Univerzita Pardubice
Fakulta elektrotechniky a informatiky

# Internetový obchod s prodejem software

Radek Prokeš

Bakalářská práce
2009

**Univerzita Pardubice**
**Fakulta elektrotechniky a informatiky**
**Katedra informačních technologií**
Akademický rok: **2008/2009**

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Radek PROKEŠ**

Studijní program: **B2646 Informační technologie**

Studijní obor: **Informační technologie**

Název tématu: **Internetový obchod s prodejem software**

Zásady pro vypracování:

Úkolem práce bude návrh a tvorba internetové aplikace elektronického obchodu s využitím relačních databází. Elektronický obchod je určen pro prodej PC softwaru. Teoretická část se bude zabývat problematikou SEO, navržené stránky e-shopu budou optimalizovány pro vyhledavače. Aplikace musí umožnit: - Registraci zákazníků - Vyhledávání zboží ze sortimentu - Objednávání a prodej - stažení software - Evidenci plateb - Sledování historie a stavu objednávek pro registrované zákazníky - Informování zákazníků o jejich objednávkách prostřednictvím e-mailu - Přístup dle práv

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

**Staníček, P. CSS Kaskádové styly - Kompletní průvodce. Computer Press, 2003. Castagnetto, J. a kol. Programujeme PHP profesionálně. Computer Press, 2004. Škultéry, R. JavaScript - Programujeme internetové aplikace. Computer Press, 2004. Oppel, A. Databáze bez předchozích znalostí. Computer Press, 2006. Smička, R. Optimalizace pro vyhledavače - SEO. Jaroslava Smičková, Dubany, 2004.**

Vedoucí bakalářské práce: **RNDr. Iva Rulicová**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. ledna 2009**

Termín odevzdání bakalářské práce: **15. května 2009**

doc. Ing. Simeon Karamazov, Dr.
děkan

L.S.

Ing. Lukáš Čegan
vedoucí katedry

V Pardubicích dne 31. března 2009

## Prohlášení autora

Prohlašuji:

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

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

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 11. 5. 2009

Radek Prokeš

## Anotace

Tato práce se zabývá problematikou návrhu a tvorby internetového obchodu. Cílem je vytvoření konkrétní implementace pro prodej softwaru, kde tento software bude poskytován elektronicky. Práce se dotýká témat správného návrhu tabulek v relační databázi a optimalizací stránek pro vyhledávače.

## Klíčová slova

internetový obchod, e-shop, relační databáze, optimalizace pro vyhledávače, SEO

## Title

Internet store with sale of software

## Annotation

This bachelor work deals with the design and creation of the internet store. The aim is to create a specific implementation for the sale of software, where this software will be available electronically. This bachelor work is concerned with the subjects of correct draft of tables in a relational database and optimizing of the sites for the search engines.

## Keywords

internet store, e-shop, relation database, search engine optimization, SEO

## Obsah

## Seznam obrázků

## Přehled zkratek

| | |
|---|---|
| **AJAX** – Asynchronous JavaScript and XML | – asynchronní JavaScript a XML |
| **CSS** – Cascading Style Sheet | – kaskádové styly |
| **HTML** – HyperText Markup Language | – hypertextový značkovací jazyk |
| **OOP** – Object Oriented Programming | – objektově orientované programování |
| **MVC** – Model – View – Controller | – model – pohled – ovladač |
| **PHP** – PHP: Hypertext Preprocessor | – PHP: hypertextový preprocesor |
| **SEO** – Search Engine Optimization | – optimalizace pro vyhledávače |
| **URL** – Uniform Resource Locator | – jednoznačné určení zdroje na Internetu |
| **XHTML** – eXtensible HyperText Markup Language | – rozšířitelný hypertextový značkovací jazyk |
| **XML** - eXtensible Markup Language | – rozšiřitelný značkovací jazyk |

# 1 Úvod

Cílem této práce je vytvořit konkrétní implementaci internetového obchodu pro prodej softwaru. Tento software je poskytován elektronicky, tedy po jeho objednání a zaplacení bude mít zákazník možnost přímého stažení ze stránek. Dále se práce zabývá optimalizací tohoto internetového obchodu pro vyhledávače.

Kapitola Internetový obchod stručně seznamuje s tématem, co to vlastně internetový obchod je, jeho historii a jak fungují dnešní moderní e-shopy.

Další kapitola nazvaná Optimalizace pro vyhledávače (SEO) nejprve představí co to je vyhledávač a jakým způsobem pracuje. Dále budou popsány konkrétní způsoby optimalizace webových stránek a to jak z pohledu jedné konkrétní stránky, tak i webu jako celku. Na konci kapitoly budou uvedeny některé zakázané metody optimalizace obsahu.

Kapitola Analýza internetového obchodu se zabývá konkrétními požadavky na vytvářený internetový obchod a jeho analýzu včetně zvolených technologií pro následný vývoj.

V následujících dvou kapitolách Návrh databáze a Implementace webové části je již prakticky popsáno, jakým způsobem byl vytvořen vzorový internetový obchod.

# 2 Internetový obchod

Internetový obchod (často nazývaný e-shop) je počítačová aplikace sloužící především k nabídce zboží, v menší míře i služeb. Tato aplikace slouží také k příjmu objednávek od zákazníků, sledování a zprostředkování plateb, informování zákazníků o různých novinkách nebo akcích v našem e-shopu a dalších obchodních aktivitách [1].

### Historie

Historie internetových obchodů sahá do první poloviny 90. let 20. století, kdy byly ve Spojených státech nasazovány první aplikace tohoto druhu. Po roce 2000 se internetové obchody rozšířili po celém světě a díky rychlosti, pohodlnosti nakupování a nižším cenám se v dnešní době staly plnohodnotnou alternativou klasických kamenných obchodů [2].

### Důležité části

Jednou z nejdůležitějších částí internetového obchodu je katalog nabízeného zboží. Ten je ve většině případů rozdělen do několika kategorií a podkategorií s možností řadit tyto produkty podle určitých kritérií. Téměř každý e-shop také obsahuje službu vyhledávání v katalogu, která dokáže najít určitý námi hledaný produkt nebo více produktů podle klíčových slov nebo specifických údajů. Některé internetové obchody umožňují vyhledávat i podle parametrů.

Další neméně důležitou částí je nákupní košík, do kterého jsou v průběhu nákupu vkládány námi vybrané produkty. Nákupní košík umožňuje jednoduše měnit množství jednotlivých produktů, případně některé nebo všechno zboží z košíku odstranit. Také zobrazuje cenu každého produktu i celkovou cenu našeho nákupu. Po dokončení nákupu je obsah nákupního košíku předán do modulu vyřizujícího objednávky.

V internetovém obchodu také bývá často implementována neveřejná část pro registrované zákazníky. Tento modul může obsahovat sekci s objednávkami, díky které je možné sledovat stav u každé z nich. Dále tu může být sekce s fakturami k jednotlivým objednávkám, reklamační sekce atd.

Díky dynamičnosti a proměnlivosti nabídky zboží obsahují dnešní internetové obchody modul pro snadnou administraci. Tento modul umožňuje měnit nabízené zboží včetně jeho zařazení do kategorií, sledovat vyřízené objednávky a také registrované zákazníky.

### Implementace

Internetový obchod je většinou realizován jako sada skriptů napsaná v určitém programovacím jazyce [1]. Tyto skripty se starají o dynamický běh obchodu. Vykreslují jednotlivé webové stránky, komunikují s databází, odkud načítají informace o jednotlivých produktech, ukládají do ní objednávky atd.

Některé pokročilejší internetové obchody mohou být navrženy pro snadnou synchronizaci údajů s ekonomickým softwarem. V dnešní době se už také stává standardem napojení internetového obchodu přímo na některé z platebních systémů bank. Tyto platební systémy umožňují jednoduše, rychle a efektivně zaplatit naši objednávku, která se může zároveň automaticky přenést do ekonomického systému používaného ve firmě.

