UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

# BAKALÁŘSKÁ PRÁCE

2017 Tomáš Tichý

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

# Webová aplikace pro správu skladového systému

Tomáš Tichý

Bakalářská práce

2017

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2016/2017**

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

| | |
|---|---|
| Jméno a příjmení: | **Tomáš Tichý** |
| Osobní číslo: | **I14189** |
| Studijní program: | **B2646 Informační technologie** |
| Studijní obor: | **Informační technologie** |
| Název tématu: | **Webová aplikace pro správu skladového systému** |
| Zadávající katedra: | **Katedra informačních technologií** |

Zásady pro vypracování:

Cílem bakalářské práce je navrhnout a vytvořit webovou aplikaci pro správu skladového systému.

V textu bakalářské práce bude provedeno srovnání aktuálně dostupných systémů podobného typu a uvedení jejich výhod a nevýhod.

Pro ukládání dat o skladových položkách bude použit databázový systém Oracle. Webová aplikace bude vytvořena pomocí technologií PHP, HTML, kaskádových stylů a JavaScriptu.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

**1. NEUSTADT I. , ARLOW J. UML 2 a unifikovaný proces vývoje aplikací. Computer Press 2007, 568 stran. ISBN 9788025115039.**

**2. ÖGGL B., KOFLER M. PHP 5 a MySQL 5 - Průvodce webového programátora. Computer Press 2007, 608 stran. ISBN 9788025118139.**

Vedoucí bakalářské práce: **Ing. Michael Bažant, Ph.D.**
Katedra softwarových technologií

Datum zadání bakalářské práce: **31. října 2016**

Termín odevzdání bakalářské práce: **12. května 2017**

Univerzita Pardubice Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Mgr. Josef Horálek, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2017

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 05. 05. 2017

Tomáš Tichý

## PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu své práce Ing. Michealu Bažantovi, Ph.D. za cenné rady a velmi vstřícný přístup. Dále bych rád poděkoval své rodině a blízkým přátelům, především pak své sestře, Šárce Tiché, za velkou podporu a motivaci.

## ANOTACE

Teoretická část práce je zaměřená na popis a porovnání technologií pro vývoj webových aplikací a na popis návrhu a postupu realizace samotné aplikace. Praktická část obsahuje analýzu aplikace, implementaci a technickou dokumentaci s popisem všech funkcí.

## KLÍČOVÁ SLOVA

Webová aplikace, databáze, sklad, skladový systém, PHP, HTML, JavaScript, CSS

## TITLE

Web application for warehouse system management

## ANNOTATION

Theoretical part is focused on description and comparison of web application development technologies and on the planning and realization of the application itself. Practical part contains application analysis, implementation and technical documentation including all the function description.

## KEYWORDS

Web application, database, warehouse, warehouse system, PHP, HTML, JavaScript, CSS

# OBSAH

# SEZNAM ILUSTRACÍ A TABULEK

# SEZNAM ZKRATEK A ZNAČEK

| | |
|---|---|
| ACID | Atomicity Consistency Isolation Durability |
| API | Application Programming Interface |
| CSS | Cascading Style Sheets |
| DTD | Document Type Definition |
| ERP | Enterprise Resource Planning |
| HSLA | Hue Saturation Lightness Alpha |
| HTML | HyperText Markup Language |
| JEE | Java Enterprise Edition |
| MVC | Model-View-Controller |
| MVCC | Multi-Version Concurrency Control |
| MySQL | My Structured Query Language |
| PHP | HyperText Preprocessor |
| PL/SQL | Procedural Language/Structured Query Language |
| RGBA | Red Green Blue Alpha |
| SMTP | Simple Mail Transfer Protocol |
| SQL | Structured Query Language |
| UML | Unified Modeling Language |
| WYSIWYG | What You See Is What You Get |

# ÚVOD

Tato bakalářská práce se zabývá návrhem a tvorbou webové aplikace pro správu skladového systému. V současné době je problém nalézt aplikaci, která by byla multiplatformní, uživatelsky přívětivá a obsahovala všechny funkce, spojené se správou skladu. Zároveň správa skladu není úplně jednoduchá problematika, proto součástí této práce bude i nastínění hlavních problémů, spojených se správou skladů.

Aplikace pro správu skladového systému bude vždy postavena na databázi pro uložení veškerých dat. Pro návrh a tvorbu databáze, stejně jako pro vývoj webových aplikací ale existuje větší množství technologií a metod. Z tohoto důvodu bude teoretická část této práce zaměřená na popis a porovnání technologií, jak pro vývoj webových aplikací, tak pro návrh a tvorbu databáze. Ve shrnutí každé z těchto dvou částí bude provedena komparace všech zmíněných technologií.

V další části této práce je obsažena analýza trhu, která je zaměřena na nejvýznamnější konkurenty v odvětví, nejen webových, aplikací pro správu skladových systémů a provede srovnání s vlastní aplikací pro správu skladového systému.

Neodmyslitelnou částí vývoje a sestavení jakéhokoli software, je určitě bezpochyby a návrh aplikace. Tvorba výše zmíněné aplikace není otázkou pouhých pár hodin samotné implementace vybraných modelů, je nutné sestavit analytické a návrhové modely, které pomohou nastínit problematiku a způsob implementace řešení. Jednou ze stěžejních částí práce tedy je právě popis fází analýzy a návrhu včetně diagramů.

Hlavním cílem této práce je samotná realizace a implementace webové aplikace pro správu skladového systému. Autor v prvních kapitolách uvádí technologie, které si pro realizaci této aplikace vybral, následně je provedena analýza a návrh řešení a na závěr je popsán způsob implementace a hlavní funkce aplikace. Součástí práce je rovněž popis postupu práce v aplikaci a popis oprávnění pro jednotlivé uživatelské role.

# 1 UVEDENÍ DO PROBLEMATIKY

Tato kapitola se bude zabývat definicí skladu, uvedením do problematiky oceňování zboží a definicí webové aplikace.

V současné době existuje mnoho průmyslových firem, které potřebují uchovávat buď materiály na výrobu, nebo už hotové výrobky před dalším zpracováním, či prodejem. Čím větší množství zásob má firma na skladě, tím větší množství peněžních prostředků je v nich drženo. S růstem množství zásob na skladě vzrůstají náklady, které musí firma vynaložit na jejich uskladňování (pronájem prostor, obaly, pracovníci atd.). Z tohoto důvodu je třeba co možná nejlepším způsobem sledovat veškeré pohyby na skladě, aby nedocházelo k manku (stav, kdy skutečný stav zásob je nižší, než účetní).

## 1.1 Sklad

Sklad lze chápat dvěma způsoby. Prvním způsobem je reálný sklad – prostor, určený ke skladování zásob (surovin, výrobků, zboží, známek, stravenek), kde jsou tyto zásoby přímo fyzicky uloženy, za účelem uchování v neměnném stavu. Druhým způsobem chápaní tohoto pojmu, je program, ve kterém jsou zachyceny informace o pohybu zásob na fyzickém skladě (včetně finanční hodnoty). Zůstatky na obou skladech by se měly rovnat, v opačném případě by docházelo k již zmíněnému manku, nebo přebytku, který by musel být řešen. Za fyzický sklad odpovídá zaměstnanec, který má hmotnou odpovědnost za zásoby, a v případě manka mu zaměstnavatel může dát dané manko k úhradě.

Firma může mít vlastní skladové prostory, nebo si může pronajímat skladovací prostory. Obě tyto možnosti však počítají s nemalými náklady na skladování a dopravu zásob do skladů. Pro minimalizaci těchto nákladů lze zmínit japonskou metodu „Just in time“, která slouží právě k již zmíněné minimalizaci nákladů na dopravu a skladování zásob. Slouží k zajištění dodávek potřebných zásob přesně ve stanovený čas potřeby. Tuto metodu používá například kolínská automobilka TPCA, která má všechny důležité dodavatele umístěné přímo v areálu automobilky, tudíž jsou náklady na dopravu i skladování zásob opravdu minimální a jejich doprava nemůže být zdržena po cestě [18].

## 1.2 Oceňování zásob

Do skladové evidence se kromě ostatních údajů (výrobce, barva, typ, atp.) uvádí i pořizovací cena, která se skládá z ceny pořízení (fakturační cena, clo, spotřební daň), vedlejších pořizovacích nákladů (přepravné, skladovací poplatky), případně i vnitropodnikových služeb

(vlastní dopravné). Úbytky zásob se uskutečňují v evidovaných pořizovacích cenách. Nákupní ceny stejného produktu se však v průběhu času mění, proto dochází ke změně ocenění. Tyto změny lze v účetnictví postihnout následujícími metodami [2]:

- **Vážený aritmetický průměr** – při každém výdeji se vydávané množství ocení přepočítanou průměrnou cenou zásob na měrnou jednotku.
- **FIFO – First In First Out** – při výdeji se zásoby oceňují cenami od nejstarší pořizovací ceny zásoby k nejnovější.
- **LIFO – Last In First Out –** při výdeji se zásoby oceňují cenami od nejnovější pořizovací ceny zásoby k nejstarší, u nás je tato metoda zakázána [35].

Pro lepší představu následuje příklad ocenění zásob při vyskladňování za použití různých metod. Naskladníme 30 kg materiálu za 11 Kč/kg a 20 kg materiálu za 9 Kč/kg. Budeme chtít vyskladnit 40 kg materiálu.

**Tabulka 1 - Porovnání metod oceňování zboží**

| Oceňovací metoda | Výpočet ceny za kus | Celková cena za 40 kg |
|---|---|---|
| **Vážený arit. průměr** | $\frac{30 * 11 + 20 * 9}{30 + 20}$ | 408 Kč |
| **First In First Out** | Dle pořadí přijetí zboží | 420 Kč |
| **Last In First Out** | Dle pořadí přijetí zboží | 400 Kč |

## 1.3 Webová aplikace

Webová aplikace je webové místo, které má buď jen částečně určený, nebo úplně neurčený obsah. Konečný obsah stránky se formuluje až po odeslání uživatelského požadavku na server, neboť pro každý požadavek může být obsah stránky úplně odlišný. Jedná se v podstatě o kolekci statických a dynamických webových stránek, kde statická stránka zůstává neměnná a pouze se na požadavek posílá prohlížeči. Na druhou stranu stránka dynamická je modifikována serverem před odesláním klientovi. Webové aplikace jsou velmi rozšířené, především z důvodu všudypřítomného a platformě nezávislého webového prohlížeče (klienta). Aplikace může mít například následující využití [16]:

- Snadné a rychlé hledání informací (prohledávání a organizace).
- Shromažďování, ukládání a analýza dat.
- Aktualizace webových míst s proměnlivým obsahem [16].

# 2 TECHNOLOGIE PRO TVORBU WEBOVÉ APLIKACE

Obsah této kapitoly bude zaměřen na aktuálně nejpoužívanější technologie pro návrh a vývoj webových aplikací a bude rozdělen na dvě části, kde první bude technologie, použitá v této práci pro tvorbu webové aplikace pro správu skladového systému, a druhá bude popisovat alternativní možnosti, které lze použít ke stejnému účelu.

## 2.1 Použitá technologie

### 2.1.1 HTML

HTML je zkratka pro Hypertext markup language, což je značkovací „programovací" jazyk pro tvorbu statických webových stránek. Slovo HyperText označuje možnost propojení textů pomocí odkazů, Markup označuje možnost dávat jednotlivým blokům textu specifické významy pomocí speciálních značek (tagů, elementů) jako například zvýraznění textu, nebo označení části textu jako nadpis. Jazyk vznikl z původního univerzálního jazyka SGML (Standard Generalized Markup Language) v roce 1990 [15].

Jedná se o značkovací jazyk, ve kterém se místo příkazů využívají tagy (znaky), takže není nutný překlad do strojového kódu. HTML soubor je složen z HTML tagů a je možné jej vytvořit v obyčejném textovém editoru jako soubor s koncovkou html nebo htm. K tagům se většinou připojuje výpis vlastností (atributů) a mezi tagy (uzavřené v úhlových závorkách) se umisťují samotné části textu. Samozřejmě zda platí určitá pravidla a možnosti, jak tagy a text do tagů zapisovat. Tagy mohou být párové (* text odstavce *) nebo jednodílné (*řádek <br />*), ale nejdůležitější je, že v případě párových tagů se tagy nesmí křížit (* <a> text </a>*) [3].

Aktuální verze je HTML5 (ve fázi vývoje již je verze 5.1, ta však ještě nemá své funkční sestavení), které oproti předchozím verzím doznalo pár vylepšení, mezi které patří například:

- V prologu není nutné uvádět verzi a odkaz na DTD (Document Type Definition).
- Element <a> může zabalovat jakékoli (i blokové) elementy [28].

Webovou stránku (soubor HTML) lze vytvořit v různých editorech, které se dají rozdělit na dvě skupiny. První skupinou jsou klasické textové editory, do kterých přímo píšeme HTML tagy a jejich obsah, a z nichž většina dokáže barevnou syntaxi (barevné odlišení tagů a obsahu tagu), intuitivní napovídání značek a v některých případech i validaci. Mezi takové textové editory patří například Brackets, PSPad nebo Notepad++.

Druhou možností tvorby webových stránek je použití WYSIWYG (What You See Is What You Get – co vidíš, to dostaneš) editoru, ve kterých přímo upravujeme stránku tak, jak bude vypadat v prohlížeči. Výhodou tohoto způsobu tvorby je bezesporu fakt, že na práci s takovým editorem není potřebná znalost jazyka HTML. Nevýhodou však je generování většinou sémanticky nebo strukturálně špatného kódu, což může vést až ke snížení přístupnosti webu. Některé WYSIWYG editory mají navíc problémy s kaskádovými styly (viz níže). K nejznámějším WYSIWYG editorům patří například Microsoft FrontPage nebo NVU [32].

### 2.1.2 CSS

CSS je zkratka pro Cascading Style Sheets (kaskádové styly) a označuje kolekci metod pro grafickou úpravu webových stránek. Kaskádové se jim říká proto, že na sebe mohou vrstvit definice stylu, ale platí pouze ta poslední. Kromě obsahu webových stránek je určitě podstatný i formát (forma), který upravuje například barvu, velikost písma, zarovnání atd. V dnešní době se již většina formátování provádí v pomocí CSS (snad kromě ztučnění a kurzívy). CSS se vyplatí především na následující [9]:

- Hezké a moderní formátování stránky (barvy, zarovnání atd.).
- Časté psaní textů na internet v jednotném formátování.
- Práce s JavaScriptem.
- Správa mnoha stránek s jednotným formátováním [9].

CSS zvládá například následující formátování:

- Přesná velikost písma, prokládání, kapitálky.
- Odsazení prvního řádku odstavce, zvětšit řádkování.
- Automatické formátování nadpisů.
- Předefinovat grafický význam běžných tagů.
- Nastavení pozice jakéhokoli elementu [9].

Styly lze deklarovat třemi různými způsoby, z nichž každý má jisté výhody a nevýhody. Prvním způsobem je deklarace přímo v textu zdroje u formátovaného elementu pomocí atributu style="…". Tomu se říká přímý styl a používá se výjimečně, neboť je obtížné takové formátování spravovat, zejména pokud se vyskytuje opakovaně [9].

```
<p style="color: red;"> odstavec </p>
```

Druhou možností je deklarace pomocí „stylopisu" (stylesheet) v hlavičce stránky. Jedná se o jakýsi seznam stylů, ve kterém je obecně napsáno, co má jak být zformátováno. Do stránky se stylopis píše mezi tagy <style> a </style> a používá se například pro rozsáhlejší styly na

první stránce (indexu), aby se zvýšila rychlost a efektivita, a nemuselo se čekat na stáhnutí externího souboru .css (viz níže) [9].

```
<style>
p {color:red;}
</style>
<p>odstavec</p>
```

Poslední možností je deklarace pomocí externího stylopisu (souboru .css), na který stránka odkazuje pomocí tagu <link>. Největší výhodou tohoto způsobu deklarace je možnost nalinkování na stejný soubor, obsahující stylopis, z většího množství stránek. Pomocí tohoto se dá docílit stejného vzhledu na několika stránkách, které budou využívat stejné tagy [9].

```
p {color: red;}
<link rel="stylesheet" type="text/css" href="styly.css">
```

CSS nejsou součástí HTML, proto mají odlišnou syntaxi zápisu. Ať už se jedná o kteroukoli ze třech možností deklarace, pro každý element má zápis stejný návrh – vlastnost: hodnota;…. Je však nutné dát si pozor, kde se používají uvozovky, dvojtečky, složené závorky, středníky a čárky. Komentáře se ve stylopisech (podobně jako v Javě) zapisují mezi znaky /* a */ [9].

Aktuální verze je CSS3, které kromě stávajících vlastností zvládá například:

- Kulaté rohy (border radius) s možností stejně, nebo různě zaoblených rohů.
- Stín pro boxy i text (box-shadow, text-shadow).
- Různé modely barev (RGBA – Red Green Blue Alpha, HSL, HSLA – Hue Saturation Lightness Alpha) kde alpha označuje neprůhlednost [27].

### 2.1.3 PHP

PHP je skriptovací jazyk pro tvorbu dynamického webu, který byl vytvořen kolem roku 1994. Zkratka PHP označuje HyperText Preprocessor, přestože v počátcích se překládala jinak (dříve například Personal Home Page). PHP si v poslední době získal na oblibě, především díky některým svým vlastnostem [33]:

- Snadná pochopitelnost.
- Syntaxe podobná jazyku C.
- Snadná komunikace s Apache serverem i MySQL databází.
- Multiplatformita [33].

Přes všechny zmíněné výhody se ale najdou i nevýhody, které by někoho mohly odradit. Mezi takové patří například fakt, že se jedná o interpretovaný, ne kompilovaný jazyk, nebo nízká

úroveň podpory objektového programování. Stále se ale jedná o univerzální jazyk vhodný na velké množství různých webových aplikací (internetové obchody, podnikové informační systémy, firemní prezentace, dynamické osobní stránky, vyhledávače, katalogy atp.) [33].

Oproti začátkům, kdy stačily statické stránky, které byly zobrazeny ve stejné podobě, ve které byly napsány, dnes je snaha vytvořit stránky co možná nejvíce dynamické a trochu tyto stránky rozpohybovat. Za tímto účelem bylo vytvořeno hned několik technologií, které se dají rozdělit na klientské a serverové [34].

Klientské technologie spočívají v tom, že spolu s HTML stránkou je prohlížeči odeslán i kus programového kódu, který se ve správný okamžik spustí. Ke spuštění dojde například po kliknutí na tlačítko, najetím na určitý element stránky nebo otevřením nového okna. Spouštění programového (klientského) kódu má na starosti prohlížeč, což může být nevýhoda, neboť ne každý prohlížeč zná všechny programovací jazyky. Příklad klientské technologie je JavaScript, kterému se bude věnovat následující kapitola [34].

Serverové technologie fungují tak, že pokaždé, když prohlížeč odešle na server požadavek na stránku, server tuto stránku nejprve sestaví a až pak ji hotovou pošle prohlížeči. Sestavení stránky je přímo závislé na přijatém požadavku, proto server skoro pokaždé sestavuje jinou stránku. Jednou z těchto technologií je právě PHP. Typický PHP skript obsahuje kromě programového kódu i HTML kód. Při sestavování stránky server vezme HTML kód, provede programový (PHP) kód, zkombinuje obě části dohromady a výsledek odešle prohlížeči [34].

Aby mohl být PHP skript spuštěn, musí být odeslán na server. Po obyčejném otevření skriptu v textovém editoru není vidět výsledek, ale pouze skript samotný. Skripty nejsou závislé na prohlížeči, veškerou práci zde odvede server a prohlížeč následně pouze zobrazí výsledek [34].

Server, na který chceme publikovat HTML stránku s PHP skriptem musí podporovat PHP. Takový server je možné nainstalovat i doma (nejčastěji včetně podpory Apache – serveru, modulu PHP a MySQL - databáze). Veškeré nastavení PHP skriptů se provádí v souboru php.ini, kde se dají nastavovat direktivy proměnných, nebo například SMTP pro odesílání mailů [33].

Už od verze PHP 4 začalo přibývat objektově orientovaných funkcí a ve verzi PHP 5 došlo k zásadnímu přepracování objektově orientovaného programování. Syntaxe objektově orientovaného programování v PHP je velmi podobná s programovacím jazykem Java. Najdou

se zde však i podobnosti s jazykem C++, jako například destruktory nebo přetěžování metod (a proměnných) [23].

Aktuální verze je PHP7, která s sebou přináší řadu novinek, jako například:

- Celočíselné dělení (funkce intdiv()).
  ```
  <?php echo intdiv(7,3); ?>
  ```
- Trojcestný operátor <=> (podobné jako funkce strcmp() nebo version_compare()).
  ```
  <?php echo $a <=> $b; ?>
  ```
  výraz vrátí -1 pokud $a<$b, 0 pro $a=$b a 1 když $a>$b

- Operátor ?? (jako ternární operátor s funkci isset()).
  ```
  <?php $mycolour = $colour ?? „blue“; ?>
  ```
  Pokud proměnná $colour není definovaná, vrátí druhý operand („blue“)

- Deklarace návratových typů funkcí
  ```
  function add( $a, $b) : int {
     return $a + $b;
  }
  ```
  Funkce add deklaruje návratový typ int (pokud by vracela něco jiného, skončí skript chybou) [7].

