co nemá pravomoci. Zároveň díky tomu pro něj vzniká uživatelsky přívětivější prostředí, neboť na něj nepůsobí žádné rušivé elementy.

Tento systém je pro firmy taktéž výhodnější z ekonomických důvodů, kdy odpadá problém s licencováním a do prostředí se zaměstnanec může přihlásit prakticky odkudkoliv, bez nutnosti instalací dalších doplňků.

Díky architektuře MVC, ve které je práce napsána ji lze jednoduše dále rozšiřovat. V budoucnu by se mohla do aplikace například přidat sekce s přehledem zákazníků. Firma by tak mohla mít kompletní přehled o uskutečněných obchodech jejich konkrétního zákazníka, případně by se mohl implementovat systém věrnostních slev. Další funkcí by mohl být specifičtější přehled artiklů, které jsou v opravně (v jaké opravně, jak je poškozeno atd.). Zajímavostí by mohlo být i přidání fotek k jednotlivým artiklům. Při větším rozšíření by se aplikace mohla přeložit do více jazyků.

Pro vytvoření práce byly použity technologie HTML5, CSS3, PHP5, MySQL, AJAX a Javascript.

POUŽITÁ LITERATURA A ZDROJE

[1] Comsys-Software. *Comsys-Software.* [online]. [2015] [cit. 2015-05-10]. Dostupné z: *http://www.comsys-sw.cz/Sklad.aspx*

[2] Shoptronic – obchodní skladový systém. *Shoptronic.* [online]. [2015] [cit. 2015-05-10]. Dostupné z: *http://www.omega-lbc.cz/index.php/produkty/shoptronic*

[3] Sklad a fakturace MAXim | MJsoft. *MJsoft.* [online]. [2015] [cit. 2015-05-10]. Dostupné z: *http://www.mjsoft.cz/skladovy-obchodni-system*

[4] Bohumil Jahoda. PDO. *Je čas.* [online]. 9.3.2014 [cit. 2015-05-03]. Dostupné z: *http://jecas.cz/pdo*

[5] Christian Bach. jQuery plugin: Tablesorter 2.0. *jQuery plugin: Tablesorter 2.0.* [online]. [2014] [cit. 2015-05-02]. Dostupné z: *http://tablesorter.com/docs/#Download*

[6] Chris Coyier. Light Javascript Table Filter. *Codepen.* [online]. 19.9.2013 [cit. 2015-05-02]. Dostupné z: *http://codepen.io/chriscoyier/pen/tIuBL*

[7] FPDF. *FPDF.* [online]. 18.6.2011 [cit. 2015-05-02]. Dostupné z: *http://www.fpdf.org/*

[8] Tomas Kirda. jQuery autocomplete. *Devbridge Sourcery.* [online]. [2015] [cit. 2015-05-02]. Dostupné z: *https://www.devbridge.com/sourcery/components/jquery-autocomplete/*

[9] GILMORE, W. Velká kniha PHP 5 a MySQL: kompendium znalostí pro začátečníky i profesionály. Nové, 3. vyd. Překlad Jan Pokorný. Brno: Zoner Press, 2011, 736 s. Encyklopedie Zoner Press. ISBN 978-80-7413-163-9.

[10] ZAKAS, Nicholas C, Jeremy PCPEAK a Joe FAWCETT. Ajax: profesionálně. Vyd. 1. Překlad Jiří Koutný. Brno: Zoner Press, 2007, 668 s. ISBN 978-80-86815-77-0.

[11] DUCKETT, Jon. Javascript: interactive front-end web development. 1. vyd. Indianapolis, IN: John Wiley, 2013, 640 s. cm. ISBN 11-185-3164-7.

[12] MD5 password scrambler ’no longer safe’. *ZDNet.* [online]. 7.6.2012 [cit. 2015-05-04]. Dostupné z: *http://www.zdnet.com/article/md5-password-scrambler-no-longer-safe/*

[13] Safe Password Hashing. *PHP.net.* [online]. [2014] [cit. 2015-05-04]. Dostupné z: *http://php.net/manual/en/faq.passwords.php*

[14] PHP. *Wikipedie: Otevřená encyklopedie.* [online]. 17.4.2015 [cit. 2015-05-08]. Dostupné z: *http://cs.wikipedia.org/wiki/PHP*

[15] PhpMyAdmin. *Wikipedie: Otevřená encyklopedie.* [online]. 3.1.2015 [cit. 2015-05-08]. Dostupné z: *http://cs.wikipedia.org/wiki/PhpMyAdmin*