# 3 Optimalizace pro vyhledávače (SEO)

*„Optimalizace pro vyhledávače je obecně řada úkonů a operací, jejichž cílem je především zlepšení pozic ve fulltextových vyhledávačích = zásadní zlepšení návštěvnosti cílovou skupinou zákazníků.“ [6]*

Pro zviditelnění a zajištění lepší pozice našich stránek ve vyhledávačích se používá několik metod. Mezi základní metody patří SEO (Search Engine Optimization) a SEM (Search Engine Marketing). SEM je metoda, kdy se platí přímo vyhledávači za zviditelnění našich webových stránek a v textu již dále zmiňována nebude.

## 3.1 Vyhledávače

*„Internetový vyhledávač je služba, která umožňuje na Internetu najít webové stránky, které obsahují požadované informace.“ [4]*

Vyhledávače se rozdělují do dvou základních kategorií, a to na katalogové a fulltextové.

### *3.1.1 Katalogový vyhledávač*

Katalogový vyhledávač je služba, která ve své databázi udržuje seznam odkazů na jiné webové stránky. Tyto odkazy bývají většinou pro lepší nalezení hledané stránky rozřazeny do kategorií. Na rozdíl od fulltextových vyhledávačů jsou tyto katalogy udržovány ručně. Záznam do určitého katalogu se provádí registrací do konkrétní kategorie, přičemž po zkontrolování odkazu administrátorem je záznam zobrazován. V katalogu se vyhledává buď pomocí procházení jednotlivých kategorií anebo zadáním dotazu, kdy se dotaz porovnává s URL, nadpisy a popisy zaregistrovaných stránek. Mezi nejznámější české katalogové vyhledávače patří Seznam, Centrum a Atlas. Poslední dva zmiňované již patří pod jednu společnost.

### *3.1.2 Fulltextový vyhledávač*

Fulltextový vyhledávač je software, který indexuje dokumenty na internetu a ukládá si je do své databáze. Fulltextový je proto, že obvykle prohledává celý text dokumentu. Uživatelům pak umožňuje v této databázi vyhledávat pomocí zadaných dotazů. Mezi dnešní nejznámější fulltextové vyhledávače patří Google, Yahoo, Live Search a u nás Seznam.

Fulltextový vyhledávač se obvykle skládá ze dvou částí. Z robota, který indexuje jednotlivé webové stránky a ostatní dokumenty na internetu a z webového rozhranní, které umožňuje přístup do indexované databáze. Samotný robot se dále dělí na část, která prochází web a stahuje dokumenty, a na část, která tyto dokumenty

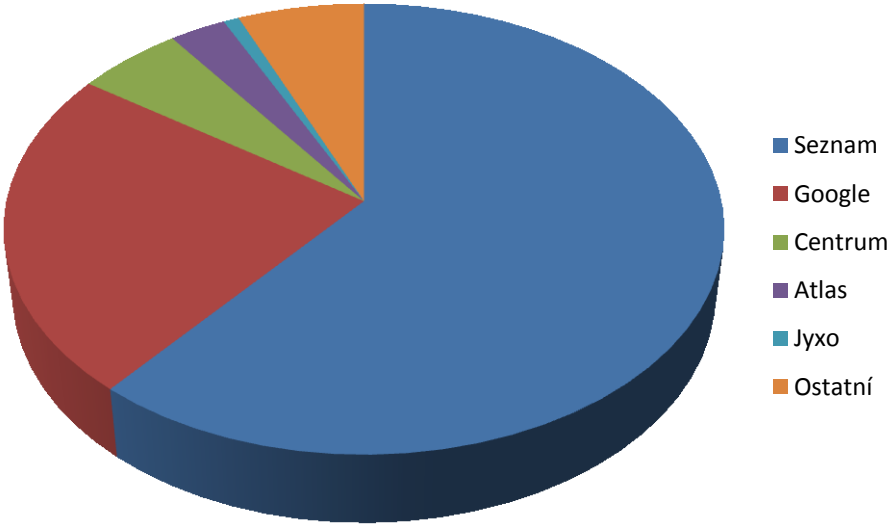
indexuje. Indexovací robot se umí pohybovat po stránkách samostatně a umí si webové stránky najít sám. Nelze mu říci, kdy a jak často se má na stránky vracet. Lze mu pouze nařídit, co nemá na webu indexovat [3].

Při vyhledávání se zadá na webové stránce vyhledávače dotaz, který je analyzován, a následně se pomocí určitých algoritmů vyhledají relevantní stránky v databázi. Jelikož jsou tyto algoritmy tajemstvím, tak každý vyhledávač po zadání stejného dotazu může vrátit rozdílnou sadu výsledných odkazů a můžeme se jenom domnívat, že tyto výsledky jsou relevantní hledanému dotazu. Výsledky hledaného dotazu jsou navíc seřazeny podle tzv. ratingu (hodnocení) stránky daným vyhledávačem. Toto hodnocení ovlivňují nejrůznější faktory, které jsou uvedeny dále.

## 3.2 Aktuální tržní podíl vyhledávačů v České Republice

- Seznam: 61,10%
- Google.cz: 23,48%
- Centrum: 5,37%
- Atlas: 2,86%
- Jyxo: 0,79%
- Ostatní (včetně zahraničních): 6,4%

**Obrázek 1: Podíl vyhledávačů v ČR**




*Zdroj: www.navrcholu.cz (říjen 2008)*

## 3.3 Předpoklady pro SEO optimalizaci

Když stránku indexuje vyhledávací robot, tak si ji interně dokáže zobrazit pouze jako staré internetové prohlížeče, které v té době byli textové. Vyhledávací robot nedokáže interpretovat a díky tomu i zaindexovat určitý obsah webových stránek, jako je JavaScript, rámce, Flash animace nebo Silverlight. Toto je důležité si při vytváření stránek uvědomit a následně se tomu i přizpůsobit.

### *3.3.1 JavaScript*

JavaScript je skriptovací jazyk navržený pro oživení částí webových stránek nebo například ke kontrole formulářů. Pokud je použit k vytvoření navigace, tak ho robot ignoruje a tudíž nebude zaindexován celý web, ale většinou pouze první stránka [3].

### *3.3.2 Rámce*

Při použití rámců je obvykle stránka rozdělena na část s menu a na část s obsahem stránky. Při indexování těchto stránek je problém v tom, že vyhledávač indexuje samostatně menu a samostatně obsah stránky. Proto když vyhledávač zobrazuje konkrétnímu uživateli tuto stránku, tak je to většinou jenom obsah. Uživatel se proto ve stránce může ztratit a nedostane se na další části naší webové prezentace [3].

### *3.3.3 URL adresy*

Dynamické tvary adres mají v části za otazníkem různé parametry (většinou ID částí, které nějak souvisí se zobrazovanou stránkou) oddělené ampersandem. Tato URL může být např. ve tvaru http://www.domena.cz/index.php?page=product&id=3. Takto vytvořená URL jsou pro uživatele velice nepřehledná a nedají se jednoduše zapamatovat. Pro vyhledávacího robota to ale znamená, že stránka je dynamická a tudíž ji nemusí přiřadit takovou váhu, jako stránkám statickým. Jedna z metod SEO optimalizace je i změna dynamických adres na statické. Pro téměř každou technologii používanou na webu existují nástroje, jak toho dosáhnout. Po transformaci výše uvedené dynamické adresy na adresu statickou je možné dostat URL ve tvaru například http://www.domena.cz/antiviry/eset-smart-security.html.