### 2.1.4 JavaScript

JavaScript je skriptovací (programovací) jazyk, který se používá na tvorbu dynamických prvků na webových stránkách. Jedná se o typického zástupce klientské technologie, což znamená, že všechny aplikace jsou spouštěny v prohlížeči (u uživatele). V dnešní době je velmi používanou technologií k oživení webu. Mimo jiné umožňuje měnit obsah webové stránky u uživatele, vytvářet dynamická menu, roletky a kontejnery, formátovat text (včetně například vkládání smajlíků) nebo vytvářet nejrůznější ukazatele času a data [11].

Vzhledem k pomalému procesu odeslání stránky a čekání na odpověď serveru se využívá klientský JavaScript například i pro validaci webových formulářů. Jelikož se ale jedná o klientskou technologii, může si ji uživatel kdykoli vypnout nebo přepsat, takže se na to nedá spoléhat a je třeba provést i serverovou kontrolu. Velkou výhodou však zůstává pohodlnost a efektivita bez zbytečných prodlev [11].

Kvůli odlišnostem v prohlížečích bylo dlouhou dobu obtížné provést kompletní implementaci jazyka. To se však zlepšilo vznikem JavaScriptových frameworků, které poskytují unifikované

rozhraní a řeší odchylky v prohlížečích. Mezi takové frameworky patří například JQuery nebo MooTools, které obsahují tzv. widgety (připravené miniaplikace, například pro menu) [11].

Stejně jako PHP, i JavaScript je jazyk interpretovaný, neboli překládaný za běhu a vykonávaný dle svého zdrojového kódu. Rovněž je to jazyk objektově orientovaný, přestože zde neexistuje nic jako třída. Objekt je zde to samé, jako slovník – kontejner. JavaScript využívá tzv. funkcionální paradigma (do běžné proměnné lze uložit funkci), což se hodí například na tvorbu lambda výrazů. Dále zde funguje prototypová dědičnost, tedy objekt je předlohou jiného objektu [11].

Skript může buď být vložen přímo do webové stránky, nebo umístěn do externího souboru, na který uvedeme odkaz na webové stránce. Samostatné soubory psané v JavaScriptu mají koncovku .js nebo .jse. Skripty se dají psát v libovolném textovém editoru (PSPad, Notepad++, Brackets). K prohlížení je už však potřeba internetový prohlížeč. Důležité je, že JavaScript netoleruje záměnu velkých písmen za malé (je case-sensitive), takže document.write není to samé jako DOCUMENT.write. Pokud se rozhodneme vkládat skript přímo do webové stránky, uzavírá se tento skript do párového tagu <script></script>. V opačném případě přidáme jako atribut skriptu zdroj [17].

```
<script src="skript.js"></script>
```

## 2.2 Alternativní technologie

### 2.2.1 Frameworky

Přestože PHP patří mezi vyšší programovací jazyky a obsahuje spoustu funkcí, jeho standardní knihovny mají značné mezery. Kvůli těmto mezerám musíme některé funkčnosti stále dokola programovat. Tento problém se dá vyřešit například pomocí frameworků, které navíc veškerou práci urychlí. Framework je soubor na sebe navazujících a propojených knihoven, bez kterých se v PHP větší projekty dělají složitěji a mnohem déle. Můžeme si jej buď vytvořit, nebo využít již existující. Frameworky mimo jiné poskytují [20]:

- Rychlý vývoj.
- Dobře organizovaný, opětovně využitelný a udržovatelný kód.
- Vysokoúrovňovou bezpečnost webu.
- Separaci prezentace a logiky (díky MVC vzoru) [20].

### Nette framework

Nette framework je kvalitně objektově navržený, kompletní framework od českého vývojáře Davida Grudla. Mimo jiné na tomto frameworku fungují projekty jako například Moneta Money Bank nebo Slevomat. Pro používání Nette potřebujeme **Sandbox**, který tvoří kostru webové aplikace, na kterou budeme navazovat webové stránky. Nette je klasický MVC (model-view-controller) framework, konkrétněji MVP [1]:

- Modely – logika aplikace (práce s databází, výpočty), každá entita – vlastní model.
- Pohledy (V Nette Templates) – Latte (programovací jazyk pro vkládání dat z PHP pomocí speciálních značek do HTML) šablony s HTML kódem.
- Presentery – komponenta pro komunikaci s uživatelem (předání parametrů) -> předání dat modelům -> získání dalších dat -> předání pohledům (šablonám) -> začlenění do HTML kódu -> odeslání HTML uživateli do prohlížeče [1].

Když uživatel zadává požadavek, přejde nejdříve na router (směrovač), který podle URL zavolá příslušný presenter. Každý presenter má několik vlastních šablon pro specifické akce (detail, seznam). Související moduly a presentery se sdružují do modulů (balíčků komponent pro určitou část webu). Kromě MVC komponent se v Nette používají i tzv. Filtery (malé pomocné funkce pro šablony, sloužící k formátování výstupu) [1].

### Laraval

I přes svou krátkou existenci (od roku 2011) je podle průzkumu nejoblíbenějším frameworkem mezi vývojáři. Jedná se o obrovský ekosystém s platformou, připravenou k okamžitému hostování a rozmisťování. Laraval má mnoho schopností, díky kterým umožňuje rychlý vývoj aplikací. Zároveň má vlastní odlehčený šablonový engine („Blade“) a elegantní syntax pro usnadnění častých úloh (autentizace, relace, fronty, cachování atp.). Laraval obsahuje i vlastní vývojové prostředí Homestead [20].

### Symfony

Komponenty frameworku Symfony jsou opětovně využitelné knihovny PHP pro kompletaci nejrůznějších úloh (vytváření formulářů, konfiguraci objektů, směrování, autentizaci atp.). Tyto komponenty jsou využity ve spoustě projektů, jako například systém pro řízení obsahu Drupal nebo software phpBB pro spouštění fór. Mimo jiné se na tento framework spoléhá i Laraval (framework zmíněný výše) [20].

**CodeIgniter**

CodeIgniter je odlehčený PHP framework, který není striktně založen na vzoru MVC. Metody třídy komponenty řadiče (Controller) je sice nutné, modely a pohledy však už povinné nejsou. Jedná se o velmi štíhlý framework, který dovoluje přidávání pluginů třetích stran pro přidání komplikovanějších funkcionalit. Instalace tohoto frameworku je velmi rychlá a snadná, neboť vyžaduje jen minimum konfigurace. Velkou výhodou je, že tento framework funguje hladce téměř na všech sdílených a dedikovaných hostovacích platformách [20].

### 2.2.2 Java EE

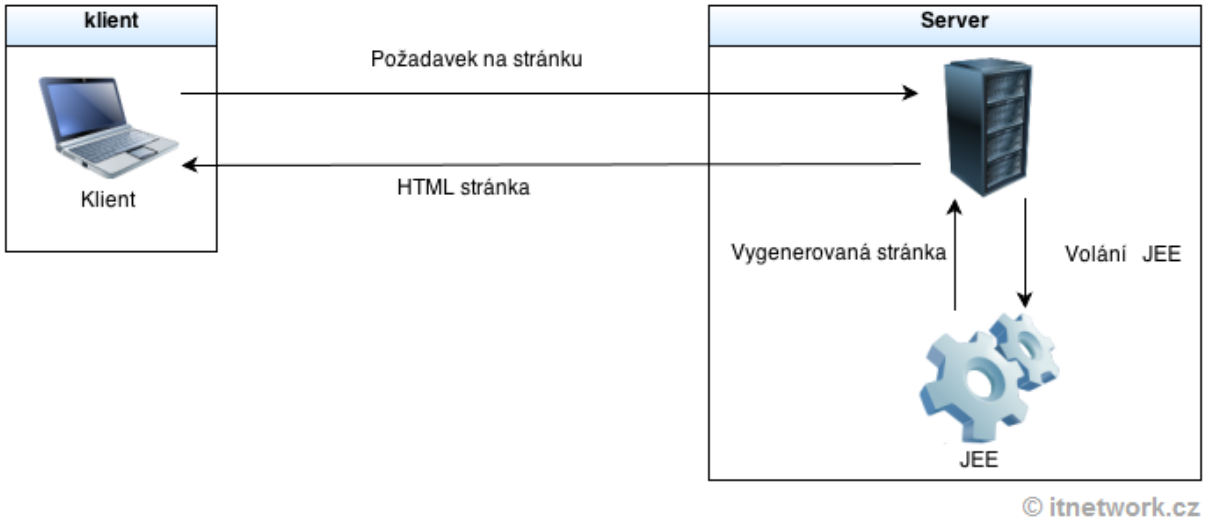
Základem Java EE (Enterprise Edition) je Java SE (Standard Edition), přičemž Java EE poskytuje knihovny pro usnadnění vývoje komplexnějších aplikací. Termín Java označuje kromě programovacího jazyka také platformu, která se skládá z virtuálního stroje a API (Application Programming Interface). Virtuální stroj je prostředí, kde běží naše aplikace (třídy přeložené do byte kódu) a je navržen pro určitý hardware a operační systém. API je sada předprogramovaných tříd, které je možné využít při psaní vlastních tříd pro přístup k funkcionalitám virtuálního stroje. Jednotlivé platformy se liší právě ve zmíněném API [10].

Jak již název napovídá (enterprise = podnik), Java EE obsahuje mnoho hotových řešení, využitelných pro rozsáhlé webové aplikace. V praxi se jedná o nejrozšířenější podnikovou technologii, kterou používá velké množství firem, a která obsahuje hodně různých pokročilých technologií, ze kterých se výsledná aplikace skládá. Je to tedy opravdu robustní řešení, které je schopné uspokojit poptávku náročných aplikací, jako jsou státní registry nebo bankovní aplikace [10].

Podnikovou aplikací, v tomto slova smyslu, je myšlena aplikace, která:

- Obsluhuje velké množství uživatelů ve stejném okamžiku.
- Pracuje s velkým množstvím dat v databázi.
- Komunikuje s dalšími systémy.
- Je robustní a bezpečná [10].

Zároveň taková aplikace pracuje na třívrstvé architektuře, což v praxi znamená, že aplikace je rozdělena na databázovou vrstvu, vrstvu obchodní logiky a prezentační vrstvu [10].




**Obrázek 1 - Grafické znázornění architektury klient-server pro JEE**

*Zdroj: [10]*

JEE funguje, stejně jako dříve zmíněné PHP na architektuře klient-server, takže výstupem aplikace je HTML stránka. Na rozdíl od JSE, kde Java aplikace běží na počítači klienta, JEE běží na straně serveru, kde na základě požadavků od klienta vygeneruje webovou stránku (včetně například vytažení dat z databáze) a poté ji vrátí klientovi [10].

Konkrétně běží JEE na tzv. aplikačním serveru, který zpracovává http/HTTPS požadavky klientů, obstarává spojení s databází nebo třeba odesílá emaily. Jednotlivé aplikační servery poskytují implementaci rozhraní JEE. Mezi nejznámější patří například JBoss Application Server, GlassFish, WebSphere Application Server nebo WebLogic Server [10].

JavaEE obsahuje například následující technologie:

- JSP (Java Server Pages) – technologie pro vkládání direktivy do HTML kódu, která spustí Java kód (vloží data, která Java získala z databáze).
- JSF (Java Server Faces) – konkurence JSP, reprezentace celé stránky jako XML souboru, web se skládá z předpřipravených komponent (tabulek, seznamů), které lze plnit daty z Javy.
- JDBC (Java DataBase Connectivity) – standardní rozhraní pro práci s databázemi v SQL.
- JPA (Java Persistence API) – rozhraní pro objektovou práci s daty, komunikace s databází přes mezivrstvu ORM.
- EJB (Enterprise Java Beans) – komponenty obchodní logiky.
- Spring framework – framework třetí strany, konkurence k JSF [10].

## 2.3 Shrnutí

Vzhledem k výše zmíněnému popisu všech technologií je zjevné, že i v alternativních technologiích je potřeba ovládat alespoň základy použitých (HTML, PHP, CSS a JavaScript). Jelikož autor ještě nemá dostatek zkušeností ani se zmíněnými použitými technologiemi, neboť se stále ještě učí, vybral si tyto, aby prohloubil své znalosti, a aby později na tyto základy mohl navazovat s alternativními technologiemi, především s frameworky. S HTML a PHP již zkušenosti měl, ale ne tak velké, aby se daly považovat za základ, nutný pro pochopení a práci s frameworky.

S Javou se autor setkal již dříve a měl mnoho příležitostí prohlubovat své znalosti v této kategorii. Jednalo se však vždy pouze o Javu SE (Standard Edition) a v poslední době se autor zaměřoval na návrh a tvorbu webových stránek právě v HTML, PHP, CSS a JavaScript. Proto se rozhodl využít tuto bakalářskou práci i pro prohloubení znalostí v těchto technologiích.

# 3 TECHNOLOGIE PRO NÁVRH A TVORBU DATABÁZE

Obdobně, jako se předchozí kapitola zaměřovala na aktuálně nejpoužívanější technologie pro vývoj webových aplikací, tato kapitola bude stejným způsobem zaměřená na aktuálně nejpoužívanější technologie pro návrh a tvorbu databáze. První část bude opět rozebírat technologii, použitou v této práci pro tvorbu databáze pro webovou aplikaci pro správu skladového systému, a druhá část bude popisovat alternativní technologie.

Databáze je uspořádaná množina dat, uložená na paměťovém médiu. Rovněž tento výraz může označovat i software pro manipulaci s daty neboli systém řízení báze dat (v angličtině Relational Database management System). Relační databáze je databáze, která je založena na tabulkách (řádky jsou záznamy a sloupce vlastnosti – atributy) [31].

Komunikace s databází se uskutečňuje pomocí jazyka SQL (Structured Query Language). Tento jazyk se během svého vývoje postupně standardizoval a tyto standardy pak implementuje, ať už do větší či menší míry, každá databáze. Většina databázových systémů si navíc přidává vlastní konstrukce, tudíž SQL pro jednotlivé databáze se může lišit [31].

## 3.1 Použitá technologie

### 3.1.1 Oracle

Oracle je firma poskytující nejvyužívanější databázový server pro strukturovanou správu dat. Díky multi-uživatelskému prostředí a zabezpečení proti neautorizovaným přístupům se jedná o velmi výkonný databázový server. Databáze je zde soubor dat uvnitř databázového serveru. Každá běžící Oracle databáze je přiřazena k Oracle instanci, což je kombinace alokované paměti a Oracle procesů [19].

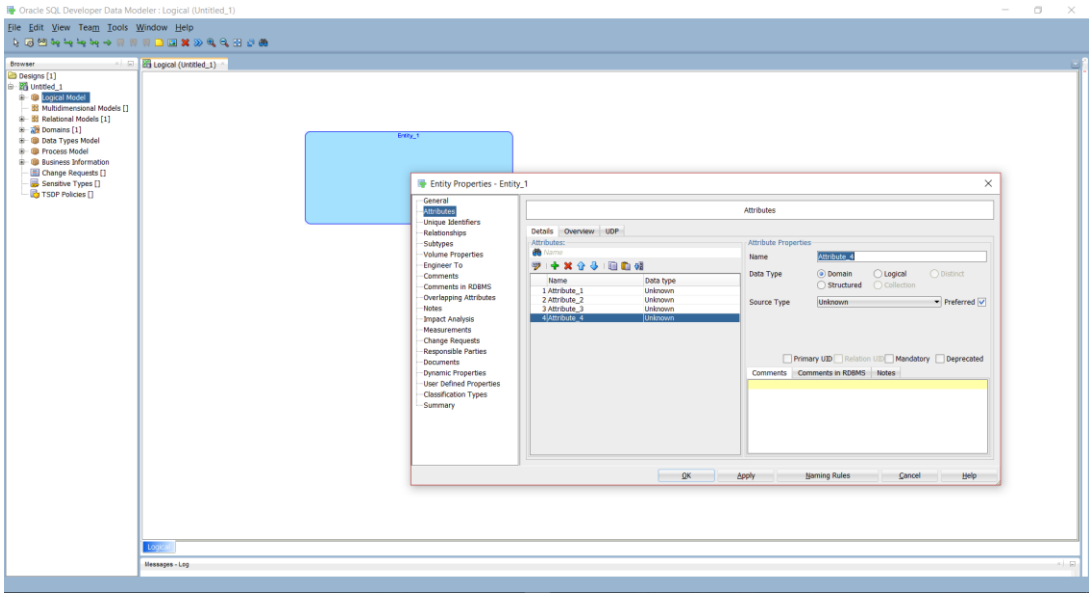
Uživatelské účty v Oracle databázi jsou definované uživatelským jménem a uživatelskými atributy, mezi které patří:

- Přihlašovací heslo.
- Oprávnění a role.
- Defaultní umístění pro databázové objekty [19].

Při vytvoření uživatele dojde i k vytvoření schéma pro tohoto uživatele, což je logický kontejner pro databázové objekty (tabulky, pohledy, triggery atp.). Jméno schématu je shodné s uživatelským jménem [19].

Oracle má vlastní software pro snadnou a uživatelsky přívětivou tvorbu databází. Prvním krokem při vytváření databáze je vytvoření logického modelu v programu Oracle SQL

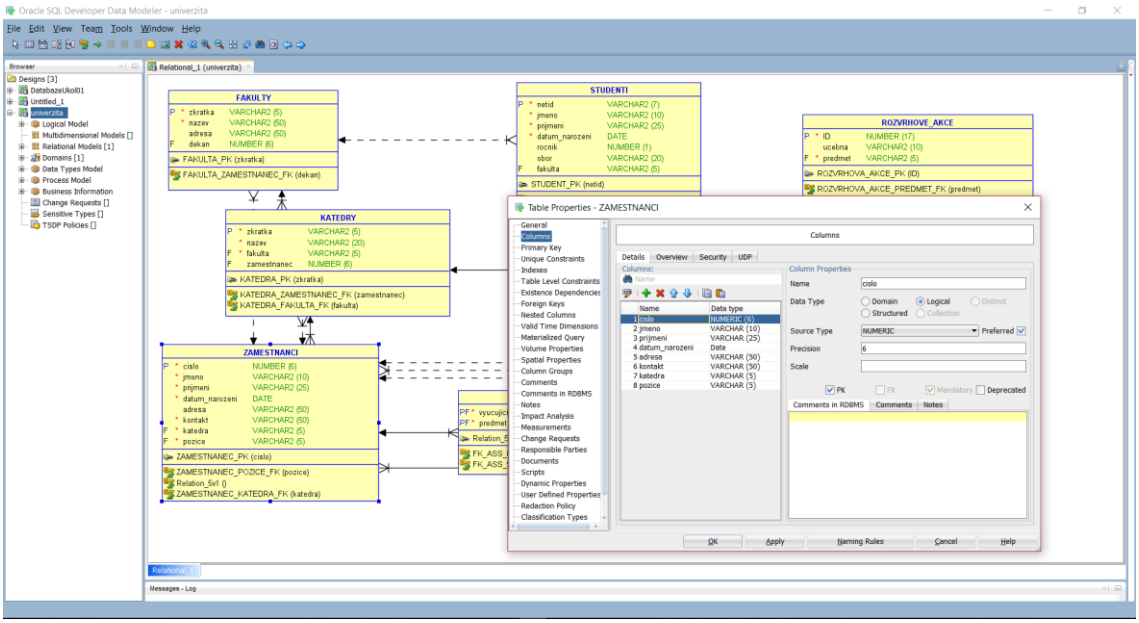
Developer Data Modeler. Při vytváření logického modelu je potřeba dodržovat normální formy a zásady pro tvorbu databáze.