Adresa by také měla být neměnná. Při jakékoliv změně by stará URL měla zůstat funkční a přesměrovat na novou adresu pomocí hlavičky HTTP 301 – Moved Permanently. Přesměrování s touto hlavičkou je nutné proto, aby vyhledávače neměli ve své databázi indexované duplicitní stránky.

S duplicitou stránek také souvisí fakt, že např. adresy ve tvaru www.domena.cz a www.domena.cz/index.html jsou vyhledávači považovány za dvě různé stránky.

Proto je nutné takovéto adresy v aplikaci detekovat a vždy přesměrovat pouze na tzv. kanonickou adresu [3].

### *3.3.4 Validita HTML kódu*

Současné vyhledávače dávají vyšší váhu webovým stránkám, které splňují webové standardy a mají validní HTML kód. Pro kontrolu, jestli jsou vytvořené stránky validní, je možné použít jakýkoliv validátor, např. jeden takový je zdarma dostupný na adrese http://validator.w3.org/.

## 3.4 Optimalizace obsahu – On page faktory

Optimalizací obsahu se rozumí, že vytvořený zdrojový HTML kód by měl být sémantický. To znamená dobře strukturovaný a s významem. Roboti vyhledávačů spoléhají na tento zdrojový kód, aby určili v jakém kontextu je daná část textu. Proto je doporučeno vytvářet zdrojový kód způsobem snadno pochopitelným pro indexovací roboty.

On page faktor dále určuje, že toto je optimalizace pouze jedné konkrétní stránky. Základem každé optimalizace je to, že každá stránka na webu musí být unikátní. Indexovací robot totiž hodnotí každou stránku zvlášť [3].

### *3.4.1 Klíčová slova*

Klíčová slova hrají nejdůležitější roli v SEO optimalizaci. Pokud se na stránce neobjevují klíčová slova, která by mohla být zadána ve vyhledávači, tak je ve většině případů vyhledávačem pod těmito slovy nenalezena. Pro výběr relevantních klíčových slov je dobré nejdříve si rozmyslet, jak by naši stránku mohl uživatel najít a co by mohl zadat do vyhledávače. Následně je potřeba tato klíčová slova ve stránce vhodným způsobem optimalizovat. Postupů jak toho dosáhnout existuje celá řada [3].

#### Klíčová slova ve stránce

Předem vybraná klíčová slova by měla být ve stránce s určitou hustotou. Hustota vyjadřuje podíl četnosti těchto klíčových slov a celkového počtu slov na celé stránce. Optimální hustota se pohybuje mezi 2 až 7 %. Při psaní takto optimalizovaných textů je ale důležité dbát především na obsah. Jednotlivé texty jsou psány pro návštěvníky webových stránek a ne pro vyhledávače, tudíž by tato forma optimalizace neměla být brána jako hlavní [3].

#### Klíčová slova v URL adrese

Význam URL adresy je velký jak pro vyhledávače, tak i pro uživatele, kteří můžou automaticky předpokládat, co se na dané stránce vyskytuje. Proto se vyplatí

umisťovat klíčová slova i do URL adresy. V mnoha případech bývá efektivní a pro uživatele příjemné dávat do URL upravený text hlavního nadpisu určité stránky.

### 3.4.2 Sémantický HTML kód

Pro vytváření obsahu stránky jsou používány HTML značky, které v dobře optimalizované stránce určují v jakém kontextu je daný text. Tímto způsobem je určeno, že konkrétní část je nadpis (tagy <h1> až <h6>), jiná část je odstavec textu, seznam nebo hypertextový odkaz. Pro vizuální podobu je poté použita jiná technologie, například kaskádové styly (CSS). Teprve pomocí kaskádových stylů se určí, že nadpisy budou modré, budou mít určitou velikost a budou podtržené.

### 3.4.3 Titulek stránky

Titulek stránky je vytvořen pomocí značky title v hlavičce stránky. Tento tag je považován za jeden z nejdůležitějších, a proto by neměl nikdy chybět na žádné stránce. Každá stránka naší webové prezentace by měla mít unikátní titulek, který se skládá z názvu naší webové prezentace a z části, která určuje, co konkrétní stránka obsahuje za informace [3].

```
<head>
   <title>Titulek stránky</title>
</head>
```

### 3.4.4 Meta description

Tento tag obsahuje popis, co která stránka obsahuje. Podobně jako titulek stránky by i obsah této značky měl být odlišný pro každou stránku. V současné době vyhledávače pokládají přítomnost této značky za důležitou součást, některé mohou dokonce obsah zobrazit ve výsledcích vyhledávání [3].

```
<meta name="description" content="Popis stránky" />
```

### 3.4.5 Meta keywords

Tento tag obsahuje klíčová slova, která jsou taktéž specifická pro jednotlivé stránky [3].

```
<meta name="keywords" content="klíčová slova stránky" />
```

### 3.4.6 Nadpisy

Pro nadpisy ve webové stránce je doporučeno používat tagy k tomu určené. Vyhledávači se tím jasně dá najevo, že texty uzavřené v těchto značkách mají vyšší váhu než okolní text. Pro nadpisy se používají značky H1 až H6. Čím vyšší číslo, tím je nižší váha tohoto textu. Pro nadpisy platí, že značka H1 se smí vyskytovat ve stránce pouze jednou, ostatní značky vícekrát [3].

```
<h1>Hlavní nadpis stránky</h1>
<h2>Podnadpis</h2>
```

### 3.4.7 Tučný text a kurzíva

Text, který je potřeba ve stránce pouze zvýraznit, ale nehodí se do nadpisů, se označuje tučně nebo kurzívou. Pro označení těchto textů se používají značky STRONG a EM. Tyto značky se zobrazují stejně jako zastaralé značky B a I, které ale pouze určují styl písma, nikoliv jejich váhu v textu [3].

```
<strong>Tučný text</strong>
<em>Kurzíva</em>
```

### 3.4.8 Popisky obrázků a hypertextových odkazů

Každý obrázek používaný ve stránce by měl mít definovaný atribut alt, který obsahuje popis obrázku. Tento atribut se zobrazí uživatelům, kteří mají vypnuté obrázky, a zároveň i vyhledávacím robotům.

Nepovinný atribut title u obrázků i odkazů navíc obsahuje informace o tom, co je na obrázku, respektive kam daný odkaz míří [3].

```
<img src="image.gif" alt="popis obrázku" title="shrnutí obsahu" />
<a href="http://www.domena.cz/odkaz.html" title="popis cíle" />
```

## 3.5 Optimalizace obsahu – Off page faktory

Mezi off page faktory patří ta část optimalizace, která se netýká jedné konkrétní stránky, ale webových stránek jako celku. Tato část optimalizace se zaměřuje především na odkazy, tedy zpětné odkazy na naše stránky a na provázání jednotlivých stránek mezi sebou.

## 3.6 Zakázané metody SEO

Tyto metody mohou někdy zvýšit pozici stránek ve vyhledávačích, ale pokud je vyhledávací robot detekuje, tak jsou stránky penalizovány nebo rovnou i vyřazeny z katalogu [6].

Mezi zakázané metody patří:

- Skrytý text obsahující klíčová slova pro zvýšení jejich frekvence na stránce
- Cloaking – speciálně upravené stránky pouze pro vyhledávací roboty
- Odkazové farmy uměle vytvářející zpětné odkazy na naše stránky
- Tapetování katalogů – metoda, kdy se do katalogů zaregistruje stejná stránka, ale pod různými doménami

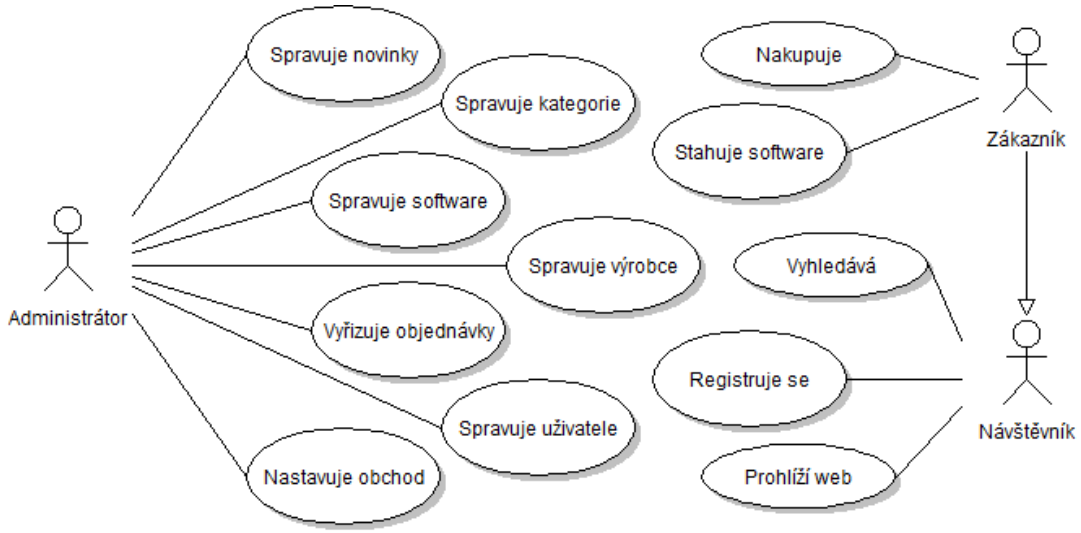
# 4 Analýza internetového obchodu

Před započetím prací na implementaci internetového obchodu byla provedena důkladná analýza, která dokázala již předčasně odhalit některé možné problémy. Bylo určeno, v jakém prostředí se bude pracovat, tedy jaké konkrétní technologie budou použity. Dále byly stanoveny určité konkrétní požadavky, které budou na internetový obchod kladeny a které by měli být implementovány.

## 4.1 Use Case diagram

Use Case diagram zobrazuje strukturu systému z pohledu uživatele, tedy popisuje interakci mezi uživatelem a systémem. Je určen k definici chování, aniž by odhalil vnitřní strukturu tohoto systému.

**Obrázek 2: Use Case diagram**




## 4.2 Požadavky

Z pohledu potenciálního zákazníka je důležitá přehlednost a jednoduchost použití. Zákazník nesmí být od nákupu odrazen a neměl by mít pocit, že by se na stránce ztratil a například nevěděl jak si objednat software. Zároveň by ale také po něm nemělo být požadováno příliš mnoho osobních údajů, ale jen ty opravdu nezbytně nutné pro úspěšné vyřízení objednávky a případné vystavení faktury.

Aplikace jako internetový obchod sestává z mnoha modulů, které různým způsobem navzájem spolupracují. Proto je důležité zachování přehlednosti ve zdrojovém kódu z důvodu možného budoucího rozšíření o další moduly. U internetových obchodů je důležitá také otázka bezpečnosti. Jde především o zabezpečení proti neoprávněnému vniknutí jak do aplikace, tak třeba i do databázového serveru. Dále je nutné zachovat integritu dat, aby se nestalo například to, že jedna objednávka bude přiřazena jinému zákazníkovi. Také je

potřeba ověřovat jakoukoliv komunikaci, která přichází od uživatele sedícího za monitorem, především validovat vstupy a ošetřovat proměnné na výstupu.

## 4.3 Rozvržení a vzhled

Rozvržení layoutu aplikace bylo zvoleno s ohledem na jednoduchost použití. Stránka je rozdělena na několik hlavních části – hlavičku, hlavní obsah, který může v levé části navíc obsahovat menu, a patičku.

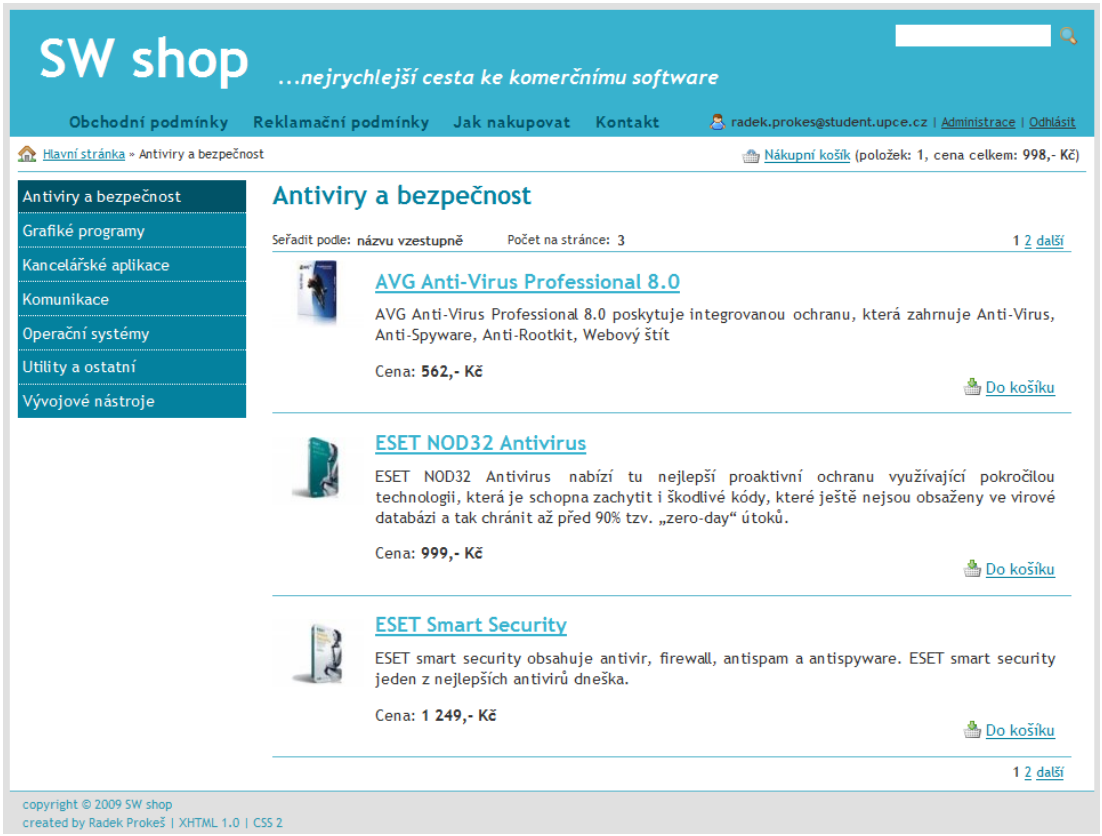
Hlavička stránky obsahuje logo (případně textový název našeho obchodu), reklamní text, vyhledávací políčko pro rychlé hledání v katalogu a komponentu s informací o tom, zda je do obchodu přihlášen nějaký uživatel. Tato komponenta má několik stavů. Pokud není do obchodu nikdo přihlášen, zobrazí odkaz na přihlášení nebo registraci nového zákazníka. V případě, že někdo již přihlášen je, zobrazí se emailová adresa, pod kterou se daný uživatel přihlásil, dále odkaz do soukromé části obchodu (u zákazníka je to odkaz do profilu s informacemi o objednávkách, fakturách atd., v případě administrátora odkaz do administrační sekce) a nakonec je zde odkaz pro odhlášení.

Pod hlavičkou najdeme lištu se dvěma dalšími komponentami. V levé části se nachází komponenta zobrazující takzvanou drobečkovou navigaci. Jedná se o menu, které dynamicky zobrazuje aktuální stav zanoření do struktury webu. Jednotlivé části jsou zároveň odkazy pro přechod o jednu nebo více úrovní směrem nahoru. V pravé části se nachází druhá komponenta, která zobrazuje aktuální stav našeho nákupního košíku, tedy kolik položek nákupní košík obsahuje a jaká je jejich celková cena, a také odkaz pro přechod do tohoto nákupního košíku.