**Obrázek 2 - Oracle Data Modeler logický model**

*Zdroj: Vlastní zpracování*

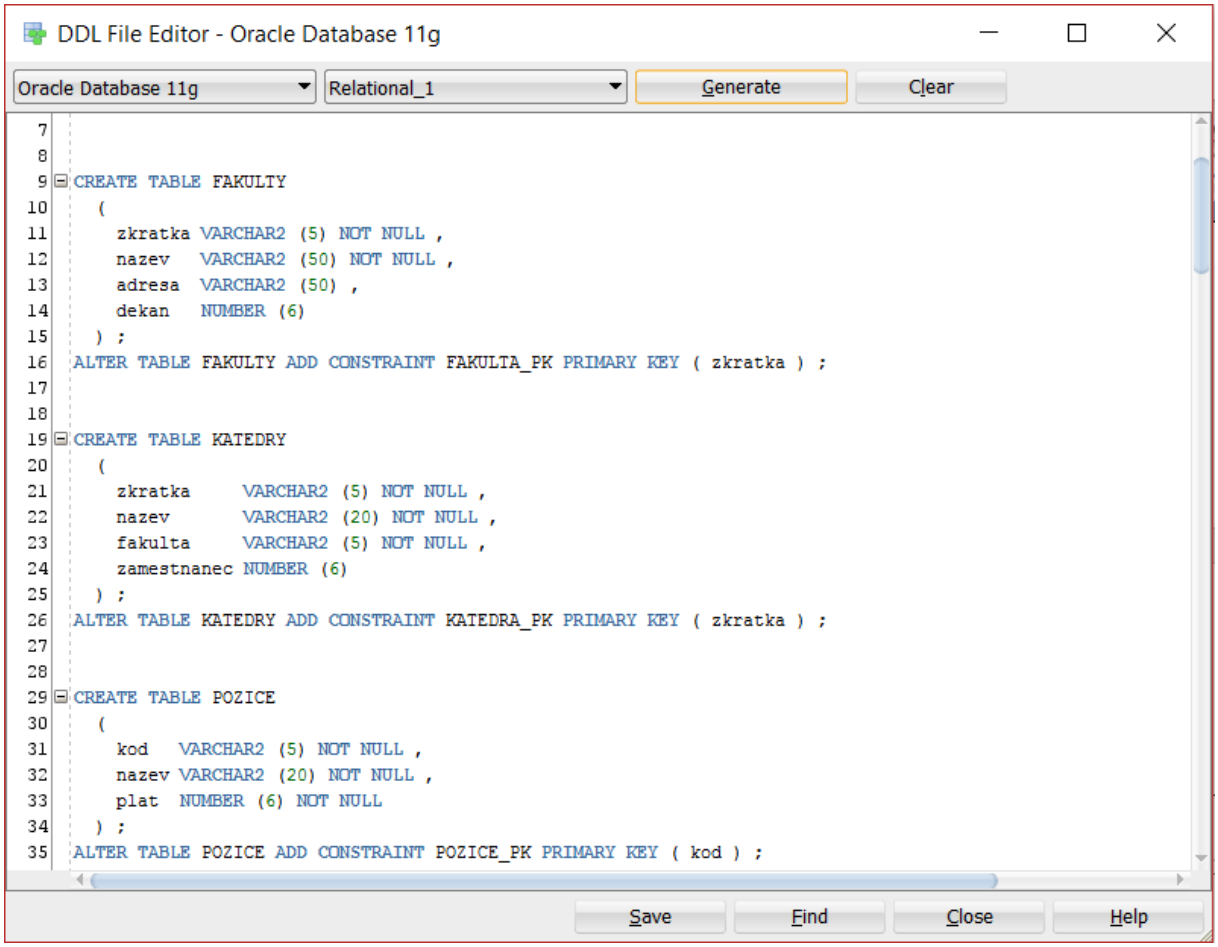
Po vytvoření logického modelu pomocí funkce SQL Data Modeleru předěláme model na model relační, který především řeší vazby více ku více. V relačním modelu je již potřeba doplnit datové typy atributů, názvy vazeb a ohlídat si případné přepsání názvů atributů, především u cizích klíčů.



**Obrázek 3 - Oracle Data Modeler relační model**

*Zdroj: Vlastní zpracování*

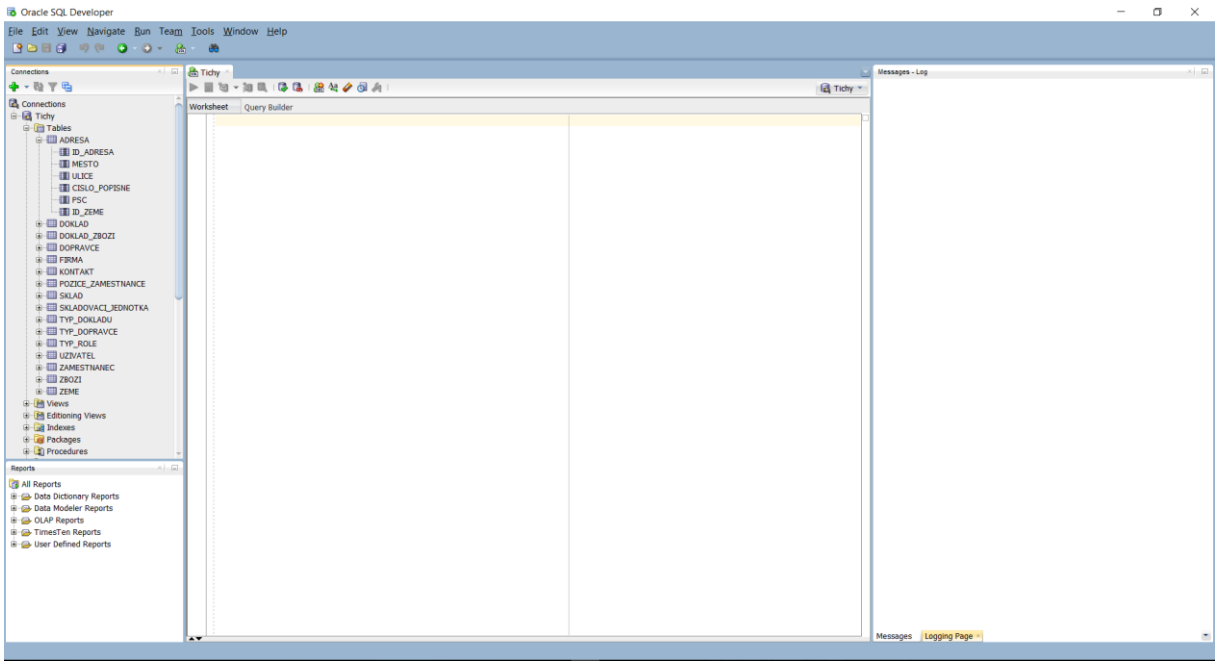
Po zkontrolování relačního modelu a upravení všech vazeb již můžeme přejít ke tvorbě DDL skriptu. Na to opět můžeme využít funkci Oracle SQL Data Modeleru, která nám skript vytvoří přímo z relačního modelu.




**Obrázek 4 - Vygenerovaný DDL skript v Oracle Data Modeleru**

*Zdroj: Vlastní zpracování*

Tento skript je potřeba zkontrolovat a opravit případné chyby. Pokud se povedlo vytvořit skript bez chyb, můžeme si jej uložit a otevřít v hlavním programu pro správu databáze, Oracle SQL Developer. V tomto programu je již potřeba se přihlásit k databázovému serveru. Po přihlášení se dostaneme k vlastnímu schématu, do kterého můžeme přidávat tabulky, funkce, procedury, triggery a další.




**Obrázek 5 - Oracle SQL Developer**

*Zdroj: Vlastní zpracování*

## PL/SQL

PL/SQL je rozšíření jazyka SQL, implementované v databázovém systému Oracle. Umožňuje psát imperativní kód, který je vykonávaný systémem řízení báze dat. Díky tomuto rozšíření je možné napsat část logiky aplikace přímo v databázovém systému, čímž můžeme docílit zrychlení aplikace, neboť není nutné natahovat data z databáze do aplikace a až tam je zpracovat, ale stačí pouze zadat požadavek a data se zpracují přímo v databázovém systému [30].

PL/SQL nám umožňuje:

- Vytvářet procedury (skupiny PL/SQL příkazů, sdružených pod jedním názvem, vhodné vytvářet v případě více příkazů, které se volají najednou v případech, které se často opakují, mohou mít vstupní i výstupní parametry a dovolují měnit tabulky – přidávat, měnit, mazat).

```
CREATE OR REPLACE PROCEDURE zvyseni_mzdy (procento IN NUMBER) AS
BEGIN
  UPDATE A_HR.zamestnanci SET mzda = mzda * (1+procento/100);
  COMMIT;
END;
```

- Vytvářet funkce (skupiny PL/SQL příkazů, sdružených pod jedním názvem, vracející jednu hodnotu, vyplatí se používat v případech podobných případům užití procedur, vždy vrací jednu hodnotu a nedovolují měnit tabulky).

```
CREATE OR REPLACE FUNCTION pocet_smen(Id_trp IN NUMBER)
RETURN NUMBER
AS
      v_pocet NUMBER;
  BEGIN
SELECT count(*) INTO v_pocet FROM A_O_SNEHURCE.tezby
WHERE Id_trpaslika=Id_trp AND skutecnost>0;
RETURN v_pocet ;
END;
```

- Vytvářet triggery (spouště – skupiny PL/SQL příkazů, které jsou zavolány před určitou akcí nebo těsně po akci na konkrétní tabulce, dovolují měnit tabulky (přidávat, odebírat i mazat, vhodné používat místo procedur nebo funkcí u akcí, které jsou přímo spojené se změnou v tabulce).

```
create or replace TRIGGER PRIDANI_FIRMY
  BEFORE INSERT on FIRMA
  FOR each ROW
  BEGIN
    :NEW.heslo := md5hash(:NEW.HESLO);
  END;
```

- Používat lokální proměnné a cykly ve výše zmíněných [30].

## 3.2 Alternativní technologie

### 3.2.1 MySQL

MySQL je v současné době jednou z nejoblíbenějších open source databází, především díky spolehlivosti, výkonnosti a jednoduchosti na použití. Za posledních několik let se stala nejpoužívanější databází pro webové aplikace jako například Facebook, Twitter, Youtube nebo Yahoo! [4].

Na rozdíl od jiných databází (například Oracle), které k autentizaci používají jen uživatelské jméno a heslo, MySQL používá jeden parametr navíc, lokaci. Tento parametr má většinou podobu hostname nebo adresy IP. Díky tomuto parametru je možné nastavit odlišná hesla a práva na základě hosta, ze kterého bylo připojení navázáno (uživatel Jan, který se přihlásil z abc.cz nemusí být ten stejný Jan, přihlašující se z xyz.cz) [22] .

**Tabulka 2 - Porovnání objektů schématu v Oracle a MySQL**

| Oracle | MySQL |
|---|---|
| AFTER trigger | Trigger |
| BEFORE trigger | trigger |
| PL/SQL function | Routine |
| PL/SQL procedure | Routine |
| Sequence | AUTO_INCREMENT for a column |

*Zdroj: [22]*

**Tabulka 3 - Porovnání datových typů MySQL a Oracle**

| Datový typ MySQL | Datový typ Oracle |
|---|---|
| INT | NUMBER(10,0) |
| DOUBLE, DECIMAL | FLOAT(24) |
| NUMERIC | NUMBER |
| TEXT | VARCHAR2 (CLOB) |
| TIMESTAMP | DATE |

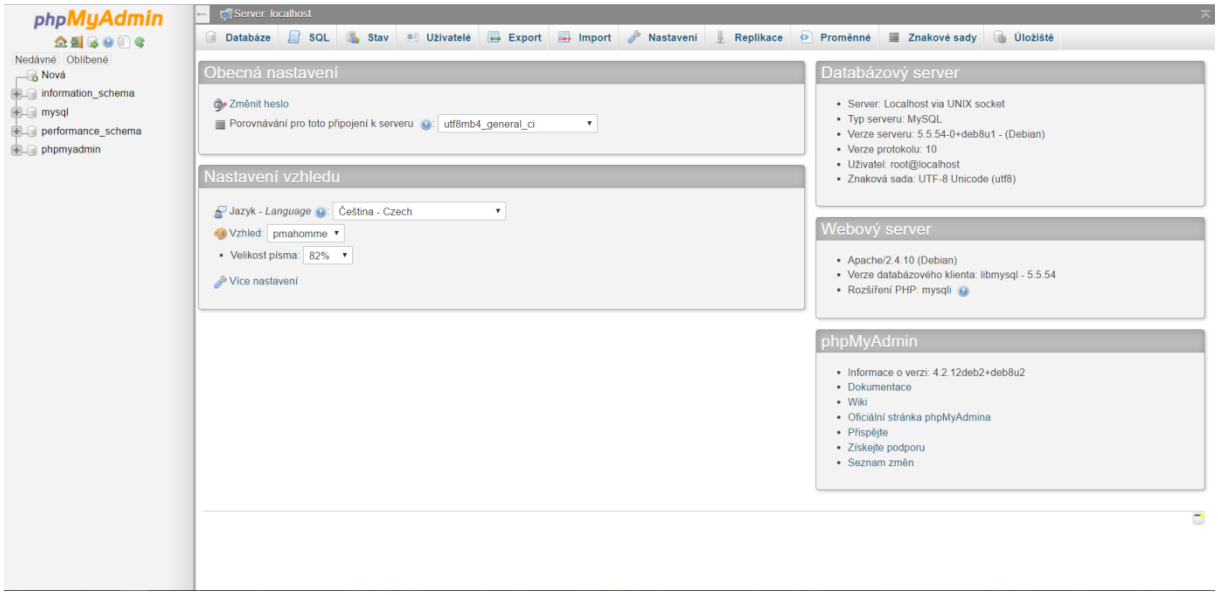
*Zdroj: [22]*

Pro správu této databáze se velmi často používá programový nástroj phpMyAdmin, který je zdarma, napsaný v PHP a jehož účelem je poskytnutí správy MySQL databáze na webu. Nejčastěji používané operace (např. správa databáze, tabulek, sloupců, relací nebo uživatelů) mohou být provedeny pomocí přívětivého uživatelského rozhraní, nebo klasicky pomocí SQL příkazu. Uživatelům je zde k dispozici [8]:

- Intuitivní webové rozhraní.
- Podpora většiny MySQL vlastností (prohlížení, vytváření, kopírování, správa účtů atd.).
- Import dat z CSV a SQL.
- Export dat do CSV, SQL, XML, PDF a dalších formátů.
- Vyhledávání v celé databázi nebo její části.
- Administrace více serverů [8].


**Obrázek 6 - Přihlášení do phpMyAdmin**

*Zdroj: Vlastní zpracování*




**Obrázek 7 - prostředí phpMyAdmin po přihlášení**

*Zdroj: Vlastní zpracování*

### 3.2.2 SQLite

SQLite je procesová knihovna, která implementuje transakční SQL databázové nástroje, bez nutnosti konfigurace a nezávisle na serveru. Kód pro SQLite je veřejně přístupný, takže je zdarma pro komerční i soukromé užití. Na rozdíl od většiny ostatních databází, SQLite nemá samostatné serverové procesy, tudíž čte a zapisuje přímo do konkrétních diskových souborů. Kompletní databáze, včetně tabulek, triggerů, pohledů a dalšího, je obsažena v jediném

diskovém souboru. Formát souboru databáze SQLite je multiplatformní (nezávisí na architektuře)[5].
SQLite je kompaktní knihovna, jejíž velikost se pohybuje kolem rozmezí 300-500 KiB a může běžet i na minimálním místě pro stack (4 KiB) i pro heap (100 KiB). Díky tomu je SQLite velmi používaná technologie především pro omezené přístroje, jako jsou telefony, PDA nebo MP3 přehrávače. Přestože databáze funguje tím rychleji, čím více místa má k dispozici, i na zařízeních s velmi malou pamětí má velmi dobrý výkon [5].

SQLite prochází při každé změně velmi náročnými testy, což přispívá k velké spolehlivosti, kterou se tato technologie pyšní. Testováním a ošetřováním chyb je dosaženo velmi dobrých reakcí v případech chyb alokace paměti, nebo chyb vstupu a výstupu [5].

Na rozdíl od Oracle SQL, SQLite nepodporuje práci se soubory typu XML. Jelikož nemá serverové procesy, není tu možné vytvářet serverové skripty, jako například v Oracle SQL pomocí PL/SQL [5] [26].

### 3.2.3 PostgreSQL

PostgreSQL je objektově-relační databázový systém, který je vytvořený pod licencí open source (volně šiřitelný). Funguje na většině operačních systémů (Linux, Unix, Windows). Tato technologie je plně v souladu s ACID (Atomicity – atomicita, Consistency – konzistence, Isolation – izolace, Durability – odolnost) vlastnostmi a podporuje cizí klíče, pohledy, spouště (triggery) a procedury (v několika jazycích). PostgreSQL obsahuje většinu datových typů SQL:2008 (INTEGER, NUMERIC, VARCHAR, DATE atd.) a podporuje ukládání velkých binárních objektů (obrázků, písní, videa) [6].

PostgreSQL se může chlubit například následujícími vlastnostmi:

- MVCC (Multi-Version Concurrency Control) – souběžný přístup.
- Okamžitá obnova.
- Online i offline zálohy.
- Plánovač a optimalizátor dotazů [6].

PostgreSQL je přizpůsoben standardům ANSI-SQL:2008 a to především v podpoře pod-dotazů (včetně pod-výběrů v klauzuli FROM) a možností vytvoření více schémat pro jednu databázi. Pokročilý indexovací systém GiST (Generalized Search Tree – generalizovaný vyhledávací strom) nabízí velké množství vyhledávacích a řadících algoritmů (jako například B-strom,

B+-strom, R-strom a další). Pomocí tohoto systému můžeme specifikovat co ukládáme, jak to chceme uložit, nebo jak bude možné tato data vyhledat [6].

PostgreSQL zvládá použití uložených procedur hned v několika jazycích (jako například Java, Perl, Python, Ruby, C/C++ a vlastní PL/pgSQL, což je obdoba Oracle PL/SQL). V základní knihovně je obsaženo velké množství funkcí od základů matematiky až po operace s textem (stringem). Součástí je také framework, který dovoluje vývojáři definovat a vytvořit vlastní datový typ spolu s připojenými funkcemi a operátory [6].

## 3.3 Shrnutí

Při výběru technologie pro návrh a tvorbu databáze se autor rozhodoval mezi SQL a MySQL, jelikož s oběma technologiemi měl zkušenosti. Vzhledem k možnosti využít školní Oracle server a nedávnými zkušenosti s Oracle SQL (včetně PL/SQL procedur a funkcí) se rozhodl využít právě tuto technologii, přestože pro webové aplikace se většinou využívá spíše MySQL. Dalším důvodem byl záměr tuto aplikaci později vytvořit i jako desktopovou aplikaci a aplikaci pro přenosná zařízení (tablety, chytré telefony).

**Tabulka 4 - Srovnání databázových technologií**

| | **Oracle** | **MySQL** | **PostgreSQL** |
|---|---|---|---|
| **Licence** | Komerční | Open Source | Open Source |
| **Implementační jazyk** | C a C++ | C a C++ | C |
| **Podpora XML** | Ano | Ano | Částečná |
| **Podpora SQL** | Ano | Ano | Ano, s výjimkami |
| **Serverové skripty** | PL/SQL | Ano/SQL | Uživatelské funkce |
| **Spouště (triggery)** | Ano | Ano | Ano |
| **Dělení** | Horizontální | Horizontální | Ne, pouze dědičnost |
| **Replikační metody** | Mistr-mistr, mistr-otrok | Mistr-mistr, mistr-otrok | Mistr-otrok |

*Zdroj: [25]*

Z tabulky srovnání databázových technologií je zjevné, že by bylo ve všech směrech ideální vybrat si MySQL, jehož hlavní výhoda oproti Oracle SQL je volná licence. Pokud by se autor znovu rozhodoval a neměl by k dispozici server Oracle (jako tomu bylo v tomto případě), rozhodl by se zcela jistě pro MySQL. To by však znamenalo naučit se vytvářet a upravovat procedury, funkce a spouště (triggery) v této databázové technologii. S tím se totiž doposud mimo Oracle SQL nesetkal, tudíž by to byla další překážka.

# 4 POROVNÁNÍ S EXISTUJÍCÍMI APLIKACEMI

Tato kapitola bude zaměřená na průzkum trhu v oblasti aplikací, určených k řešení stejných nebo podobných problému, které má za úkol řešit tato webová aplikace pro správu skladového systému. Autor vybral pár aplikací (webové i desktopové), se kterými se buď někde setkal, nebo o kterých se doslechl. U každé se pokusí napsat dostatek informací, pro nastínění způsobu implementace, a porovnat dané řešení s vlastním.

Touto problematikou se obecně zabývá dost programů a aplikací, které jsou primárně určeny pro správu účetnictví. Jelikož však s účetnictvím jsou většinou spjaty sklady pro uchovávání materiálů a surovin za nejrůznějšími účely, byly do těchto programů přidány moduly pro správu skladového systému.

## 4.1 Helios

Podnikové informační systémy Helios jsou určeny k plnému pokrytí potřeb velkých, středních i malých firem a ostatních ekonomických subjektů. Tyto systémy jsou produktem společnosti Asseco Solutions, a.s., která je největším producentem podnikových informačních systémů na českém i slovenském trhu, a jejíž softwarová řešení jsou distribuována i na další zahraniční trhy v rámci střední Evropy. Společnost se zaměřuje na vývoj, implementaci a podporu specializovaných systémů pro organizace všech velikostí i oblastí působení [24].

Z opravdu širokého sortimentu produktů Helios většina obsahuje modul pro správu skladového systému nebo majetku. Ze všech produktů, které obsahují tyto moduly, se autor rozhodl vybrat dva zástupce, které mají největší podporu správy skladových systémů. Důležitý rozdíl mezi produkty Helios a autorovým systémem je fakt, že systémy Helios jsou realizovány jako desktopové, případně mobilní aplikace, nikoli webová aplikace [24].

### 4.1.1 Helios Green

Helios Green je moderní ERP (Enterprise Resource Planning – Plánování podnikových zdrojů) systém pro středně velké a velké firmy. Tento systém obsahuje modul pro logistiku a sklady, který mimo jiné umožňuje evidovat skladové zásoby, sledovat jejich pohyb, šarže, výrobní čísla a podobně. Stejně jako autorova práce, i tento systém umožňuje evidenci skladových zásob (zboží, materiálu) na jednotlivých skladech i umístěních a zajišťuje tok dokladů pro logistický proces (příjemky, výdejky atp.) [13].

Helios Green navíc zajišťuje účtování prvotních dokladů dle nastavení, správu reklamačních i komisních skladů, pomáhá s tvorbou statistik a provádí inventury a závěrky skladů. Tento systém kromě modulu pro logistiku a sklady obsahuje i moduly pro:

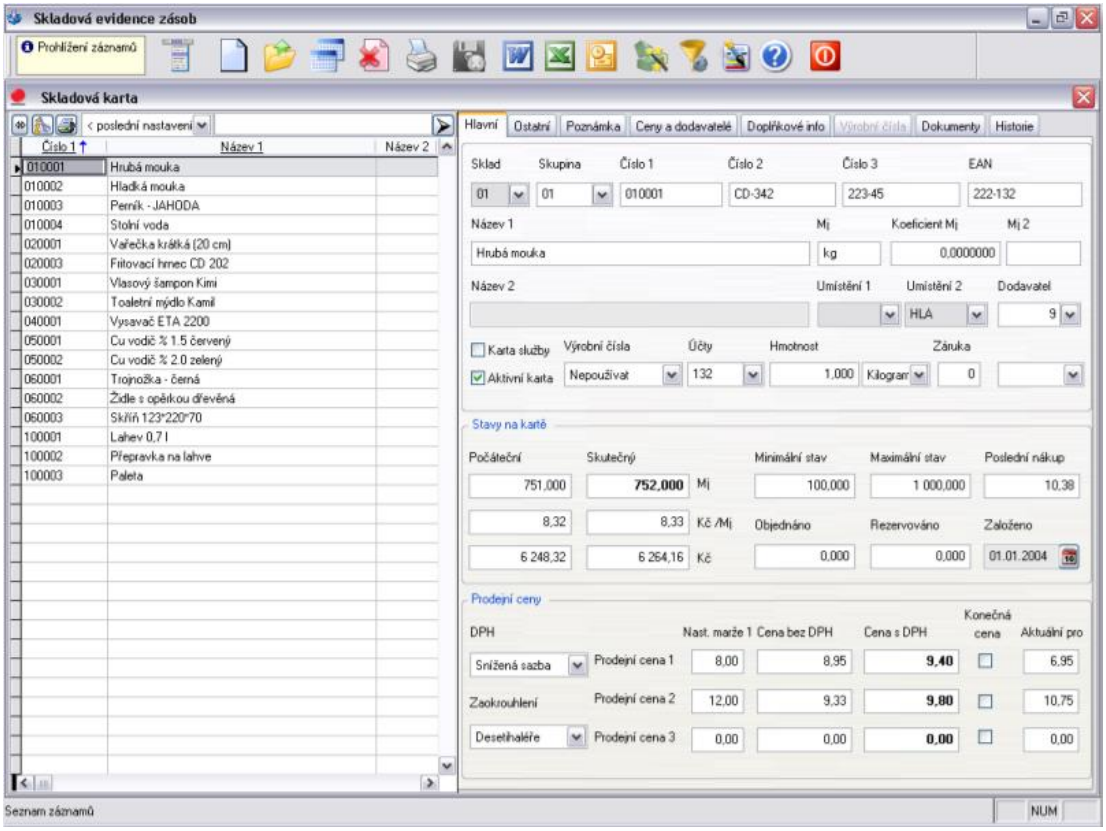
- Řízení společnosti.
- Finance a ekonomiku.
- Obchod a marketing.
- Lidské zdroje.
- Provozní a podpůrné agendy.
- Výrobu [13].

### 4.1.2 Helios Red

Helios RED je ekonomický systém pro podnikatele a menší firmy, který pracuje veškerou potřebnou agendu. Balíček tohoto systému je dokonce možné sestavit podle vlastních aktuálních potřeb. Kromě skladové evidence je možné si do balíčku navolit například daňovou evidenci, bankovní operace, personalistiku a mzdy, pokladní prodej, knihy jízd a cestovní příkazy nebo manažerské vyhodnocování [14].

Modul Skladová evidence je možné používat i samostatně, mimo aplikace Helios RED. Základem evidence (stejně jako v autorově systému) je skladová karta, obsahující identifikační číslo, popis položky měrnou jednotku a identifikaci umístění. Tento modul je možné navázat na programy Účetnictví, Fakturace, Nákup a Prodej. Rovněž je zde možné sledovat historii operací na skladové kartě, vystavovat doklady, spojené s příjmem i výdejem zboží. Oproti autorově systému je zde možné například [14]:

- Využívat on-line čtečku čárového kódu či datový terminál.
- Variabilní způsob inventury skladů.
- Evidence výrobních čísel.
- Napojení na libovolný e-shop pomocí XML [14].




**Obrázek 8 - Skladová evidence zásob Helios RED**

*Zdroj: [14]*

Z popisu produktů Helios společnosti Asseco Solutions, a.s., je zřejmé, že tyto systémy mají větší množství modulů a funkcí než autorův systém. Příčinou je beze sporu větší časová i personální dotace na vývoj jednotlivých modulů i celých produktových řešení. Na stránkách není možné zjistit, jaká technologie byla využita k vývoji aplikace, nebo realizace databáze.

Jelikož se však jedná převážně o desktopové aplikace, s největší pravděpodobnosí to nebude ani jedna z výše zmíněných technologií pro tvorbu webových aplikací. Databáze bude zřejmě řešena pomocí Oracle SQL, SQLite nebo PostgreSQL. Tento fakt však dává částečnou výhodu autorově aplikaci, neboť jakožto webová aplikace je nezávislá na platformě nebo operačním systému. Jediné problémy mohou vzniknout v případě použití webového prohlížeče, který může některé komponenty webu zobrazovat jinak, nebo vůbec.

## 4.2 Prodej Online

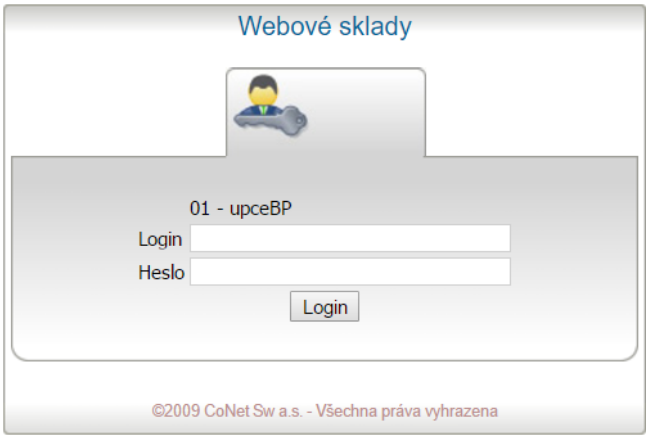
Prodej Online nabízí jako jeden z mála řešení skladového programu, které nevyžaduje žádnou instalaci do počítače. Díky internetu je totiž možné využívat tento produkt kdekoli. Systém je navíc připraven na EET, tudíž automaticky odesílá informace o dokladech, placených v hotovosti do elektronické evidence tržeb. Webové a serverové rozhraní umožňuje mít data ze

všech provozovaných obchodů, skladů či firem na jednom místě a chrání před problémy, vzniklými poruchami hardwaru nebo ztrátou dat [12].

Program není závislý na operačním systému, neboť se obsluhuje jako klasická webová stránka (ve webovém prohlížeči). Díky tomuto faktu nemusíte do svého počítače nic stahovat, instalovat ani se starat o aktualizace. Všechno se děje na straně webových stránek, na které se pouze přihlásíte přes prohlížeč (z libovolného zařízení) a můžete pracovat [12].

I přes to, že se jedná „jen“ o webovou stránku, je zde k dispozici velké množství možností. Jedná se v podstatě o podobný systém, který je předmětem této práce, avšak opět s větším množstvím možností a větší propracovaností, což ale muselo mít vliv na časovou náročnost vývoje.

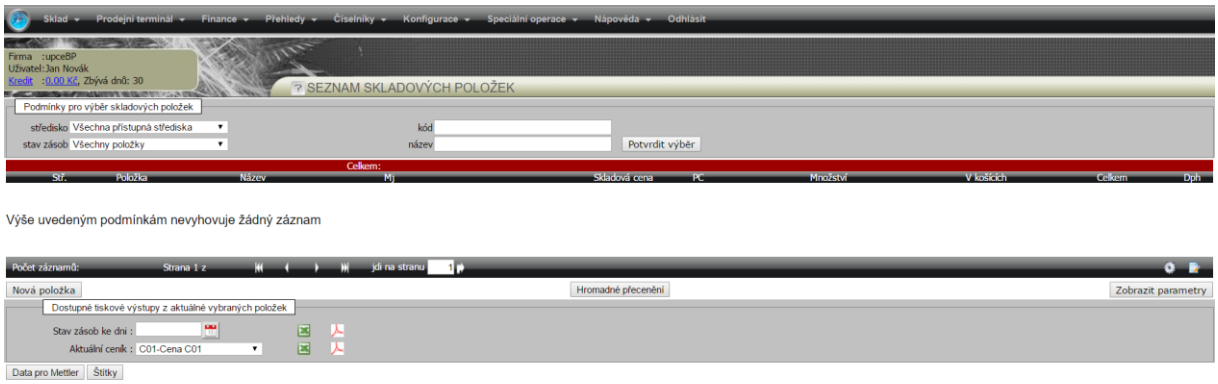
V systému je například možná hromadná fakturace, vytváření šablon faktur a plánování fakturace nebo dokonce fakturace přímo u zákazníka. Dále je zde možné vytvářet příkazy k úhradě, podrobně konfigurovat uživatelská práva i připojení vzdálených středisek s přístupem pro externí účtárny [12].




**Obrázek 9 - prodejOnline přihlášení do systému**

*Zdroj: [12]*

Z domovské stránky poskytovatele služeb Prodej Online se dá přihlásit do systému. Po kliknutí na „přihlásit“ se dostaneme na novou stránku, kde není nic než přihlašovací okno na bílém podkladu. Ani ve výchozím stavu, ani po vyplnění nesprávných údajů se nezobrazí možnost získat zapomenuté heslo, což je významným nedostatkem. Jistě je nějak možné ho zjistit, ale není zde uvedeno, jak. Po úspěšném přihlášení se dostaneme do skladového systému.




**Obrázek 10 - prodejOnline skladový systém**

*Zdroj: [12]*

Je zde daleko více funkcí než v autorově systému, což však vede i k menší uživatelské přívětivosti, neboť trvá nějaký čas, než se v takovémto systému člověk zorientuje. Všechny funkce jsou sdruženy ve vrchní části v kategorii, do které patří. Autor se snažil vydat se spíše cestou uživatelsky příjemného prostředí a ovládání, což se dle jeho subjektivního názoru na tomto webu nevyskytuje.

Soudě podle analýzy zdrojového kódu je tento web vytvořen pomocí podobných (ne-li stejných) technologií, kterých využil autor této práce. Je zde velké množství JavaScriptu spolu s HTML tagy. Jakou databázi tento skladový systém využívá, však není nikde uvedeno. Nejpravděpodobněji se bude jednat o MySQL, neboť je to nejčastější volba pro webové aplikace.

# 5 ANALÝZA A NÁVRH APLIKACE

Při vytváření jakékoli aplikace nebo programu je nutno projít několika fázemi, než se pustíme do samotného vývoje software. Prvním krokem je ve většině případů sběr požadavků a jejich specifikace. K těmto požadavkům se připojí jejich definice, významy a začne se vytvářet první metamodel. K modelu se sestaví případy užití, ve kterých se s tímto modelem bude pracovat.

Další dvě fáze jsou asi nejdůležitější a budou náplní této kapitoly. Jedná se o analýzu a návrh aplikace. Každá z těchto fází bude nejprve lehce teoreticky uvedena a poté bude provedena na autorově webové aplikaci pro správu skladového systému. Celá tato kapitola čerpá ze zdroje [29].

V této části je také vhodné stanovit hlavní požadavky na aplikaci. Rozhodně se nesmí opomíjet uživatelská přívětivost. Ta musí být na prvním místě, jelikož nikdo nechce používat aplikaci, kterou je složité ovládat a naučení se některých rutin může trvat hodiny. Je však potřeba zaměřit se na požadavky na samotný systém – co by měl umět a které prvky implementovat. Jelikož se jedná o webovou aplikaci, je určitě potřeba se také zaměřit na kompatibilitu prohlížečů.

Aplikace pro správu skladových systémů by měla zvládat správu více než jednoho skladu pro každou firmu, a pro každý sklad pak většího množství skladovacích jednotek, ve kterých bude velké množství zboží. Správa zboží a jednotek, ve kterých je uloženo je tedy základ, mimo to by aplikace měla zvládat udržovat databázi dopravců (jednotnou pro všechny firmy, do které by každá firma, využívající tuto aplikaci mohla přidávat). A samozřejmě ve skladovém systému je rovněž potřeba vystavovat a udržovat doklady o přijetí či výdeji zboží.

Vzhledem k tomu, že aplikace má více režimů (účelů) použití, měla by implementovat více než jeden uživatelský režim. Optimální množství uživatelských rolí by byly 3:

- Host – uživatelská role pro přístup zákazníků skladu, kteří budou moci pouze prohlížet sklady a zboží na nich umístěné.
- Uživatel – uživatelská role pro přístup zaměstnanců, kteří budou moci přidávat zboží, vystavovat doklady i upravovat skladovací jednotky.
- Administrátor – uživatelská role pro přístup administrátorů, kteří budou mít veškerá práva na úpravu všech tabulek i uživatelů.

## 5.1 Analýza

Analýza se v prostředí objektově orientovaného programování provádí za účelem tvorby analytického modelu, který je zaměřen na požadovanou funkčnost. Analytický model však

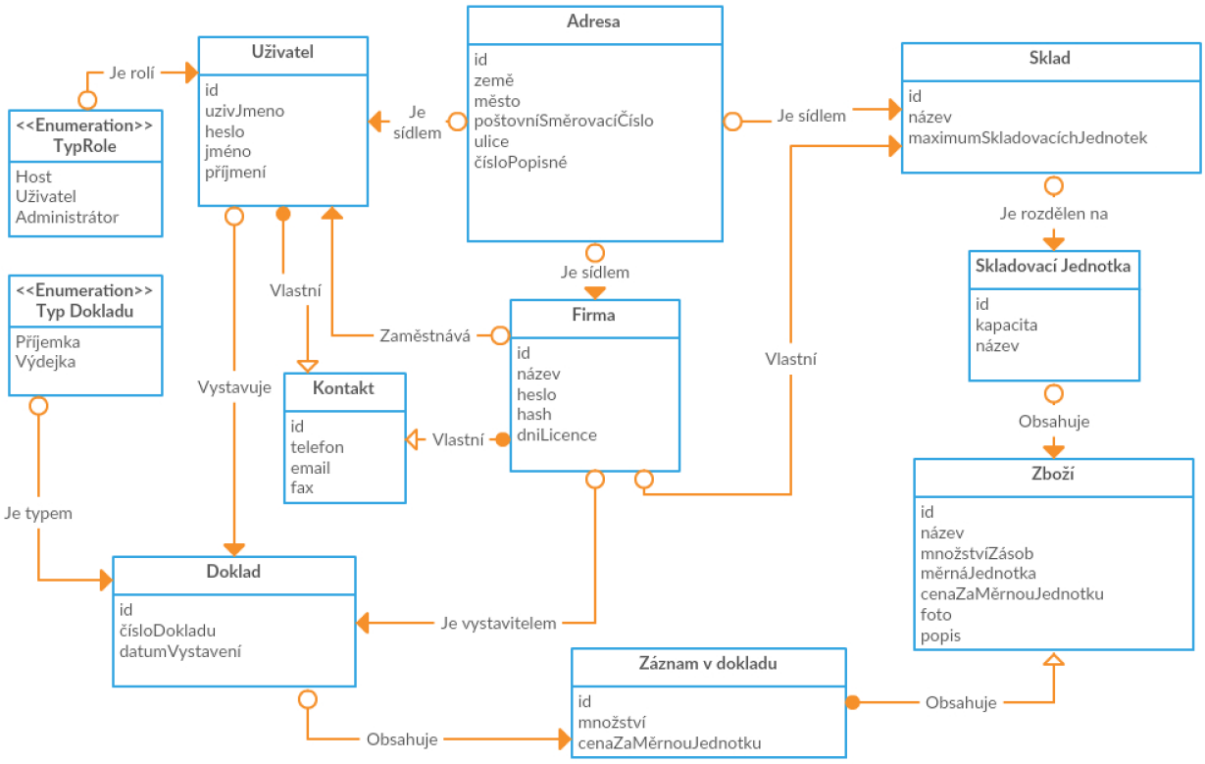
zůstává pouze u funkcí, které musí program umět, neřeší, jakým způsobem je bude provádět (to má na starosti následující fáze – návrh).

Prvním krokem je tvorba analytického metamodelu, který obsahuje hlavní artefakty analýzy (analytické třídy s klíčovými pojmy a realizace případů užití). V tomto kroku je důležité omezit se pouze na třídy, které jsou součástí slovníku problémové domény (obchodních požadavků). V Analýze je rovněž důležité zachycovat případy z určité perspektivy a nezacházet do přílišných detailů (to bude opět úkolem návrhu).

Analytický model musí být srozumitelný pro všechny zainteresované skupiny osob (uživatele, návrháře, vývojáře atp.). Díky dobré srozumitelnosti dosáhneme velkého počtu veřejných revizí, což povede ke zvýšení kvality výsledku.

Cílem této práce je vytvoření webové aplikace, která bude spolupracovat s databází, kde budou uložena všechna data. Proto diagram tříd bude obsahovat návrh databázové struktury. Třídy v tomto modelu budou představovat budoucí databázové tabulky a atributy sloupce v těchto tabulkách.

Pro tvorbu modelu bude využit univerzální jazyk pro vizuální modelování systémů UML (Unified Modeling Language). Tento jazyk je navržen na propojení nejlepších existujících postupů modelovacích technik a softwarového inženýrství. Není však omezen jen na modelování objektově orientovaných softwarových systémů, neboť má díky svým rozšiřovacím mechanismům velmi široké využití.




**Obrázek 11 – Analytický model (diagram tříd) z webové aplikace Creately**

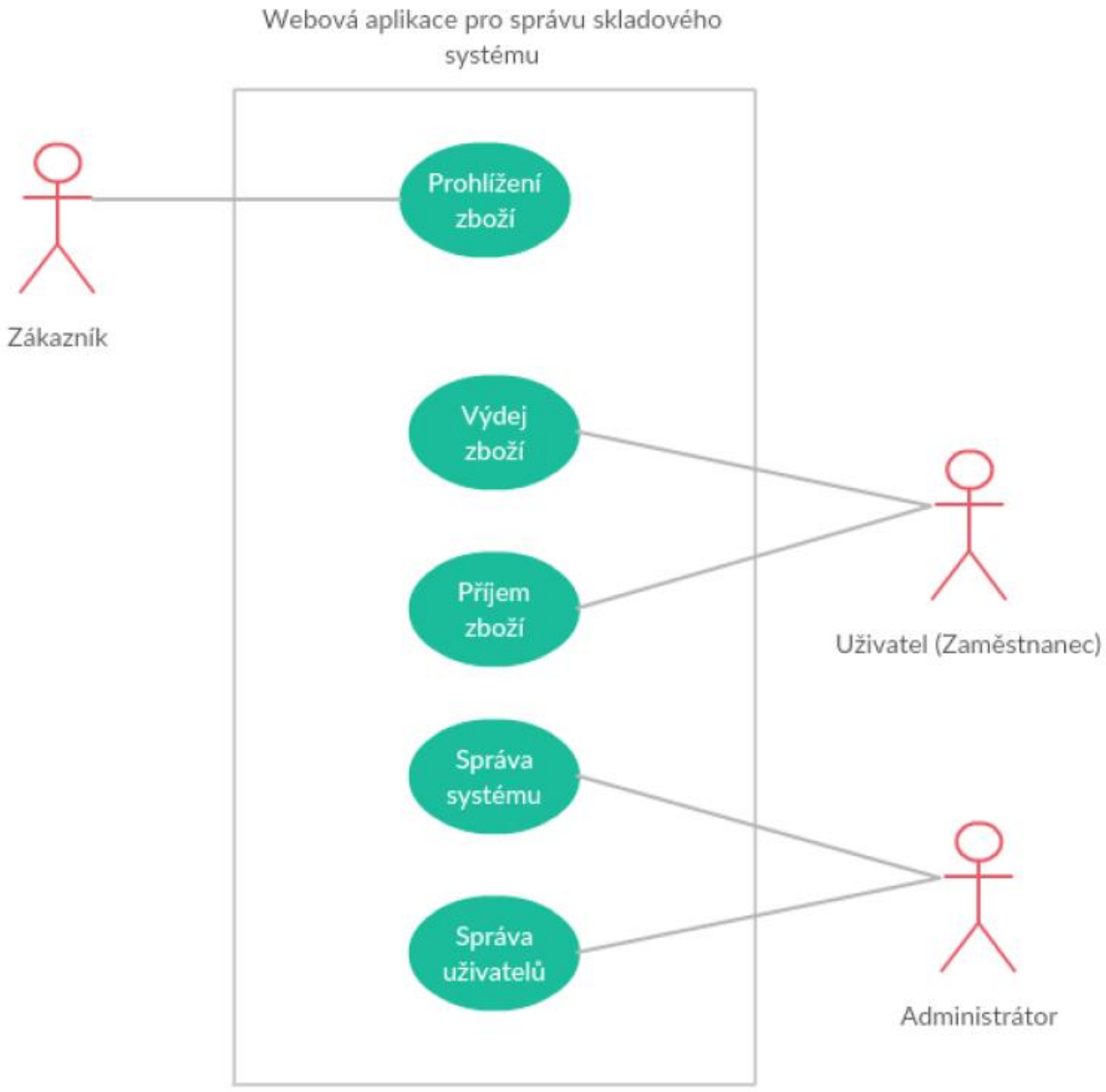
*Zdroj: Vlastní zpracování*

Druhým krokem jsou realizace případů užití, které slouží k popisu spolupráce instancí analytických tříd pro dosažení požadovaného chování systému. Při analýze průběhů realizace je možné zjistit interakce jednotlivých analytických tříd, což vede k realizaci chování specifikovaného případem užití. Během tohoto procesu je možné odhalit nové analytické třídy.