Největší plochu zabírá střední část zobrazující právě požadovanou stránku. Tato část může, ale i nemusí obsahovat menu v levé části. Toto menu se zobrazuje při procházení katalogem, ale pro úsporu místa se nezobrazuje, pokud se zákazník aktuálně nachází v nákupním košíku nebo dokončuje objednávku.

Poslední viditelnou, ale nevýraznou částí webové aplikace je patička. Tato část obsahuje copyright a další právní informace o internetovém obchodě.

**Obrázek 3: Layout webu**




## 4.4 Technologie využívané u složitějších dynamických stránek

Jelikož internetový obchod patří mezi složitější aplikace, byla v analýze zamítnuta jakákoliv implementace bez pomocných knihoven. V dnešní době se pro vývoj webů používají určité ověřené techniky, které přinášejí zrychlení vývoje a lepší udržovatelnost zdrojových kódů. Některé z nich jsou popsány dále v této kapitole.

### *4.4.1 Frameworky*

Framework je sada knihoven a podpůrných nástrojů, které pomáhají při vývoji aplikací a obvykle přebírají typické problémy, které vznikají při vývoji na určité platformě. Díky tomu se nemusí řešit stále se opakující úkoly a drobnosti, které odvádějí pozornost při programování, ale je možné se soustředit pouze na konkrétní funkčnost aplikace. Celkově tedy jakékoliv frameworky urychlují vývoj a ulehčují práci díky psaní méně kódu, který se tak stává přehlednějším [13].

#### Využití

Framework se hodí pouze pro některé typy aplikací, které můžeme rozlišit například podle rozsáhlosti a složitosti [16].

- **Jednoduché aplikace:** Pro velmi jednoduché aplikace asi nemá smysl, protože velikost frameworku je většinou větší než velikost aplikace. Samotné načítání tohoto frameworku při každém požadavku navíc spotřebuje největší část výkonu potřebnou pro výsledné vygenerování stránky.
- **Středně složité aplikace:** U středně složitých aplikací, jako jsou blogy, internetové obchody, webová fóra nebo wiki stránky se opravdu projevuje síla frameworků. Umožňují psát opravdu kvalitní aplikace a navíc vedou programátora tak, aby aplikaci napsal efektivně. Pokud je aplikace napsána alespoň trošku slušně, tak je díky těmto frameworkům většinou i připravená pro budoucí rozšiřitelnost.
- **Velmi složité aplikace:** U těchto typů aplikací je dobré si rozmyslet, jestli opravdu má smysl je psát v technologii, kterou jsme se rozhodli pro ni použít. Například je nesmyslné psát bankovní systém ve skriptovacím jazyce PHP.

### *4.4.2 Návrhový vzor MVC*

*„Model-view-controller (MVC) je softwarová architektura, která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent tak, že modifikace některé z nich má minimální vliv na ostatní.“ [14]*

#### Princip

Architektura MVC vyžaduje přítomnost tří komponent, mezi které patří model, view a controller.

#### Model

Toto je nejnižší vrstva aplikace, která uchovává data zpracovávaná aplikací. Obsahuje rozhranní pro výběr a manipulaci s těmito daty a stará se o jejich konzistenci.

Tato vrstva neobsahuje pouze skripty používané v aplikaci pro přístup k datům, ale například v případě využívání databázového úložiště obsahuje i toto úložiště včetně dalších databázových objektů, jako jsou pohledy nebo uložené procedury.

#### View (pohled)

Pohled je část aplikace, která pouze zobrazuje data získaná z modelu. Tato část aplikace neobsahuje žádný programový kód, který by nějakým způsobem manipuloval s těmito daty.

**Controller (ovladač)**

Ovladač především reaguje na události pocházející od uživatele a na základě uživatelova požadavku vydává příkazy pro model, který následně aktualizuje data. Ovladač pak také rozhoduje o tom, jaký pohled se vybere pro zobrazení výsledků.

## 4.5 Použité technologie

Při výběru technologií pro implementaci aplikace byl kladen důraz především na spolehlivost, cenovou politiku, snadnost použití a v neposlední řadě i na budoucí možnost dalšího rozšiřování aplikace.

### *4.5.1 Webový server Apache*

Apache HTTP server je multiplatformní webový server s otevřeným zdrojovým kódem. I když jeho popularita poslední dobou klesá, jedná se stále o nejrozšířenější webový server, který obsahuje velmi mnoho dalších dodatečných modulů. Není proto problém nakonfigurovat server ke spolupráci s některým ze skriptovacích jazyků.

### *4.5.2 Skriptovací jazyk PHP*

PHP je skriptovací programovací jazyk, určený převážně pro vývoj webových aplikací. Využívá se na serverové straně aplikace, kde slouží pro dynamické generování HTML kódu na základě určitého požadavku.

Pro vývoj ukázkové aplikace bylo využito PHP ve verzi 5.2.9, nicméně aplikace by měla být funkční na jakékoliv verzi, která je vyšší než 5.2.0. Toto omezení je především dáno některou z dále popsaných použitých technologií.

### *4.5.3 Databázový server MySQL*

MySQL je multiplatformní databázový server (RDBMS - relational database management system) s otevřeným zdrojovým kódem. Databáze je dostupná jak bezplatně pod GNU GPL licencí, tak i pod placenou komerční licencí. Tato databáze se stala populární především ve spojení s operačním systémem GNU/Linux, webovým serverem Apache a skriptovacím jazykem PHP. Spojením těchto softwarových produktů (tzv. LAMP balíček) vzniká zdarma dostupné ucelené řešení pro vývoj webových aplikací [10].

MySQL si prošlo velmi dlouhým vývojem. Její vývoj započal roku 1994. Teprve po několika letech vývoje v roce 2005 přinesla poslední majoritní verze 5 podporu uložených procedur, triggerů a pohledů, které jsou v aplikaci využívány.

MySQL používá několik typů tabulek, které se liší svými možnostmi. Mezi nejznámější patří MyISAM a InnoDB. MyISAM je velice rychlé databázové

úložiště, které ale nepodporuje integritní omezení, kdy jsou cizí klíče jedné tabulky navázány na klíče primární v jiné tabulce. Také neumí pracovat s transakcemi. Naproti tomu InnoDB nabízí podporu integritního omezení a transakcí, ale zase nepodporuje fulltextové vyhledávání.

### 4.5.4 Nette Framework

Nette Framework je sada open source knihoven pro vývoj moderních webových aplikací využívajících skriptovací jazyk PHP verze 5.2.0 a vyšší. Tento framework je založen na návrhovém vzoru Model-View-Controller, který pomáhá oddělit aplikaci do tří nezávislých částí. Obsahuje podporu pro tvorbu komponent a využívá událostmi řízené programování.

*„Nette Framework je napsaný v PHP 5 s plným využitím objektů (OOP). Jeho licence, která vychází z BSD, patří k těm nejvolnějším. Vyrostla kolem něj jedna z nejaktivnějších komunit českých PHP vývojářů, ne-li nejaktivnější vůbec. Nette používají významné tuzemské společnosti. A podle testů je jedním z nejvýkonnějších frameworků.“ [16]*

Tento framework je koncipován jako „otevřený“, tedy definuje určitá předem daná rozhranní, nad kterými je následně postavena konkrétní implementace. Pokud nám nějaká implementace nevyhovuje, není nic jednoduššího, než si napsat implementaci vlastní a pak pomocí jednoho řádku kódu nebo konfiguračně tuto část zdrojového kódu zapojit do systému. Tímto způsobem je také možné zkombinovat funkčnost s jiným systémem nebo frameworkem.