Při tomto kroku se také dá zjistit, jaké zprávy musejí instance daných tříd posílat instancím jiných tříd za účelem realizace určeného chování. Tyto skutečnosti odhalí:

- Klíčové operace analytických tříd.
- Klíčové atributy analytických tříd.
- Důležité relace mezi analytickými třídami.

Jelikož se jedná o webovou aplikaci s více uživatelskými rolemi, bude zde větší množství případů užití. V režimu host může aplikace běžet například na počítačích na prodejně nebo na skladě, kde do systému budou mít přístup všichni zákazníci, ale pouze v režimu prohlížení a jen některých tabulek (tříd).




**Obrázek 12 - Diagram realizace případu užití z webové aplikace Creately**

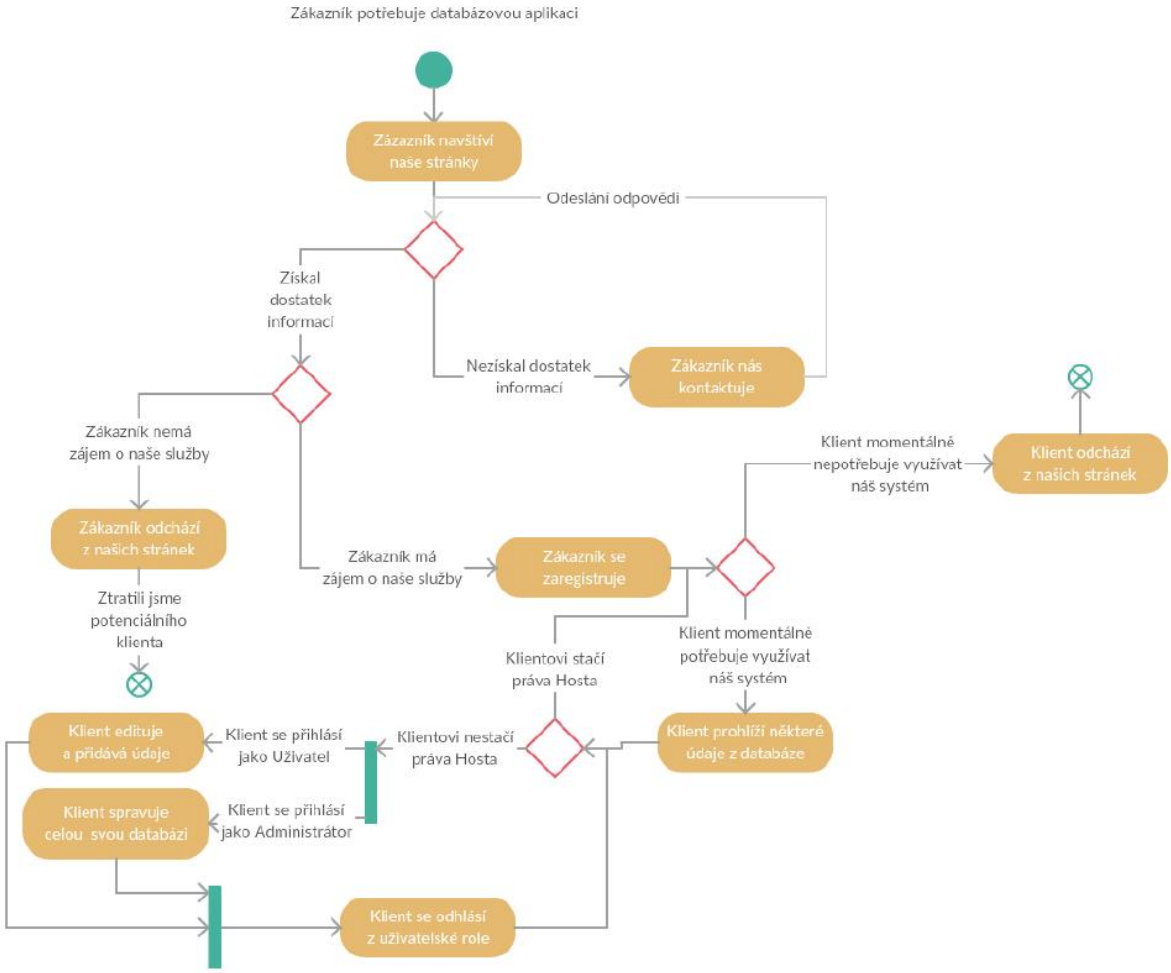
*Zdroj: Vlastní zpracování*

Dalším krokem analýzy je vytvoření diagramu aktivit, který umožňuje modelování procesu jako aktivity, která se skládá z kolekce uzlů spojených hranami. Diagram by měl být přehledný a měl by obsahovat jen minimum nezbytných informací. Tyto diagramy je možné kromě řízení aktivit použít i například k modelování obchodních (podnikatelských) procesů a pracovních postupů.

Tyto diagramy se modelují v UML 2 a v této verzi mají sémantiku založenou na Petriho sítích. Tato nová sémantika má dvě hlavní výhody:

- Větší přizpůsobivost při modelování různých typů cest.
- Zřetelné odlišení diagramů aktivit od stavových automatů.

Diagram aktivit pro autorovu práci obsahuje popis aktivity chování zákazníka na stránkách webové aplikace. Zákazník se nejprve dostane do kontaktu s úvodní stránkou, kde jsou poskytnuty základní informace, na základě kterých se může rozhodnout, zda využije naše služby, či nikoli. Struktuře stránek a aplikace se budou věnovat další kapitoly.




**Obrázek 13 - Diagram aktivit z webové aplikace Creately**

*Zdroj: Vlastní zpracování*

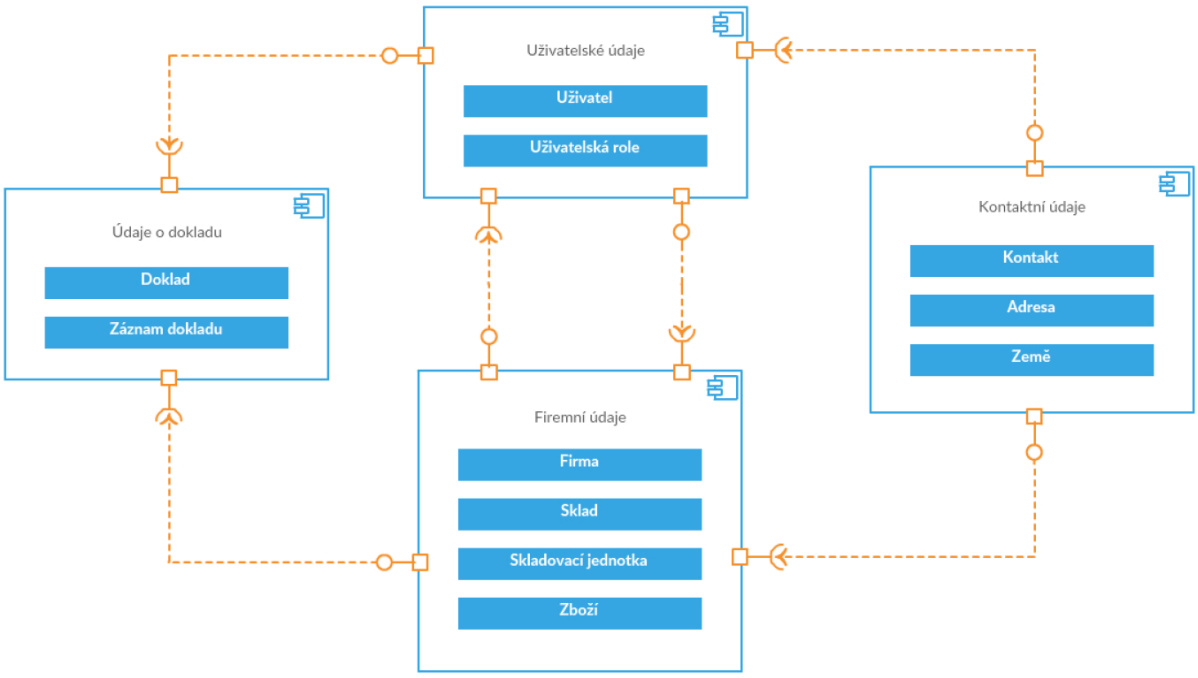
Kromě výše zmíněných částí analýzy můžeme doplnit ještě pokročilé realizace případů užití a pokročilé diagramy aktivit. Pokročilé realizace případů užití jsou založené na výskytu interakcí, které řeší opakující se posloupnosti zpráv. Tyto interakce je možné parametrizovat, takže je možné při každém výskytu interakci předat jiné hodnoty. Pokročilé diagramy aktivit obsahují například spojky, přerušitelné oblasti aktivit, ošetření výjimek, chráněné uzly, přídavné uzly, signály nebo multiplexní vysílání a příjem objektů.

## 5.2 Návrh

K návrhu dochází v poslední etapě fáze rozpracování a v první polovině fáze konstrukce. Velký důraz je zde kladen na požadavky a na analýzu, u které dochází k postupnému upřesňování. Oproti analýze, která je zaměřena na funkce, jež má systém poskytovat, smyslem návrhu je přesná specifikace způsobu implementace těchto funkcí. Na implementaci se lze dívat, jak z pohledu problémové domény, ze které přicházejí požadavky, tak z pohledu domény řešení, která obsahuje například knihovny tříd nebo mechanismy persistence. Výsledkem návrhu by měl být model systému, který lze skutečně implementovat.

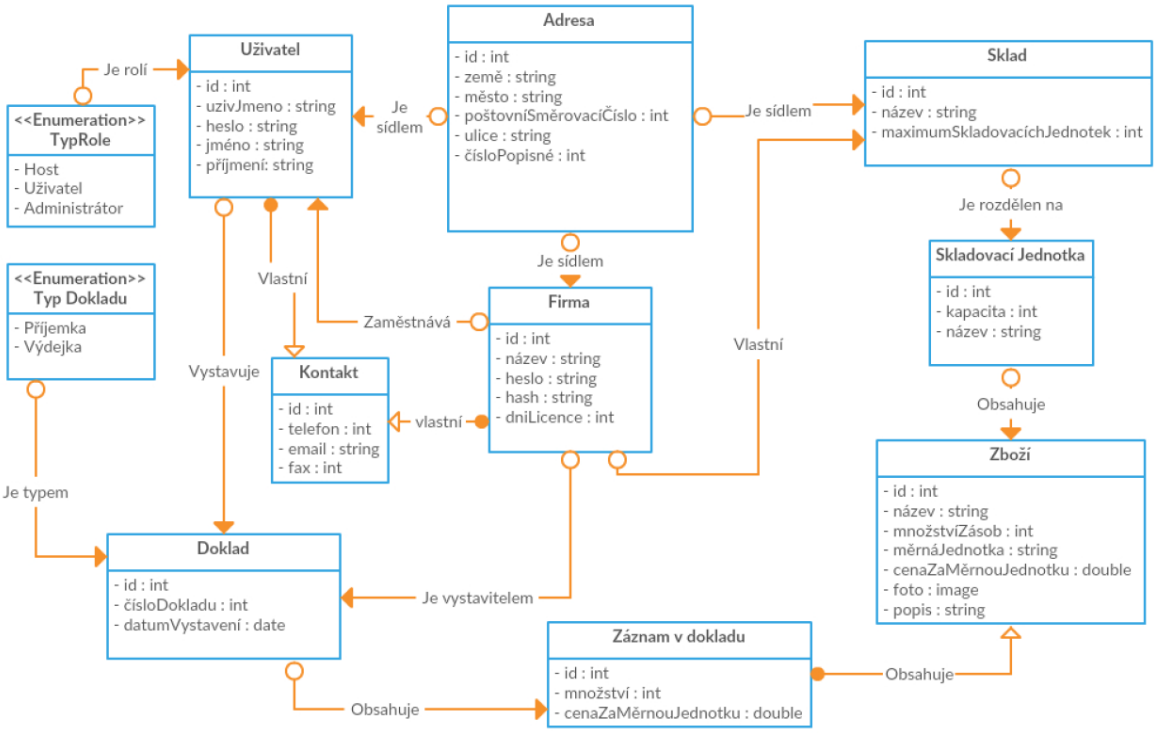
Návrhový model je založen na modelu analytickém a lze jej tedy považovat za upřesnění rozpracovaného analytického modelu s doplněním podrobností a konkrétních technických řešení. Klíčovým artefaktem vytvářeným ve fázi návrhu jsou rozhraní, které slouží ke spojení subsystémů. V návrhu je rovněž modelována první verze diagramu nasazení, který ukazuje rozmístění softwarového systému na jednotlivých fyzických výpočetních uzlech.

Prvním krokem je tedy předělání analytického metamodelu do návrhového metamodelu. Je možné po předělání uchovat oba diagramy, ale většinou se nechává jen nový diagram komponent, neboť se jedná o upřesnění diagramu tříd. Pro upřesnění a hlavní realizaci rozhraní je dobré vytvořit i diagram komponent.




**Obrázek 14 - Diagram komponent z webové aplikace Creately**

*Zdroj: Vlastní zpracování*



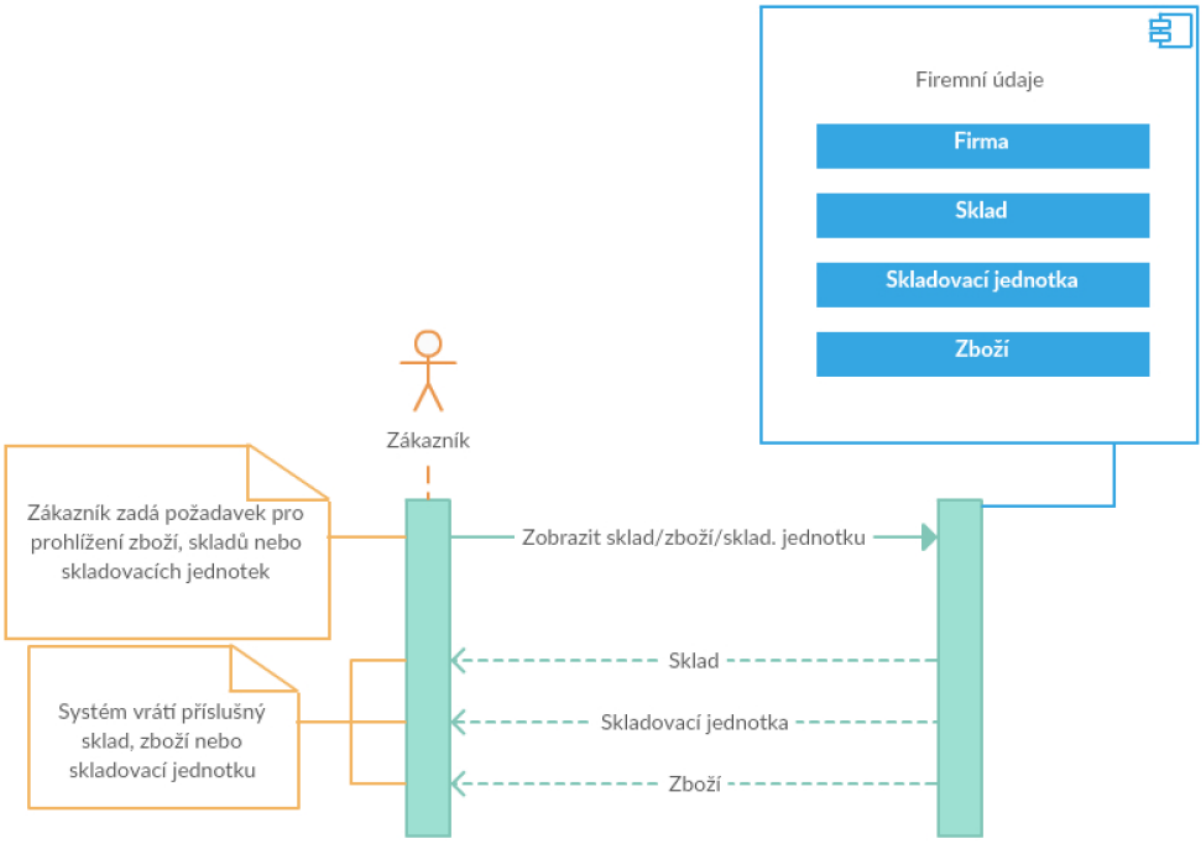

**Obrázek 15 - Návrhový metamodel (diagram tříd) z webové aplikace Creately**

*Zdroj: Vlastní zpracování*

Z modelu je patrné, že oproti analytickému metamodelu obsahuje návrhový větší množství informací o atributech (příznak veřejnosti a datový typ). Rovněž by se zde mělo nacházet větší množství informací o metodách, kdyby se jednalo o typické třídy v objektově orientovaných programovacích jazycích. Jelikož však autor bude uchovávat veškeré informace v databázi, metody se zde nenachází. Je zde pouze příznak budoucích tabulek a sloupců tabulek v databázi.

Dalším krokem je návrh případů užití, kde místo analytických tříd využíváme třídy návrhové, rozhraní a komponenty. Realizace případů užití v návrhu specifikuje implementační rozhodnutí a implementuje nefunkční požadavky. Skládá se z návrhových diagramů interakce a diagramů tříd obsahujících zúčastněné návrhové třídy. Vzhledem k tomu, že model případů užití je vytvářen za účelem lepšího pochopení vznikajícího systému, měl by být objem prací redukován na užitečné minimum (strategický návrh).

Diagramy interakce mohou být buď upřesněním klíčových analytických návrhů diagramů interakce rozšířených o implementační detaily, nebo zcela novými diagramy vytvořenými pro objasnění otázky vzniklé během návrhu. V případě autorovy práce si probereme první případ užití a doplníme jejich model přidáním konkrétních tříd (nebo celých komponent), které zajišťují dané užití a určitou událost.