#### Kompaktní verze

Nette Framework obsahuje několik desítek souborů, které obsahují jednotlivé části funkčnosti tohoto frameworku. Při požadavku na stránku je potřebné určité soubory nalinkovat do naší aplikace. Nahrávání většího množství souborů je obecně pomalejší než nahrání jednoho většího souboru. Proto Nette Framework přichází s jednosouborovou kompaktní verzí, která je především určená pro běh aplikace v produkčním prostředí [16].

### 4.5.5 Databázový layer Dibi

Dibi je minimalistický databázový layer, který je přenositelný mezi databázovými systémy. Aktuálně podporuje databáze MySql, PostgreeSql a SqLite, experimentálně MS SQL a Oracle. Cílem je zapouzdřit funkce do intuitivního objektového rozhranní, asistovat při psaní SQL dotazů a zjednodušit práci s výslednými daty získanými z databáze [19].

V aplikacích se obvykle neposílají do databáze předem připravené a neměnné SQL dotazy. Většinou je potřebné omezit rozsah vybíraných dat a dotaz nějak parametrizovat. Pokud je navíc velký počet zobrazovaných dat, je vhodné použít stránkování, případně i řazení podle určitých sloupců v tabulce. Při používání těchto proměnných parametrů se již musí skládat SQL dotaz dynamicky. Navíc by měly být ošetřeny všechny parametry proti SQL Injection. Takovéto dynamické skládání má mnoho nevýhod. Je velice nepřehledné a při pozdějším zkoumání zdrojového kódu nemusí být úplně jasné, co tím autor původně zamýšlel. Navíc v něm lze snadno udělat chyby.

**Klasicky používané dynamické sestavení dotazu**

```
mysql_query("
   SELECT id, name, price
   FROM products
   WHERE category_id = " . intval($categoryId) . "
   ORDER BY " . addslashes($orderBy) . "
   LIMIT " . intval($limit) . " " . intval($offset) . "
");
```

Dibi používá pro zápis SQL dotazů speciální syntaxi, kde se v dotazu místo konkrétního parametru zapíše pouze určitá sekvence znaků. Podle této sekvence pak Dibi samo rozhodne, jak s parametrem naloží. Význam nejdůležitějších znaků je následující: %i – celé číslo, %f – desetinné číslo, %s – řetězec, %sn – řetězec, ale pokud je prázdný tak se automaticky zamění za hodnotu NULL.

**Parametrizovaný dotaz vytvořený za použití dibi**

```
dibi::query("
   SELECT id, name, price
   FROM products
   WHERE category_id = %i
   ORDER BY %s
   %lmt %ofs
", $categoryId, $orderBy, $limit, $offset);
```

V jedné z posledních verzí Dibi se objevila ještě jedna možnost sestavování dynamických SQL dotazů, která je vhodná přímo pro aplikace založené na MVC architektuře. V principu jde o to, že model slouží pouze k získání určitých dat. Ale model by nemělo zajímat, jestli jsou tato data stránkována nebo co konkrétně se má zobrazit. Proto je možné pomocí DataSource objektu vytvořit pouze základní dotaz. Teprve controller na tento dotaz aplikuje stránkovací a popřípadě ještě filtrovací logiku a view si určí, které sloupečky z databáze chce vlastně zobrazit. Tímto způsobem se dají v modelu eliminovat všechny velice podobné metody vracející data, ale obsahující velmi málo rozdílů. [21]

**Datový zdroj vytvořený pomocí dibi, který je možné ještě později upravovat**

```
dibi::dataSource("SELECT * FROM products")
   ->where("category_id = %i", $categoryId)
   ->orderBy($orderBy)
   ->applyLimit($limit, $offset)
   ->select(array("id", "name", "price"));
```

V implementaci aplikace byly využity dvě posledně vyjmenované metody. Model tedy provádí pouze základní filtraci požadovaných dat pomocí dibi parametrů a vrací vytvořený DataSource objekt. Controller pak aplikuje stránkovací logiku pomocí metod tohoto objektu a výsledný objekt je předán do šablony, která si z něj vytáhne pouze potřebná data.

### *4.5.6 JavaScript framework jQuery*

jQuery je framework, který velice zjednodušuje programování klientské části aplikace v JavaScriptu. Slouží pro manipulaci s elementy ve stránce, pro práci s CSS styly, AJAXem a zvládá i jednoduché animace. Jeho výhodou je podpora všech prohlížečů a velmi příjemná velikost. Komprimovaná verze má kolem 20kB [23].

V následujícím jednoduchém příkladě je ukázáno, jak tento framework dokáže ušetřit čas i velikost výsledné stránky, tedy objem přenášených dat ke klientovi. Na některé stránce je například rozsáhlý formulář s velmi velkým počtem řádků a na každém řádku je tlačítko smazat sloužící pro odstranění určitého záznamu. Při kliknutí je po uživateli nejprve požadováno potvrzení, jestli tuto akci opravdu myslí vážně. Proto je do HTML vložen kousek javascriptového kódu.

**Klasická metoda napojení události k tlačítku formuláře:**

```
<input type="submit" name="delete-button" value="Smazat"
   onclick="return confirm('Opravdu si přejete tento záznam
   smazat?');" />
```

Takovýto přístup má mnoho nevýhod. Čím má formulář více řádků, tím větší je objem vygenerované stránky, protože tento kousek kódu se neustále opakuje. Dalším argumentem proti je to, že s přibývajícím javascriptem se stránka postupně stává nepřehlednou.

**Metoda používaná v jQuery:**

```
// definice tlačítka
<input type="submit" name="delete-button" value="Smazat" />

// v jiné části kódu nebo externím souboru
$(document).ready(function() {
   $('input[name=delete-button]').click(function() {
      return confirm('Opravdu si přejete tento záznam smazat?');
   });
});
```

# 5 Návrh databáze

Při návrhu databáze byl kladen důraz především na umístění veškeré databázové logiky přímo do databáze. Dalším kritériem byla i bezpečnost, a to jak z pohledu integrity dat, tak i opatření proti neoprávněnému zásahu do těchto dat.

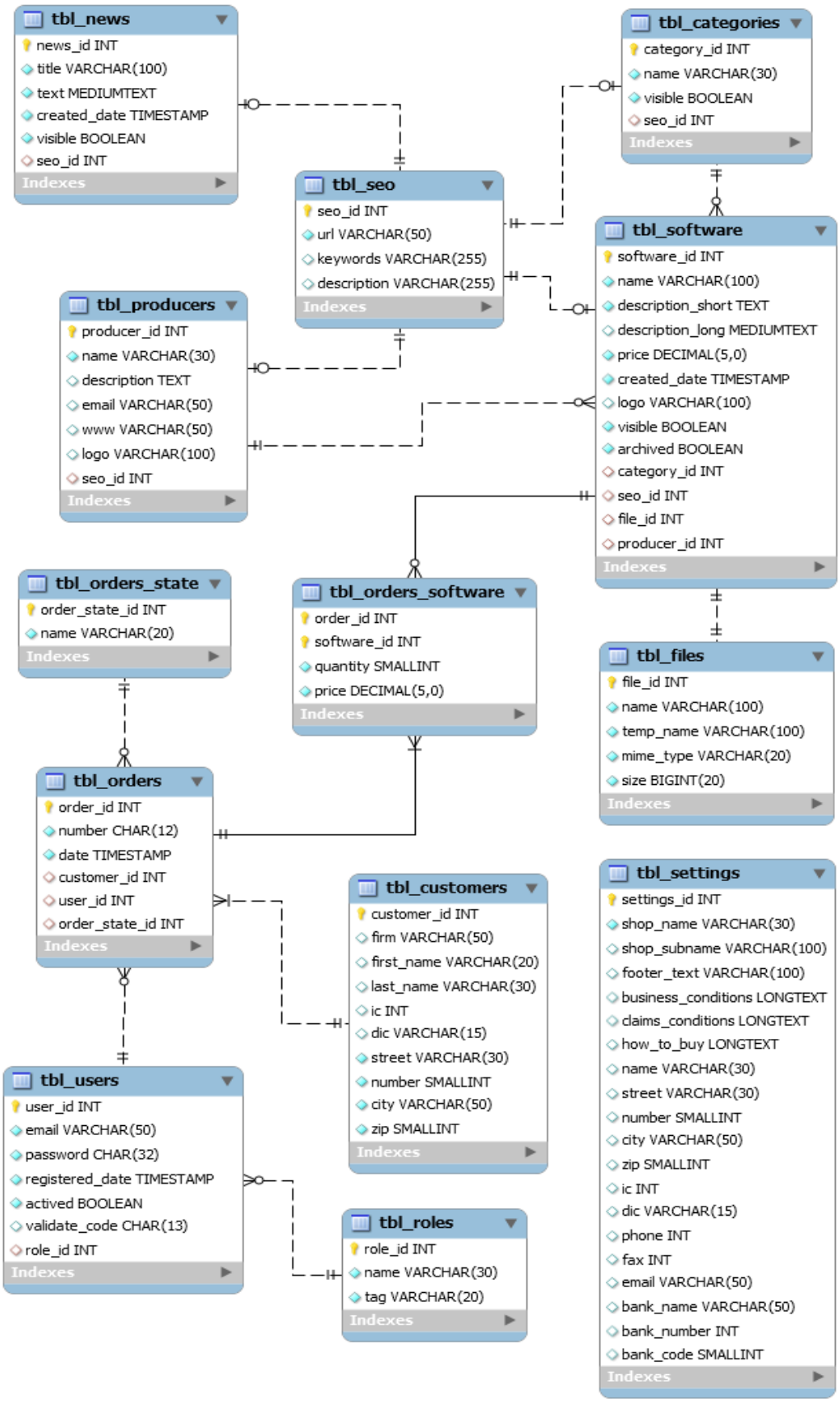
## 5.1 Architektura

Aplikace využívá možnosti poskytované databází pro řízení přístupu k datům. Jsou využity nejen tabulky, ale i pohledy a uložené procedury. Použití této architektury a následné omezení přístupu k tabulkám dokáže ochránit data proti neoprávněným změnám.

Jakákoliv změna dat, ať už je to vložení nových dat anebo úprava stávajících, je řešena pomocí uložených procedur. Tyto procedury ve většině případů, kdy je potřebná změna ve více tabulkách, využívají transakce. Transakce obsahuje skupinu příkazů, které převedou data v databázi z jednoho konzistentního stavu do stavu druhého, opět konzistentního. Jedná se tedy o logickou a zároveň atomickou část databázového dotazu, která se buď provede celá, anebo se neprovede žádný dotaz, který daná transakce obsahuje.

Pro přístup k datům jsou výhradně využívány pohledy. Tyto pohledy nekopírují tabulkovou strukturu, ale ve většině případů spojují několik souvisejících tabulek. Zároveň omezují (neobsahují dané sloupce tabulky) přístup k některým sloupcům tabulek, které jsou využívány pouze pro aplikační logiku na úrovni databáze.

## 5.2 E-R diagram



## 5.3 Základní popis tabulek

Všechny tabulky mají prefix ‚tbl_' z důvodů zabránění kolize s názvy pohledů. Tento prefix je použit právě u tabulek, protože s tabulkami se nikde v celé aplikaci nepracuje. Pokud není uvedeno dále v popisu jinak, tak všechny tabulky splňují třetí normální formu.

- **tbl_seo** – tato tabulka uchovává informace, potřebné k implementaci SEO optimalizace, jako je hezká URL adresa, krátký popis stránky a klíčová slova. Tuto tabulku následně využívají tabulky novinky, kategorie, výrobci a software.
- **tbl_news** – tabulka s novinkami, které informují o aktuálním dění v internetovém obchodě. Obsahuje informace o názvu novinky, její text, datum, kdy byla vytvořena, a příznak viditelnosti.
- **tbl_categories** – kategorie e-shopu, které umožňují použít pouze jednoúrovňovou strukturu. Tabulka obsahuje informace o názvu kategorie a příznak viditelnosti.
- **tbl_producers** – výrobci, od kterých prodáváme software. Můžeme zde uložit název a popis tohoto výrobce, jeho internetové stránky, e-mail a případně i logo.
- **tbl_software** – obsahuje informace o jednotlivém software, který je anebo byl v nabídce. Její atributy jsou název software, krátký popis zobrazovaný v seznamu a dlouhý popis, který se zobrazí v detailu. Dále cenu bez DPH, datum vložení tohoto software do nabídky využívaný při zobrazení nejnovějšího software, logo tohoto software a příznak viditelnosti a archivace. Archivovaný software je ten, který byl smazán, ale je nutné ho udržovat v databázi díky návaznosti na jednotlivé objednávky. Tabulka je napojena na tabulky kategorie, výrobci a soubory.
- **tbl_files –** tato tabulka obsahuje informace o souborech uložených na filesystému. Zde máme informace o originálním názvu souboru a názvu, pod nímž je soubor aktuálně uložen, dále informaci o velikosti a typu tohoto souboru.
- **tbl_orders_software** – toto je spojovací tabulka mezi tabulkou software a objednávkami. Obsahuje informace o počtu software a jeho ceně v době koupě. Tabulka je napojena na tabulky software a objednávky.
- **tbl_orders –** toto je tabulka uchovávající informace o objednávce jako celku, jako je číslo objednávky, datum vytvoření a stav této objednávky. Dále je propojena s uživatelem, který tuto objednávku vytvořil, a v poslední řadě i fakturačními údaji zákazníka.

- **tbl_orders_state –** tabulka obsahující jednotlivé stavy, kterých může nabýt objednávka.
- **tbl_customers –** tato tabulka obsahuje fakturační informace, které se vážou ke konkrétní objednávce. Obsahuje údaje firma, její IČ a DIČ, jméno a příjmení zákazníka a adresu (ulice, číslo popisné, město a PSČ). Tato tabulka není ve třetí normální formě, jelikož existuje závislost mezi městem a PSČ.
- **tbl_roles –** seznam rolí, které jsou využívány v aplikaci. Obsahuje název role a speciální název, který používá interně aplikace.
- **tbl_users –** tabulka se seznamem registrovaných uživatelů, kteří mají přístup do zabezpečené sekce. Obsahuje přihlašovací e-mail, heslo a odkaz na roli, ve které je daný uživatel zařazen. Dále zde najdeme datum registrace, příznak aktivity účtu a nakonec i validační kód pro účty, které ještě nejsou ověřeny a aktivovány.
- **tbl_settings** – tato tabulka obsahuje základní nastavené internetového obchodu, jako je název, podnázev, údaje v patičce stránky a kontaktní informace. Dále texty obchodních a reklamačních podmínek a návod jak nakupovat. Tato tabulka také není ve třetí normální formě, jelikož existuje závislost mezi městem a PSČ v kontaktních informacích a dále existuje závislost mezi názvem banky a jejím kódem.

## 5.4 Pohledy

Využívané databázové pohledy mimo toho, že zvyšují bezpečnost aplikace, jsou používány i pro spojování souvisejících tabulek. Ve většině případů by totiž toto spojení bylo potřeba udělat přímo v aplikaci, což by zbytečně zvyšovalo množství potřebného kódu. Tento kód by byl následně rozeset po celé aplikaci a při potřebě změny by se musely upravit všechny SQL dotazy, které toto spojení vyžadují.