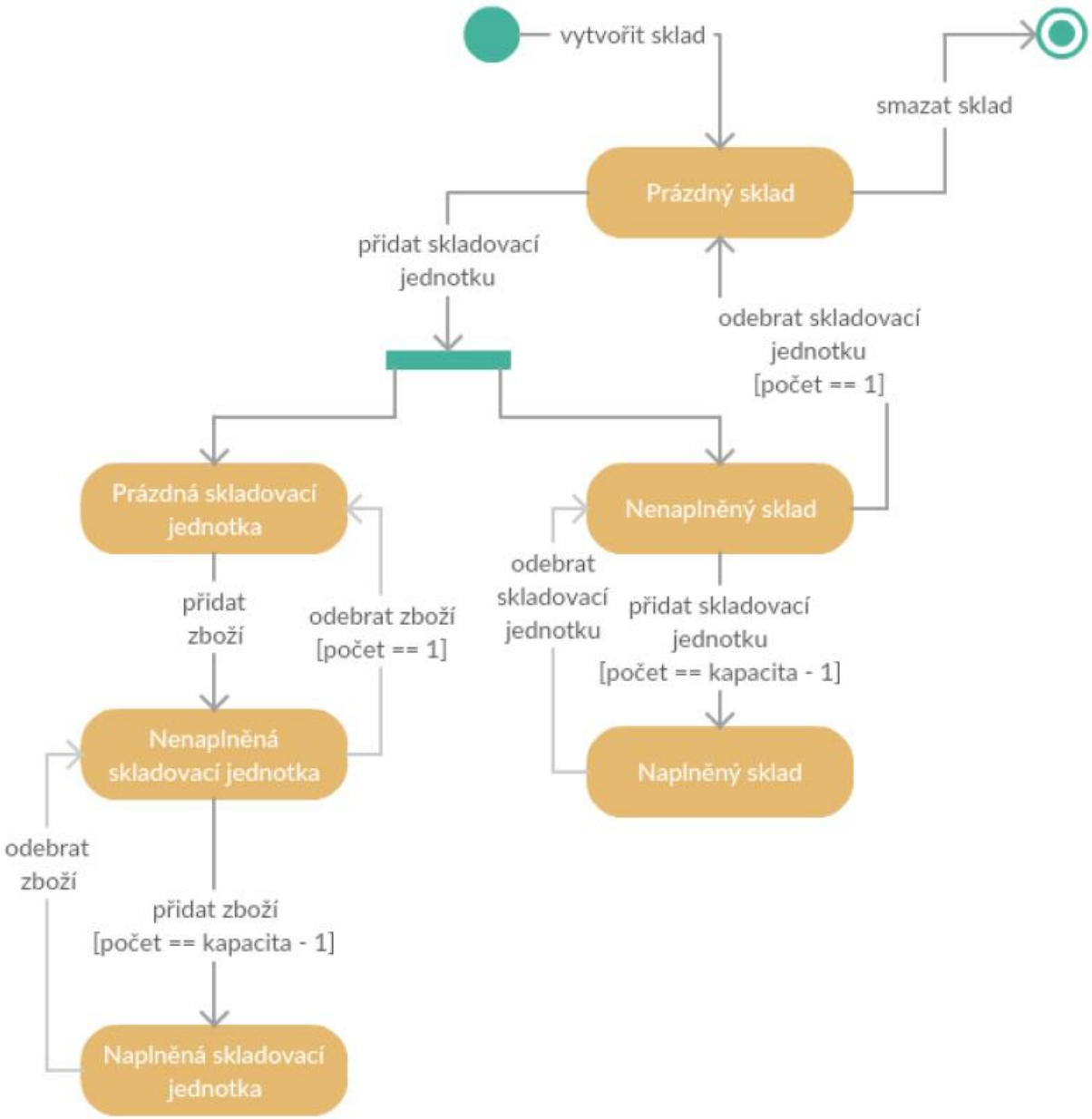



**Obrázek 16 - Diagram interakce pro obsluhu zákazník z webové aplikace**

*Creately. Zdroj: Vlastní zpracování*

Již zde je vidět, že oproti původnímu analytickému modelu případu užití je zde daleko více informací, které už naznačí způsob implementace problematiky. Přímo je zde patrné, jaké třídy budou využity (dokonce celá jedna komponenta). V této části návrhu je také vhodné uvažovat souběžnost více aktivit. Alternativou k využitému diagramu interakce je sekvenční diagram. V diagramech této části návrhu je vhodné zahrnout interakci podsystémů (viz využití podsystému Firemní údaje).

Stejně jako předchozí dvě, i třetí část vytvořená v analýze je rozvíjena ve fázi návrhu. Na diagram aktivit, který modeluje aspekty dynamického chování systému, navazují stavové automaty. Na rozdíl od diagramů aktivit, které vycházejí z Petriho sítí a používají se k modelování obchodních procesů, stavové automaty jsou založeny na Harelově práci a používají se k modelaci historie životního cyklu jednoho reaktivního prvku jako stavového automatu.




**Obrázek 17 - Stavový automat z webové aplikace Creately**

*Zdroj: Vlastní zpracování*

Kromě výše zmíněných částí fáze návrhu je možné ještě další modely a návrhové prvky dodělat. Mezi takové patří například pokročilé stavové diagramy, které na rozdíl od využitých stavových automatů využívají například:

- Složené stavy (jednoduché a ortogonální).
- Podautomaty (jejich stavy a vzájemnou komunikaci).

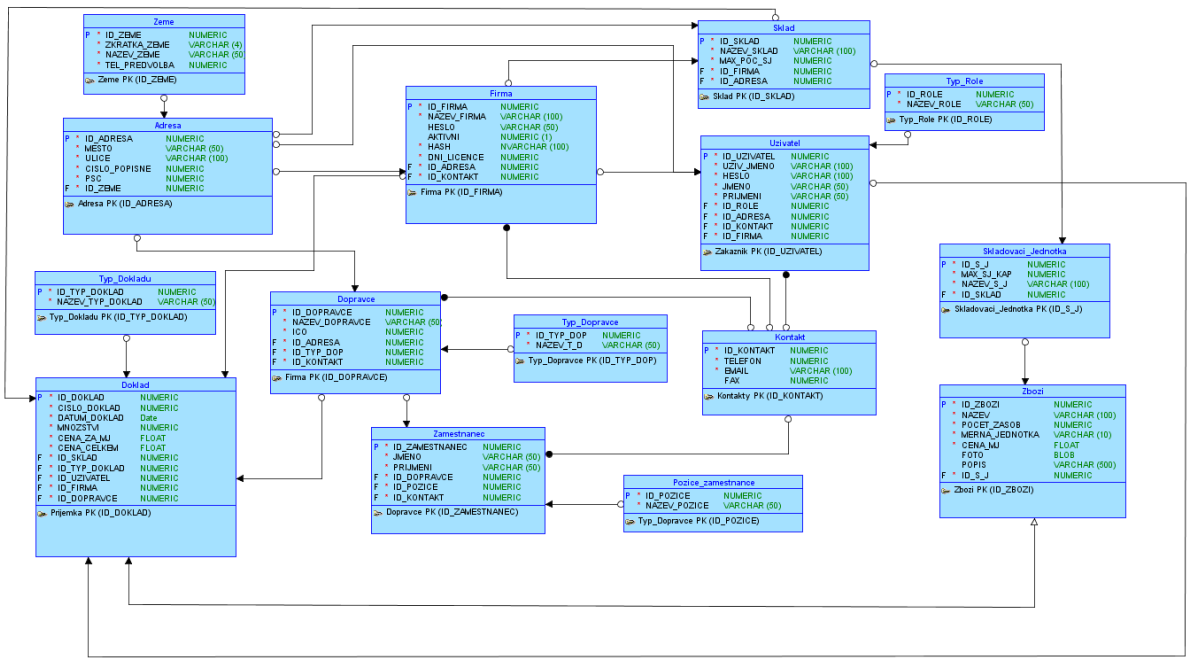
# 6 IMPLEMENTACE A POPIS FUNKCÍ APLIKACE

Tato kapitola je zaměřena na popis implementace, a především na popis hlavních funkcí webové aplikace pro správu skladového systému. Část zaměřená na implementaci navazuje na předchozí kapitolu, neboť je to fáze, následující po fázi návrhu.

## 6.1 Implementace

Implementace spočívá v převodu návrhového modelu do spustitelného kódu. Vedlejším produktem této aktivity může být implementační model, který může v některých případech být využit ke zpětné analýze. Implementační model může být i součástí návrhového modelu, jedná se totiž o implementační pohled na návrhový model. Hlavním účelem je konkretizovat, jak se návrhové prvky projevují pomocí artefaktů, a jak jsou tyto artefakty nasazovány na uzly. Artefakty v tomto smyslu značí cokoli skutečného, jako třeba zdrojové soubory, zatímco uzly označují hardware, do něhož jsou artefakty nasazovány [29].

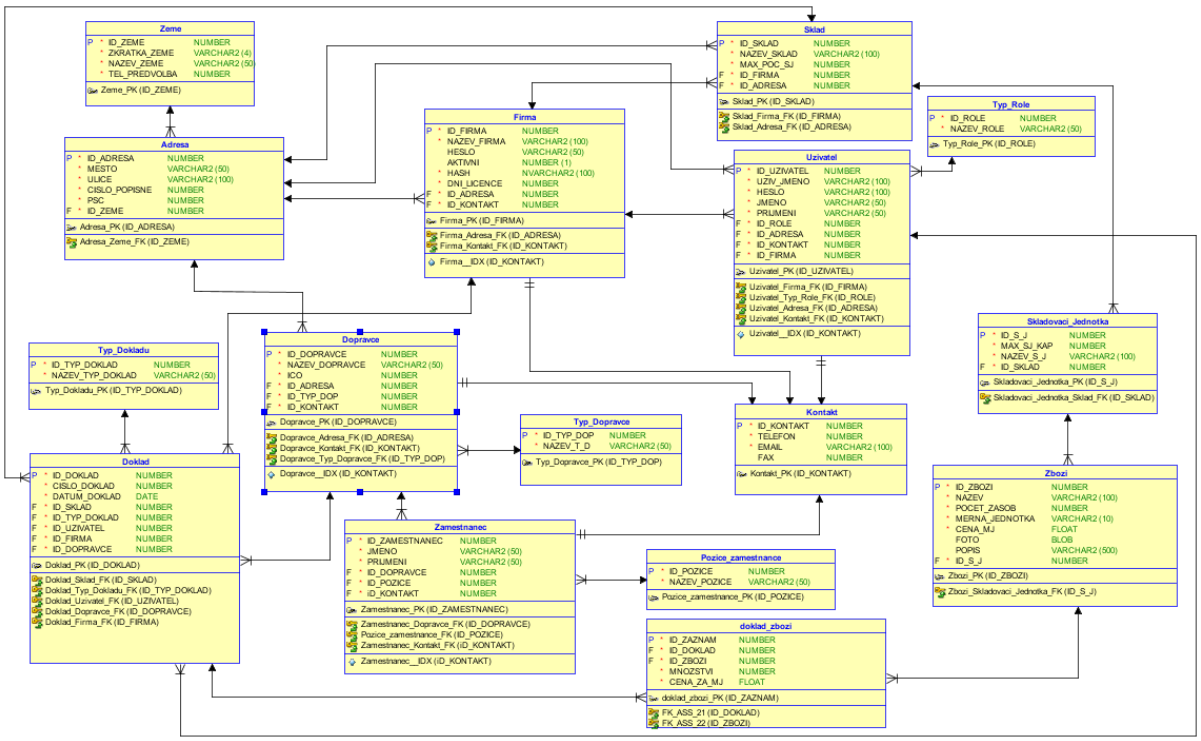
V případě autorovy webové aplikace pro správu skladového systému byla prvním krokem implementace tvorba databázové struktury. Třídy, které byly vymodelovány během fáze analýzy a doplněny o implementační informace ve fázi návrhu, autor v této fázi převedl do databázových tabulek a atributy těchto tříd do sloupců tabulek. Z diagramů tříd tedy nejprve vytvořil logický databázový model. K tomuto je vhodné použít Oracle SQL Developer, neboť si autor k implementaci databáze zvolil Oracle SQL.




**Obrázek 18 - Logický databázový model (Bachmanova notace) z Oracle SQL Developer**

*Zdroj: Vlastní zpracování*

Bachmanova notace, kterou autor využil při tvorbě logického databázového modelu má výhodu, že na rozdíl od Barkerovy notace zobrazuje mimo jiné i datové typy sloupců a cizí klíče tabulek. Datové typy je vhodné nastavit už při tvorbě logického datového modelu, neboť když se vyskytne chyba a bude potřeba předělat logický model, není nutné pokaždé nastavovat datové typy v relačním modelu. Z logického databázového modelu následně autor udělal model relační, který především řeší relace více ku více (M:N) a to tvorbou průnikové tabulky.




**Obrázek 19 - Relační databázový model z Oracle SQL Developer**

*Zdroj: Vlastní zpracování*

U automaticky vytvořeného relačního modelu je nutné zkontrolovat všechny vazby a názvy cizích klíčů. Většinou se najde minimálně jeden cizí klíč, jehož název je moc dlouhý, což by při generování DDL skriptu vyvolalo chybu. Pro přehlednost je rovněž vhodné srovnat relační databázový model tak, aby byl přehledný, tedy přesunout některé tabulky, aby se co možná nejméně vazeb křížilo. Poté, co autor všechny potenciální chyby ošetřil, vygeneroval DLL skript, který posléze použil pro tvorbu tabulek v databázi.

Diagram tříd byl tedy „převeden“ nejprve do logického, později do relačního databázového modelu, a nakonec byl použit k tvorbě tabulek v databázi. Tento diagram byl první částí obou z fází analýzy a návrhu, následovaly případy užití a diagramy aktivit. Tyto dvě části byly implementovány ve webové aplikaci, která využívá data z databáze. Částečně implementaci, ale především popisu funkcí webové aplikace se bude věnovat následující podkapitola.

## 6.2 Popis funkcí

Tato podkapitola bude zaměřená na popis realizace webové aplikace, především pak na popis všech funkcí. Webová aplikace byla vytvářena a zkoušena na školním serveru, kde bohužel byla zablokovaná jakákoli možnost odesílání emailů. Ačkoli autorova práce obsahuje hned několik prvků, ve kterých je využité odesílání emailů, nebylo možné z výše uvedených důvodů tuto funkčnost patřičně otestovat.

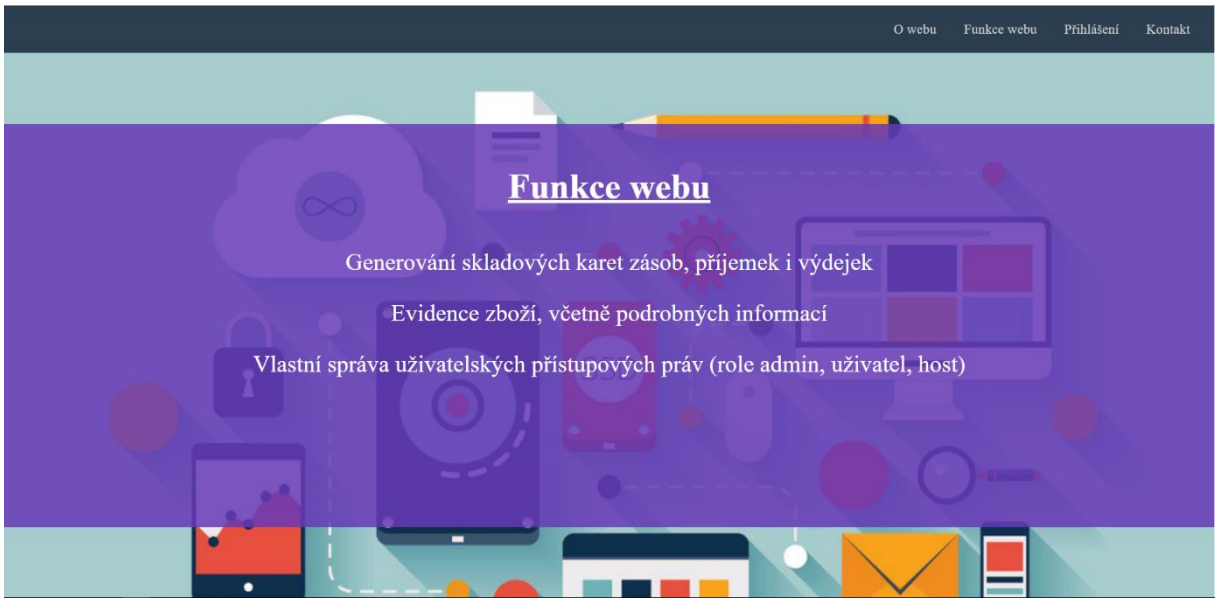
### 6.2.1 Úvodní stránka

Základ aplikace tvoří index, který se skládá ze 4 karet, z čehož dvě mají čistě informační účel a dvě vlastní funkčnost. Mezi kartami je možné přecházet buď rolováním, nebo pomocí navigačního menu, které je umístěné v horní části stránky.


**Obrázek 20 - První karta indexu s informacemi o webu**

*Zdroj: Vlastní zpracování*

Na této kartě se nachází stručný popis autorovy aplikace, který je neměnný a staticky nastavený. V horní části je rovněž vidět zmíněný navigační panel. Tento panel vždy zůstává viditelný a na stejné pozici, během rolování celou stránkou.

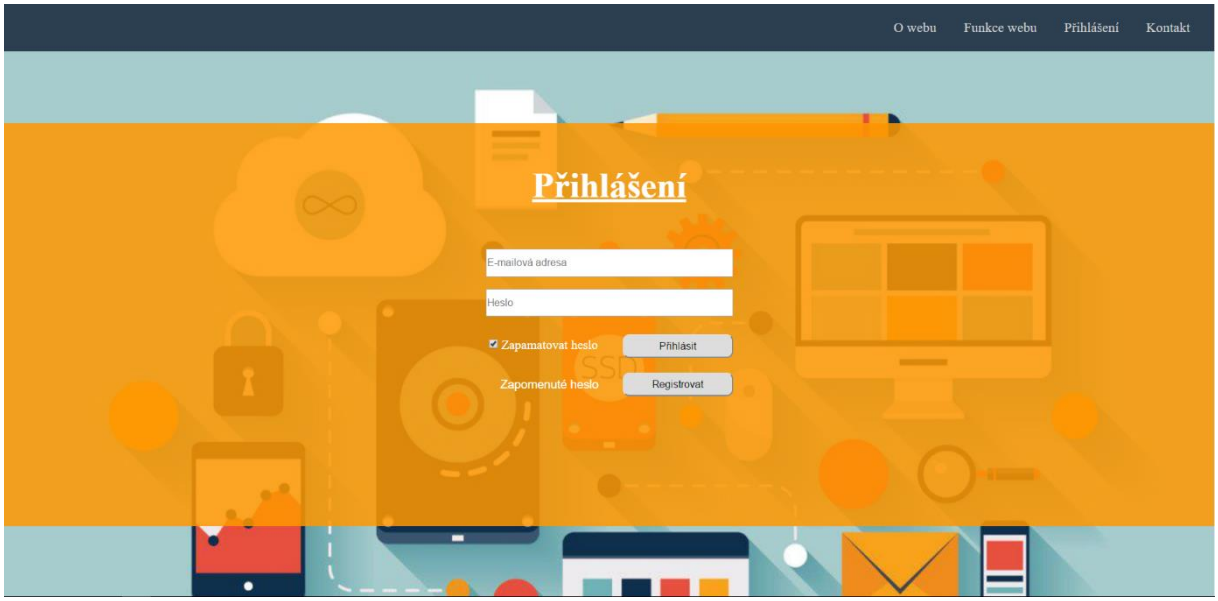



**Obrázek 21 - Druhá karta indexu s přehledem funkcí webové aplikace**

*Zdroj: Vlastní zpracování*

Na druhé kartě se nachází stručný soupis funkcí webu. Obsah této karty je rovněž neměnný.

Následují dvě karty, které již obsahují dynamické prvky. Obě karty obsahují formuláře, které mají vlastní funkčnost.




**Obrázek 22 - Třetí karta indexu s formulářem pro přihlášení**

*Zdroj: Vlastní zpracování*

Karta s formulářem pro přihlášení obsahuje hned několik prvků. Prvním viditelným je obyčejné přihlášení, kde se zadané přihlašovací údaje kontrolují s údaji, uloženými v databázi. Ačkoli

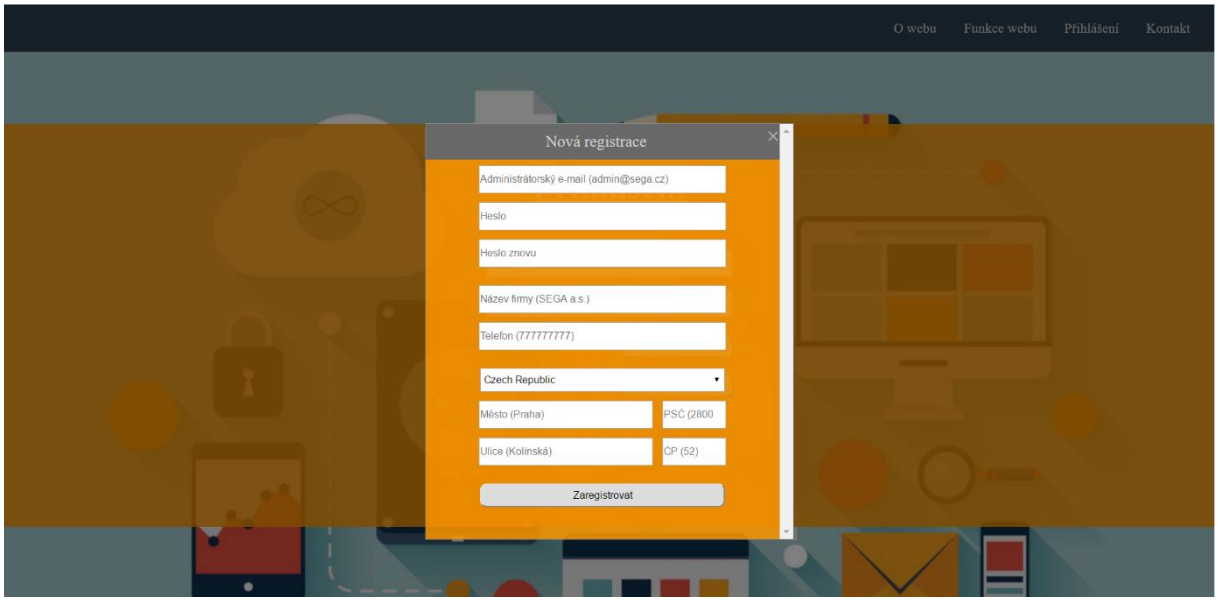
jsou hesla v databázi uložena ve formě md5 hashového kódu, je zde ještě stále mezera v zabezpečení, jelikož heslo z formuláře je posíláno do databáze ve formě textu.

Heslo je tedy jako text odesláno do databáze, konkrétně do funkce balíčku SECURITY.AUTENTIFIKACE, která zahashuje příchozí heslo a porovná ho s hashem z databáze, který odpovídá zadanému přihlašovacímu emailu. Pokud se hashe rovnají, vrátí tato funkce ID firmy, pro kterou byly zadány správné přihlašovací údaje, pokud se nerovnají, vrátí hodnotu 0.

Pokud výsledek databázové funkce pro autentifikace není nulový, inicializuje se relace a nastaví se relační proměnná id_firmy na hodnotu z této funkce a přes nový dotaz se zjistí id hosta této firmy. Po nastavení relačních proměnných dojde k přesměrování na stránku systému, kde již uživatel může procházet a spravovat své tabulky.

Dále jsou zde dvě zatím jen předpřipravené funkcionality – zapamatovat heslo a zapomenuté heslo. Vzhledem k faktu, že autor realizovat tuto webovou aplikaci na školním serveru, ze kterého je zakázáno posílat emaily, zapomenuté heslo nemohlo být odzkoušeno (stejně jako dalších několik funkcionalit).

Posledním prvkem karty přihlášení je tlačítko registrovat. Po stisknutí tohoto tlačítka dojde k otevření nového modalu, který obsahuje registrační formulář.




**Obrázek 23 - Registrační formulář**

*Zdroj: Vlastní zpracování*

Tento formulář autor vyřešil pomocí tzv. placeholderů (text, který se nachází v jednotlivých polích, pokud jsou prázdná). Tímto způsobem jsou řešena všechna pole, kromě výběru státu. Výběr státu je řešen pomocí roletky (combo boxu), který obsahuje data importovaná z databáze a ve výchozím stavu je zvolena Česká republika. Dále pole, která mají obsahovat pouze číselné hodnoty (telefon, poštovní směrovací číslo a číslo popisné) jsou typu „number“, tudíž do nich není možné zadat jinou než číselnou hodnotu.

Všechna pole v tomto formuláři jsou nastavena jako nutná, takže pokud kterékoli zůstane nevyplněné a uživatel se pokusí formulář odeslat, bude upozorněn, které pole ještě musí vyplnit. Pokud budou všechna pole vyplněna, dojde k odeslání a následné kontrole formuláře. První prochází kontrolou zadaná emailová adresa, pomocí funkce FILTER_VALIDATE_EMAIL. Pokud nebude mít správný formát, vypíše to chybu a znovu otevře dialog pro registraci.

Druhá kontrola je zaměřená na hesla. Možným nedostatkem zde by mohla být absence kontroly složitosti hesla, neboť v současném stavu projde bez problémů i heslo „heslo“. Kontrola, která se zde nachází, je zaměřena pouze na shodu dvakrát zadávaného hesla, které je zde pro případ překlepu, aby uživatel opravdu nastavil heslo, které zamýšlel. Při odlišných heslech nastane stav stejný jako při špatně zadané emailové adrese. Všechny informace (kromě hesla) zůstanou uložené ve formuláři, takže není nutné vše vyplňovat znovu.


**Obrázek 24 - Výsledek registrace po zadání emailu ve špatném formátu**

*Zdroj: Vlastní zpracování*

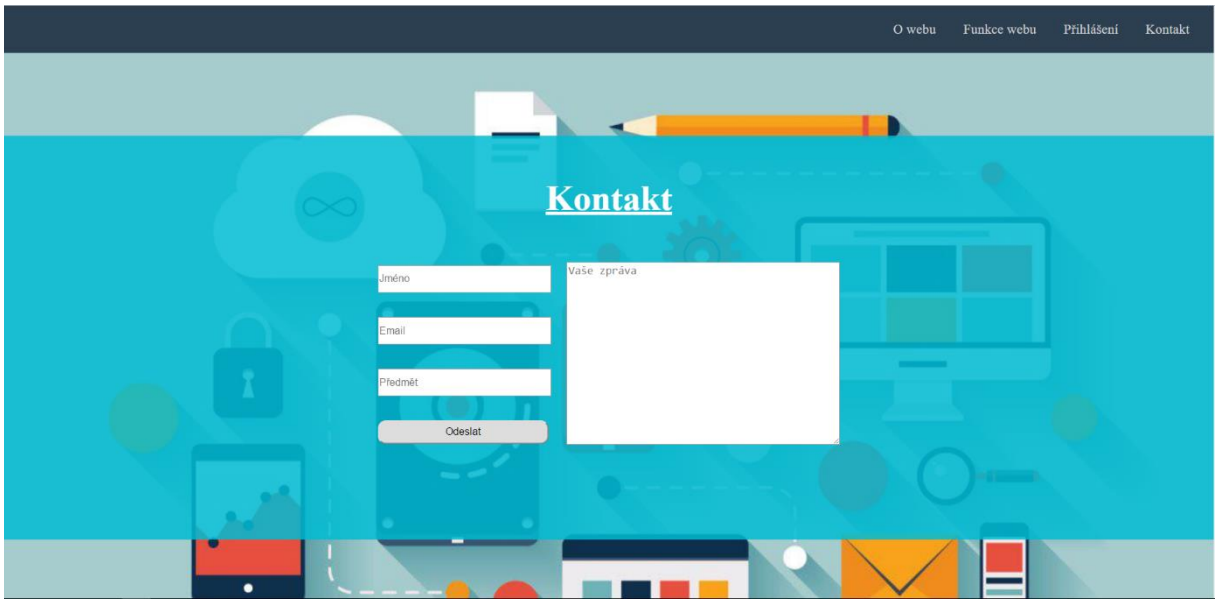
Další kontrola je na duplicitu emailových adres. Tato kontrola probíhá v databázové proceduře PRIDANI_FIRMY. Zde se opět vyskytuje bezpečnostní nedostatek, jelikož heslo je do

databáze posíláno ve formě textu. Zmíněná procedura zkontroluje, zda v databázi neexistuje záznam s uvedeným emailem, pokud ano, nahlásí chybu. Pokud tento email ještě nebyl použit, postupně vytvoří záznam adresy, záznam kontaktu a následně záznam firmy, který bude spojený se dvěma vytvořenými záznamy (adresou a kontaktem).

Po registraci je opět implementován potvrzovací email, který má být odeslán na emailovou adresu, která byla zadána při registraci. Tento email obsahuje odkaz, na který když uživatel klikne, bude přesměrován na stránku aplikace, která obsahuje skript pro změnu sloupce v databázi „aktivní“ z hodnoty 0 na hodnotu 1. Tato funkčnost ale opět nemohla být dostatečně odzkoušena, neboť autor neměl právo posílat ze školního serveru jakýkoli email.

Zároveň se po úspěšné registraci vytvoří dva záznamy v tabulce uživatelů pro příslušnou firmu. Vytvoří se uživatel s uživatelskou rolí administrátora a jeden uživatel s uživatelskou rolí host. Výchozí nastavení nastavuje uživatelské jméno i heslo pro administrátora admin a admin a pro hosta host a host.

Následuje karta kontaktu. Jak bylo již zmiňováno ve fázích analýzy a návrhu, informovanost potenciálního zákazníka je velmi podstatná. Proto když nedostane potřebné informace z webových stránek aplikace, je potřeba aby měl možnost kontaktovat vývojáře aplikace se svým dotazem. Od toho je tu tato karta, která obsahuje formulář, do kterého návštěvník stránky zadá své jméno, email, předmět a text zprávy a zpráva bude odeslána na emailovou adresu podpory webové aplikace.



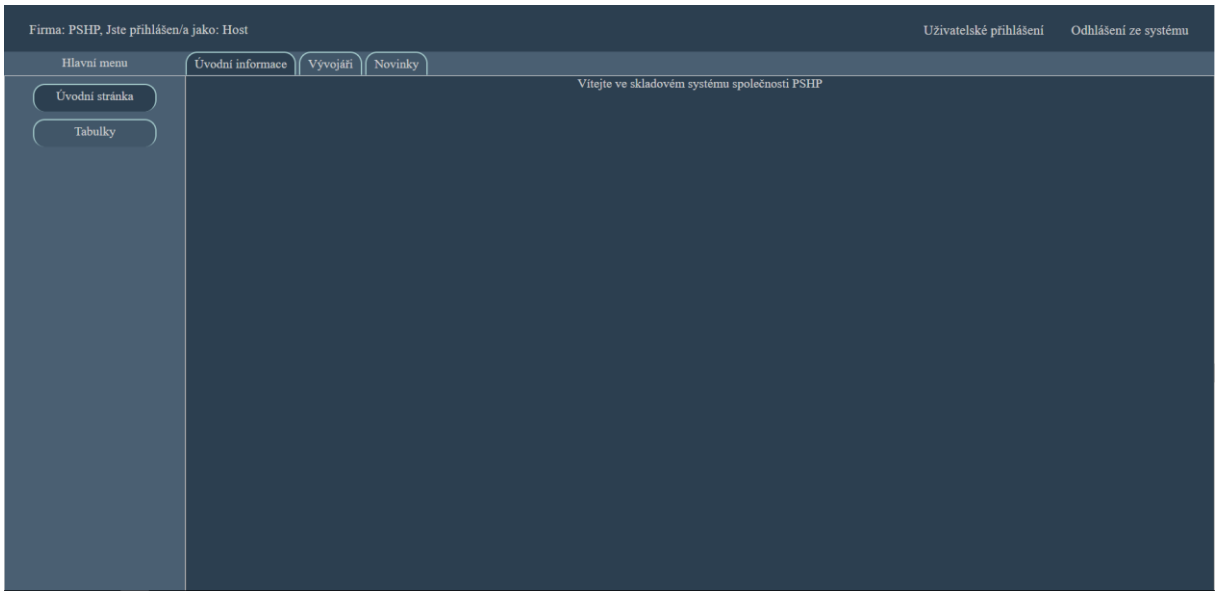

**Obrázek 25 - Čtvrtá karta indexu s kontaktním formulářem**

*Zdroj: Vlastní zpracování*

Jak však již bylo zmíněno dříve, na školním serveru není umožněno studentům vytvořit funkční odesílání emailu. Přestože je tato funkčnost naimplementovaná, na školním serveru nemohla být dostatečně odzkoušena.

### 6.2.2 Systém v režimu Host

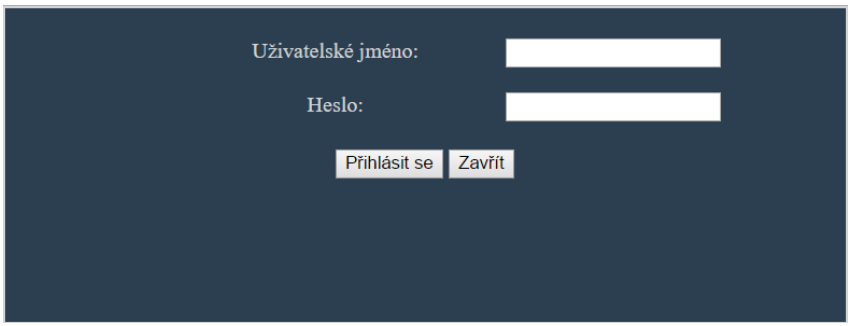
Po přihlášení do systému se uživatel dostane do samotné webové aplikace pro správu skladového systému. Ve výchozím stavu hned po přihlášení se uživatel nachází v režimu s uživatelským oprávněním hosta. Tento režim povoluje jen minimum operací, především procházení a prohlížení určitých tabulek. Uživatelské rozhraní systému je rozděleno na 3 části.




**Obrázek 26 - Webová aplikace po přihlášení v režimu host**

*Zdroj: Vlastní zpracování*

Horní panel stránky je rozdělen na dvě části. Levý panel horní části obsahuje dynamickou textovou část, ve které se nachází informace o firmě, k jejímž systému je uživatel přihlášen a o uživatelské roli, na které se aktuálně uživatel nachází. V pravé části se pak nachází možnost uživatelského přihlášení do systému (změna uživatelské role na uživatele nebo administrátora) a tlačítko pro odhlášení ze skladového systému firmy, které smaže relační proměnné, na kterých stojí celá aplikace, zruší relaci a přesměruje uživatele na úvodní stránku (index).




**Obrázek 27 - Uživatelské přihlášení**

*Zdroj: Vlastní zpracování*

V levé části se nachází hlavní menu. Počet zobrazených prvků je závislý na uživatelské roli. Host vidí jen nezbytné minimum, tedy úvodní stránku a tabulky. Po kliknutí na tlačítko tabulek se zobrazí nabídka konkrétních tabulek k zobrazení, kde opět host vidí jen některé vybrané.


**Obrázek 28 - Hlavní menu pro uživatelskou roli Host**

*Zdroj: Vlastní zpracování*

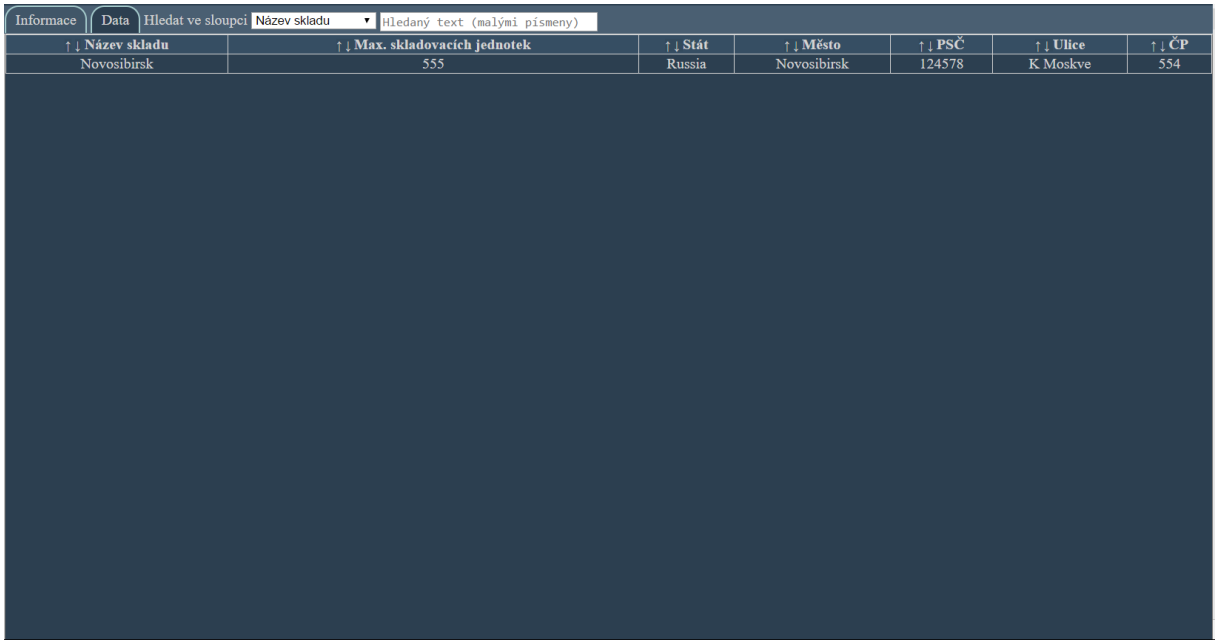
Poslední část stránky zahrnuje nabídku karet a hlavní zobrazovací část. Na úvodní stránce se nachází 3 karty:

- Úvodní informace.
- Vývojáři.
- Novinky.

Po rozkliknutí každé ze zmíněných karet se změní obsah zobrazovací části, v tomto případě na stručné informace, které se ještě dají doplnit. Na stránce tabulek se nachází pouze dvě karty – Informace o tabulkách a Popis funkcí. Tyto karty neobsahují jen velmi stručné informace, které jsou statické a je možné je kdykoli doplnit. Důležitější jsou karty, ze kterých uživatel dostane

na výběr po rozkliknutí konkrétní tabulky (Sklady, Skl. jednotky a Zboží). Jedná se o karty Informace a především Data.

Po kliknutí na konkrétní tabulku se zobrazí karta Data, která obsahuje výpis z databáze. Zároveň pokud je aktivní karta Data, zobrazí se vedle karet i roletka a textové pole pro vyhledávání. Roletka slouží k výběru sloupce, ve kterém má být hledána hodnota a textové pole funguje dynamicky – jakmile se změní jeho obsah, vyhledává zadanou hodnotu ve vybraném sloupci. Data z databáze, která jsou vypsána z databáze, je rovněž možné řadit podle všech vypsaných sloupců pomocí šipek vedle názvu sloupce.



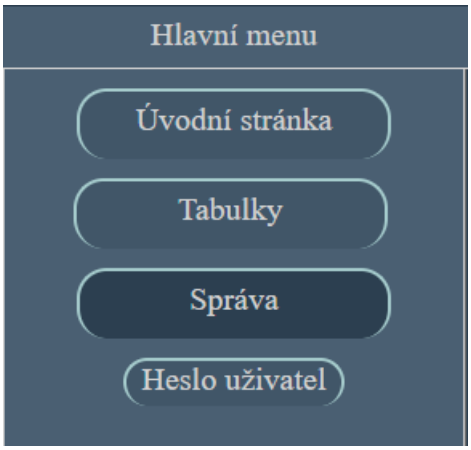

**Obrázek 29 - Zobrazení dat tabulky pro uživatelskou roli Host**

*Zdroj: Vlastní zpracování*

Host však má pouze omezené možnosti práce se systémem, proto se zde nenachází žádná možnost úpravy, přidání ani smazání. Tato uživatelská role je zamýšlena například na pobočku firmy nebo na sklad, kde bude počítač s touto webovou aplikací v uživatelském režimu Host, ve kterém bude moci kterýkoli zákazník vyhledávat zboží.

### 6.2.3 Systém v režimu Uživatel

Po uživatelském přihlášení do účtu s uživatelským oprávněním uživatele se odemkne hned několik nových funkcionalit. Všechny karty, které obsahují pouze statický text (jako například Vývojáři, Informace o tabulkách nebo Popis funkcí) zůstávají stejné. Co se mění jsou však samotné tabulky s daty a oprávnění pro práci s nimi. Nově se navíc v hlavním menu odemkne možnost Správa s jedinou možností – Heslo uživatel pro změnu uživatelského hesla.




**Obrázek 30 - Hlavní menu s možností změny uživatelského hesla**

*Zdroj: Vlastní zpracování*

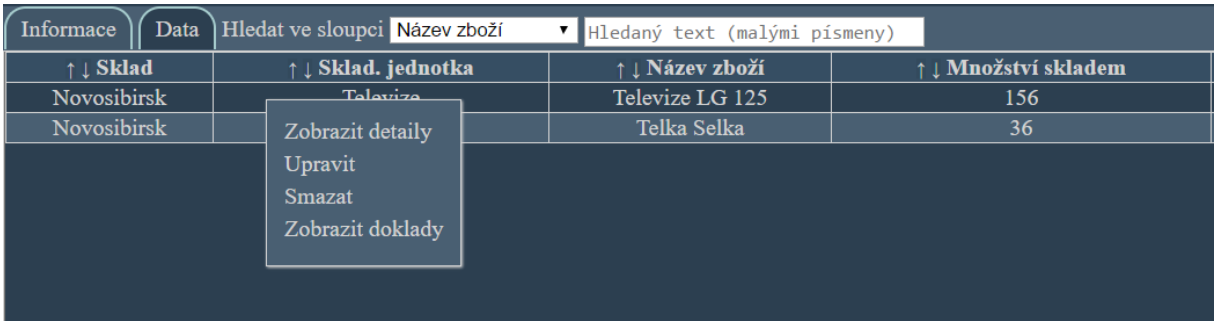
Jelikož uživatelem je v tomto systému myšlen zaměstnanec firmy (skladu), odemkne se i pár dalších tabulek a u všech tabulek se odemkne základní uživatelská práce s tabulkami.


**Obrázek 31 - Hlavní menu a tabulky dostupné pro Uživatele**

*Zdroj: Vlastní zpracování*

Nově se tedy odemknou tabulky Doklady a Dopravci. Do těchto tabulek logicky neměl host přístup, protože to je firemní záležitost. U hosta šlo vyloženě o prohlížení zboží, případně skladů a skladovacích jednotek. Uživatel má však už za úkol správu skladového systému, proto by měl mít přístup ke všem potřebným tabulkám. Navíc je potřeba nějak umožnit uživateli práci s tabulkami (přidávat zboží, upravovat skladovací jednotky atp.), proto aplikace implementuje

vlastní kontextové menu, zvlášť pro kliknutí na řádek tabulky a zvlášť pro kliknutí mimo tabulku.




**Obrázek 32 - Uživatelská kontextová nabídka pro řádek tabulky**

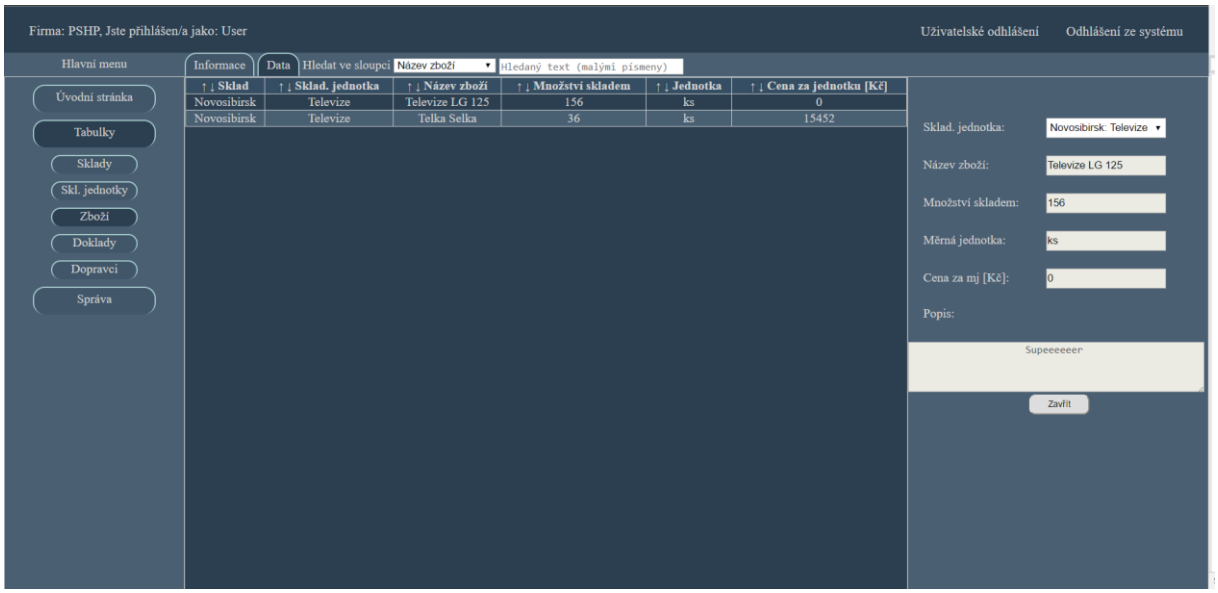
*Zdroj: Vlastní zpracování*

Každá tabulka obsahuje různé možnosti. Na obrázku 32 je například kontextová nabídka pro tabulku Zboží, která především obsahuje možnosti Zobrazit detaily, Upravit a Smazat. Tohle jsou možnosti, které nabízí skoro každá tabulka pro uživatele. Rozdíl je například u tabulky Sklad, která nenabízí možnosti Upravit ani Smazat pro uživatele (tahle oprávnění má pouze administrátor), ale naopak obsahuje možnosti Zobrazit sklad. jednotky a přidat sklad. jednotku. Podobně je na tom i tabulka Skladovacích jednotek, která umožňuje upravit i smazat skladovací jednotku, a navíc umožňuje vložit a zobrazit zboží.


**Obrázek 33 - Uživatelská kontextová nabídka mimo tabulku**

*Zdroj: Vlastní zpracování*

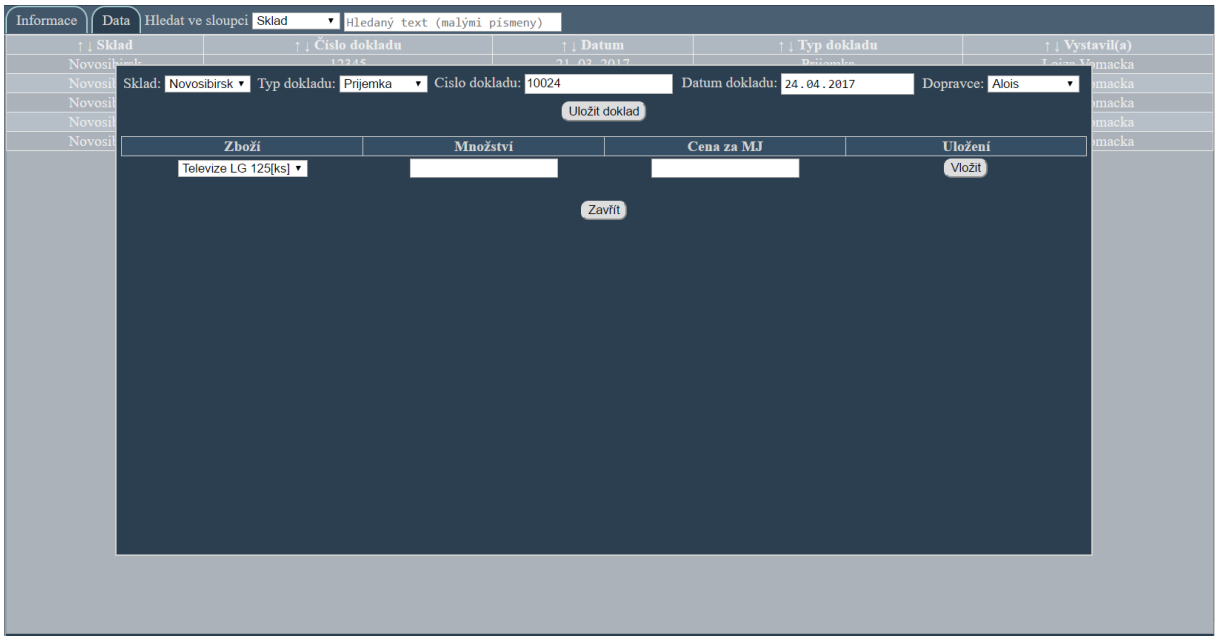
Kontextové menu mimo tabulku obsahuje jen jednu možnost – přidání nového záznamu. Tato možnost je pro uživatele otevřena ve všech tabulkách s výjimkou skladů, ty může přidávat pouze administrátor. Uživatel rovněž nemá právo mazat dopravce, ani mazat a upravovat doklady. Po vybrání jedné z možností zobrazení detailů, přidání nebo úpravy se v pravé části zobrazovací otevře nové okno – formulář se všemi údaji o vybraném řádku.




**Obrázek 34 - Webová aplikace s otevřenými detaily vybraného řádku tabulky**

*Zdroj: Vlastní zpracování*

Pokud bude menu otevřeno možností zobrazit detaily, budou všechna pole formuláře uzamčena – určena jen pro čtení s jedinou možností – zavřít. V případě, že uživatel bude chtít přidat nebo odebrat záznam tabulky, budou pole odemčená. Pro úpravu budou pole vyplněna podle vybraného řádku a pod formulářem přibude tlačítko Uložit změny. Pro přidání nového záznamu budou všechna pole formuláře prázdná a přibude tlačítko Vložit nový. Tento formulář je řešen pro všechny tabulky stejně s jedinou výjimkou – doklady.




**Obrázek 35 - Webová aplikace s dialogem pro přidání nového dokladu**

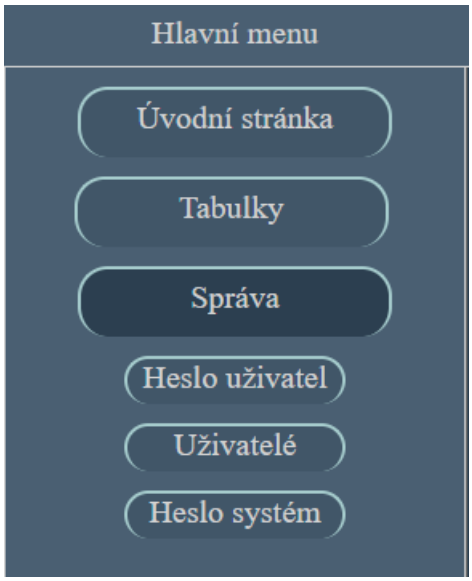
*Zdroj: Vlastní zpracování*

Uživatel může doklady pouze zobrazovat nebo přidávat. Při přidávání je v dialogu nejprve potřeba vytvořit doklad pomocí formuláře v horní části dialogu. Teprve po vytvoření prázdného dokladu se otevře možnost přidávat záznamy dokladu, a nakonec uložit doklad.

Přepnout zpět do uživatelského režimu Host je možné kliknutím na Uživatelské odhlášení, které se nachází v horní části stránky na místě, na kterém původně bylo uživatelské přihlášení.

### 6.2.4 Systém v režimu Administrátor

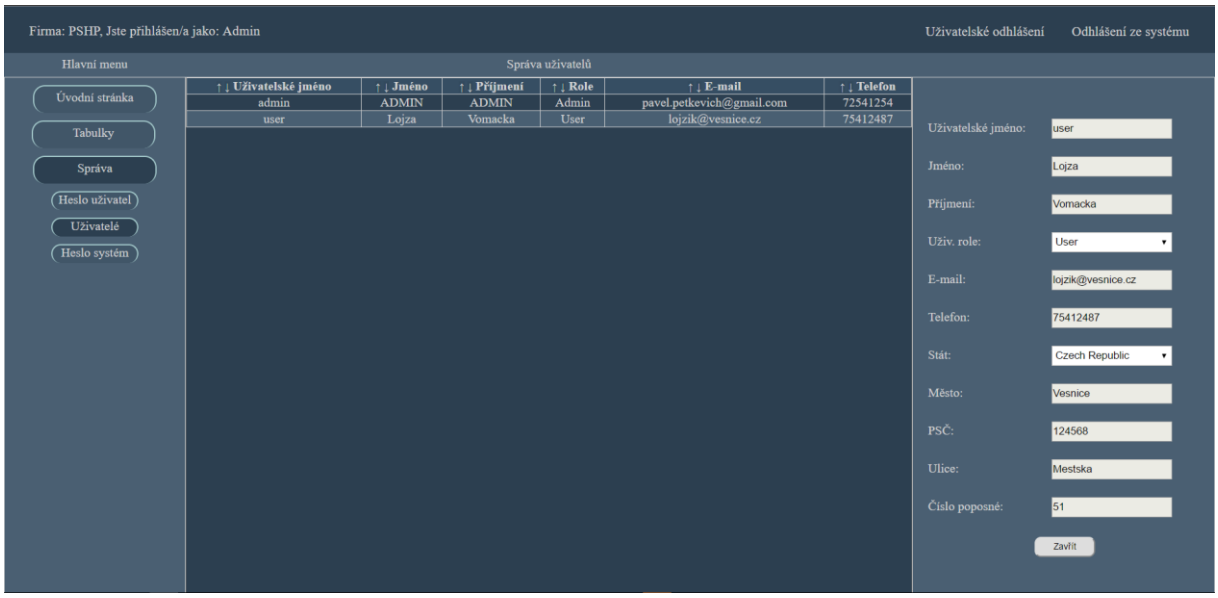
Přihlášením do systému s uživatelskými právy administrátora se odemkne veškerá funkcionalita. Vzhledem k tomu, že už uživatel měl přístup ke všem tabulkám, administrátorovi se už žádné další neodemknou. Na druhou stranu se odemkne veškerá funkcionalita u všech tabulek (přidávání, editace, mazání atp.). Hlavním prvkem, který se administrátorovi zpřístupní je správa účtu společnosti v systému a správa uživatelů.




**Obrázek 36 - Hlavní menu s možností správy pro administrátora**

*Zdroj: Vlastní zpracování*

V sekci Heslo systém je možné změnit heslo, pomocí kterého se uživatel přihlašuje do celého systému společnosti. Po kliknutí na možnost Uživatelé se administrátor dostane do správy uživatelů a jejich uživatelských rolí. Tato sekce je řešena jako všechny ostatní výpisy tabulek z databáze. Stejně jako u ostatních tabulek, i zde funguje stejné kontextové menu a je zde stejná pravá část s formulářem, který obsahuje všechna data z vybraného řádku.




**Obrázek 37 - Správa uživatelů webové aplikace pro administrátora**

*Zdroj: Vlastní zpracování*

# ZÁVĚR

Autor se rozhodl začít objasněním problematiky správy skladů, kde popsal hlavní postupy z oblasti managementu zboží a zásob, především pak metody oceňování zboží a základní popis webové aplikace. Jednalo se o malou vsuvku z jiného oboru, která však nebyla od věci, neboť pomohla k nastínění problému, který tato práce měla řešit. Navíc zde objasnil rozdíl mezi webovou stránkou a webovou aplikací, který by se někomu mohl plést.

Další část práce byla zaměřena na popis technologií pro vývoj webových aplikací a pro návrh a tvorbu databází. Autor nejprve popsal současně nejpoužívanější technologie pro webové aplikace a provedl jejich srovnání. Z těchto technologií si pak na základě vlastního uvážení a nedostatku zkušeností s ostatními technologiemi vybral základní technologie pro návrh a tvorbu webových stránek a aplikací. Tuto svou volbu obhájil a objasnil ve shrnutí.

Podobný postup uplatnil při popisu a porovnávání technologií pro návrh a tvorbu databáze. Popsal nejpoužívanější technologie, a i přes to, že si nevybral nejčastěji volenou databázi pro webové aplikace, zdůvodnil svou volbu opět větší mírou zkušeností s vybranou technologií.

Jako konkurenci, se kterou by se chtěl rovnat, si zvolil především produkty Helios, které jsou v současné době jedněmi z nejrozšířenějších a nejpoužívanějších. Tyto produkty poskytují velkou škálu informačních systémů jako řešení pro různě velké firmy. Laťku si tedy nastavil velmi vysoko, přesto si udržel výhodu v nezávislosti na platformě a absenci nutnosti cokoli stahovat, instalovat a aktualizovat. Proto si vybral i druhého konkurenta – webovou aplikaci, poskytující služby ve velmi podobném duchu. Oproti této aplikaci si však udržel výhodu v podobě větší uživatelské přívětivosti.

Fáze analýzy i návrhu byly dostatečně popsány a doplněny všemi potřebnými diagramy pro lepší nastínění problematiky. Byl zde vidět rozdíl mezi fází analýzy a fází návrhu, kde došlo ke značnému zkonkrétnění popisu fáze i diagramů.

Hlavní cíl práce se autor pokusil splnit pomocí své webové aplikace pro správu skladového systému, implementované v HTML, CSS, PHP a JavaScript a postavené na databázi Oracle SQL. Základní funkce, která měla aplikace obsahovat implementoval a tuto implementaci i výslednou funkčnost popsal s doplněním obrázků pro ilustraci. Rovněž implementoval uživatelské role, jejichž oprávnění a účely uplatnění rovněž popsal na ilustračním příkladu.

# POUŽITÁ LITERATURA


[1] 1. díl - Úvod do Nette frameworku pro PHP. *ITnetwork* [online]. [cit. 2017-04-11]. Dostupné z: http://www.itnetwork.cz/php/nette/zaklady/uvod-do-php-frameworku-nette/

[2] 7.1 Ocenění zásob. *STORMWARE: SOFTWARE DEVELOPMENT* [online]. [cit. 2017-04-07]. Dostupné z: https://www.stormware.cz/prirucka-uctujeme-online/Zasoby/Oceneni_zasob/

[3] 7.Co je to HTML a jak jej využít na svém webu? *Webyan* [online]. [cit. 2017-04-08]. Dostupné z: http://podpora.goneo.cz/535451-7Co-je-to-HTML-a-jak-jej-vyu%C5%BE%C3%ADt-na-sv%C3%A9m-webu

[4] About MySQL. *Mysql.com* [online]. [cit. 2017-04-12]. Dostupné z: https://www.mysql.com/about/

[5] About SQLite. *SQLite* [online]. [cit. 2017-04-14]. Dostupné z: http://sqlite.org/about.html

[6] About. *PostgreSQL* [online]. [cit. 2017-04-14]. Dostupné z: https://www.postgresql.org/about/

[7] BODNÁR, Jan. Novinky jazyka PHP 7. *Root.cz* [online]. [cit. 2017-04-11]. ISSN 1212-8309. Dostupné z: https://www.root.cz/clanky/novinky-jazyka-php-7/

[8] Bringing MySQL to the web. *PhpMyAdmin* [online]. [cit. 2017-04-13]. Dostupné z: https://www.phpmyadmin.net/

[9] CSS styly - úvod. *Jak psát web* [online]. [cit. 2017-04-11]. Dostupné z: https://www.jakpsatweb.cz/css/css-uvod.html

[10] ČÁPKA, David. 1. díl - Úvod do Java Enterprise Edition (JEE). *ITnetwork* [online]. [cit. 2017-04-11]. Dostupné z: http://www.itnetwork.cz/java/jee/java-enterprise-edition-uvod-do-jee-j2ee

[11] ČÁPKA, David. 1. díl - Úvod do JavaScriptu. *ITnetwork* [online]. [cit. 2017-04-11]. Dostupné z: http://www.itnetwork.cz/javascript/zaklady/javascript-tutorial-uvod-do-javascriptu-nepochopeny-jazyk

[12] EET - Skladový program, software pro fakturaci, dodací listy, příjemky, výdejky, objednávky, úhrady …. *ProdejOnline* [online]. [cit. 2017-04-16]. Dostupné z: http://www.prodejonline.cz/sklad-a-fakturace/

[13] HELIOS Green. *HELIOS* [online]. [cit. 2017-04-16]. Dostupné z: http://www.helios.eu/produkty/helios-green/

[14] HELIOS Red. *HELIOS* [online]. [cit. 2017-04-16]. Dostupné z: http://www.helios.eu/produkty/helios-red/

[15] HTML. *Adaptic* [online]. [cit. 2017-04-08]. Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/html/

[16] Informace o webových aplikacích. *Dreamweaver* [online]. [cit. 2017-04-08]. Dostupné z: https://helpx.adobe.com/cz/dreamweaver/using/web-applications.html

[17] JavaScript. *Tvorba-webu* [online]. [cit. 2017-04-11]. Dostupné z: https://www.tvorba-webu.cz/javascript/

[18] JIT (Just-in-time). *ManagementMania* [online]. 2016 [cit. 2017-04-07]. ISSN 2327-3658. Dostupné z: https://managementmania.com/cs/just-in-time

[19] MICHELL. Oracle – základní pojmy. *IT Blok* [online]. [cit. 2017-04-12]. Dostupné z: http://itblok.bastler.cz/subdom/itblok/oracle-zakladni-pojmy/

[20] MONUS, Anna. 10 PHP Frameworks For Developers – Best of. *HONGKIAT* [online]. [cit. 2017-04-11]. Dostupné z: http://www.hongkiat.com/blog/best-php-frameworks/

[21] MySQL Editions. *Mysql.com* [online]. [cit. 2017-04-12]. Dostupné z: https://www.mysql.com/products/

[22] Oracle and MySQL Compared. *ORACLE Help Center* [online]. [cit. 2017-04-13]. Dostupné z: https://docs.oracle.com/cd/E12151_01/doc.150/e12155/oracle_mysql_compared.htm#CHDIIBJH

[23] *PHP 5 a MySQL 5: průvodce webového programátora*. Brno: Computer Press, 2007, s. 113-137. ISBN 9788025118139.

[24] Produkty HELIOS. *HELIOS* [online]. [cit. 2017-04-16]. Dostupné z: http://www.helios.eu/produkty/

[25] System Properties Comparison MySQL vs. Oracle vs. PostgreSQL. *DB-ENGINES* [online]. [cit. 2017-04-14]. Dostupné z: https://db-engines.com/en/system/MySQL%3BOracle%3BPostgreSQL

[26] System Properties Comparison Oracle vs. SQLite. *DB-ENGINES* [online]. [cit. 2017-04-14]. Dostupné z: https://db-engines.com/en/system/Oracle%3BSQLite

[27] ŠŤASTNÝ, Jiří, ŠIMEČEK, Martin, ed. CSS3 - držte krok s dobou (nové vlastnosti). *Programujte.com* [online]. [cit. 2017-04-11]. ISSN 1801-1586. Dostupné z: http://programujte.com/clanek/2010070801-css3-drzte-krok-s-dobou-nove-vlastnosti/

[28] ŠŤASTNÝ, Jiří, ŠIMEČEK, Martin, ed. HTML5 - nové vlastnosti. *Programujte.com* [online]. 2011 [cit. 2017-04-11]. ISSN 1801-1586. Dostupné z: http://programujte.com/clanek/2010082200-html5-nove-vlastnosti/

[29] *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. 2., aktualiz. a dopl. vyd. Brno: Computer Press, 2007, s. 135-478. ISBN 978-80-251-1503-9.

[30] Úvod do PL/SQL. *IT4KT* [online]. [cit. 2017-04-12]. Dostupné z: http://it4kt.cnl.sk/c/dbs/student/10.html

[31] VITA. Kterou SQL databázi použít? *ITnetwork* [online]. [cit. 2017-04-12]. Dostupné z: http://www.itnetwork.cz/programovani/databazovy-dotazovaci-jazyk-sql/kterou-sql-databazi-pouzit

[32] WYSIWYG. *Adaptic* [online]. [cit. 2017-04-11]. Dostupné z: http://www.adaptic.cz/znalosti/slovnicek/wysiwyg/

[33] ZAJÍC, Petr. PHP (1) - Historie a budoucnost. *Linuxsoft* [online]. [cit. 2017-04-11]. ISSN 1801-3805. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=171

[34] ZAJÍC, Petr. PHP (2) - Jak to funguje. *Linuxsoft* [online]. [cit. 2017-04-11]. ISSN 1801-3805. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=172

[35] Zásoby - pojmy, ocenění. *Otevřená škola* [online]. [cit. 2017-04-07]. Dostupné z: https://www.oalib.cz/oskola/mod/book/tool/print/index.php?id=552

# PŘÍLOHY

[1] Kompletní textová část bakalářské práce ve formátu PDF na přiloženém CD

[2] Logický databázový model z programu SQL Developer na přiloženém CD

[3] Relační databázový model z programu SQL Developer na přiloženém CD

[4] Skripty pro vytvoření databáze, procedur, funkcí a triggerů na přiloženém CD

[5] Zdrojové kódy webové aplikace na přiloženém CD