Nyní aplikace pouze využívá tyto pohledy, na které následně aplikuje již jednoduché omezující podmínky pro výběr určitých dat a pro stránkování.

**Příklad pohledu, který spojuje tabulku kategorií a SEO informací:**

```
CREATE VIEW categories
AS
SELECT c.category_id AS id, c.name, c.visible, s.url AS seo_url,
   s.keywords AS seo_keywords, s.description AS seo_description
FROM tbl_categories c
JOIN tbl_seo s ON c.seo_id = s.seo_id
ORDER BY c.name
```

## 5.5 Uložené procedury

Uložená procedura je jeden z typů databázových objektů. Obsahuje programový kód, který se po spuštění této procedury vykoná nad daty uloženými v databázi. V ukázkové aplikaci byly uložené procedury použity pro jakoukoliv změnu dat v tabulkách.

**Příklad uložené procedury pro vložení nové kategorie:**

```
CREATE PROCEDURE category_add(
   p_name VARCHAR(30), p_seo_url VARCHAR(50),
   p_seo_keywords VARCHAR(255), p_seo_description VARCHAR(255))
BEGIN
   DECLARE EXIT HANDLER FOR SQLEXCEPTION
      CALL raise_error('Chyba při vkládání kategorie');
   START TRANSACTION;

   INSERT INTO tbl_seo (url, keywords, description)
   VALUES (p_seo_url, p_seo_keywords, p_seo_description);

   INSERT INTO tbl_categories (NAME, seo_id)
   VALUES (p_name, LAST_INSERT_ID());

   COMMIT;
END
```

**Příklad uložené procedury pro úpravu existující kategorie:**

```
CREATE  PROCEDURE category_edit(
   p_id INT(11), p_name VARCHAR(30), p_visible TINYINT(1) UNSIGNED,
   p_seo_url VARCHAR(50), p_seo_keywords VARCHAR(255),
   p_seo_description VARCHAR(255))
BEGIN
   DECLARE EXIT HANDLER FOR SQLEXCEPTION
      CALL raise_error('Chyba při editaci kategorie');
   START TRANSACTION;

   UPDATE tbl_categories
   SET
      NAME = p_name,
      visible = p_visible
   WHERE category_id = p_id;

   UPDATE tbl_seo
   SET
      url = p_seo_url,
      keywords = p_seo_keywords,
      description = p_seo_description
   WHERE seo_id = (
      SELECT seo_id
      FROM tbl_categories
      WHERE category_id = p_id
   );

   COMMIT;
END
```

MySQL ve verzi 5.1 bohužel stále neumožňuje vyvolání uživatelské chyby mimo databázi, která by v transakci mohla nastat. Proto byla vytvořena pomocná

procedura nazvaná raise_error, která v parametru přijímá text chybové hlášky. Tato procedura pouze zruší právě probíhající transakci a následné se pokouší uložit chybovou hlášku do číselné proměnné. Takto vzniklá chyba je následně předána aplikaci, která má konečně možnost nějak zareagovat.

**Uložená procedura sloužící pro vyvolání aplikační chyby**

```
CREATE PROCEDURE raise_error(p_message varchar(50))
BEGIN
   DECLARE v_msg int(1);

   ROLLBACK;
   SET v_msg = p_message;
END;
```

## 5.6 Indexy

Veškeré používané indexy u jednotlivých tabulek byly vytvořeny automaticky samotnou databází. Jedná se především o sloupce primárních a cizích klíčů. Indexy byly automaticky také vytvořeny u sloupců obsahujících unikátní data, jako je registrační emailová adresa u uživatelů nebo SEO adresa v tabulce tbl_seo. Žádné jiné uživatelsky definované indexy použity nebyly, jelikož by rychlost aplikace dále asi nezvýšily.

# 6 Implementace webové části

Webová aplikace je kompletně postavena na Nette Frameworku a využívá všech jeho výhod, která byla dostupná v momentě programování této aplikace. Díky velice rychlému vývoji a časté změně API může být v době čtení této práce dostupná novější verze frameworku, která ale již nemusí být s aplikací kompatibilní.

## 6.1 Adresářová struktura

Nette Framework doporučuje určité rozdělení aplikace do podadresářů, nicméně toto doporučení není nutné dodržovat. Ukázková aplikace obsahuje tři základní adresáře (application, library a public), které svým rozdělením dokážou významným způsobem zvýšit bezpečnost a ochránit tak zdrojové a konfigurační soubory.

### Adresář Application

Tento adresář obsahuje základní zaváděcí soubor, konfiguraci aplikace a dále další podadresáře obsahující konkrétní implementaci jednotlivých částí internetového obchodu (jako jsou objekty pro přístup do databáze, formuláře, opakovaně využívané komponenty, presentery a šablony).

### Adresář Library

Obsahem tohoto adresáře jsou knihovny využívané v aplikaci. V případě vytvořené aplikace to je Nette Framework, databázová knihovna Dibi a knihovna PHPMailer zjednodušující práci s odesíláním e-mailů.

### Adresář Public

Toto je jediný adresář, který je dostupný z webového prohlížeče. Pro správnou funkčnost musí být v konfiguraci webového serveru nastavena direktiva definující výchozí adresář dokumentů na tuto složku. Tato složka obsahuje výchozí soubor index.php a soubory obrázků, stylů a skriptů.

## 6.2 Rozdělení do modulů

Celá aplikace je kvůli přehlednosti rozdělena do tří modulů. Základní funkčnost a implementaci veřejné části zajišťuje FrontModule. Další dva moduly implementují neveřejnou část, konkrétně CustomerModule sekci pro přihlášeného zákazníka a AdminModule administrátorskou část.

Toto rozdělení reflektuje i adresářová struktura, kde ve složce presenters a templates najdeme podsložky pojmenované právě podle názvů jednotlivých modulů.

## 6.3 Inicializace aplikace

Zpracování stránky vždy začíná výchozím souborem index.php, který je umístěn ve složce Public. Tento soubor pouze definuje konstanty k ostatním adresářům aplikace a poté předá řízení zaváděcímu souboru bootstrap.php. Soubor bootstrap.php je již bezpečně umístěný v adresáři Application a stará se o inicializaci aplikace. Nejprve je načten Nette Framework a databázový layer Dibi. Následně se povolí Laděnka, což je komponenta, která umí zobrazit chybové hlášky uživatelsky příjemnějším způsobem. Dále proběhne načtení konfiguračního souboru a připojení k databázi. Následuje definice routovacích pravidel, která umožňují vytvářet vlastní podobu URL adres, a definice přístupových práv k jednotlivým částem aplikace. Po inicializaci všech těchto částí je předáno řízení frameworku. Ten následně vybere určitou část aplikace (presenter), který se postará o obsloužení požadavku.

## 6.4 Konfigurace

Konfigurační soubor aplikace config.ini se nachází v adresáři Application. Struktura tohoto souboru je stejná jako v klasickém ini souboru a může být rozdělena do několika částí. Jednotlivé části mohou dědit některou z předchozích úrovní, díky čemuž se dá snadno odstranit případná duplicita informací. Pomocí konfigurace je možné do aplikace deklarativně zapojit i různé komponenty, které mají určité předem definované rozhranní. Proto je například možné bez jediného zásahu do kódu aplikace vyměnit autentizační komponentu za komponentu jinou, která bude ověřovat uživatele jiným způsobem a třeba i na jiném systému.

V aplikaci jsou využity tři části a to základní konfigurace (common), konfigurace v produkčním prostředí (production) a konfigurace určená pro vývoj (development). Díky dědičnosti jednotlivých úrovní je pro vývojovou úroveň zapnuto pouze profilování SQL dotazů a ostatní informace jako údaje pro připojení k databázi jsou automaticky zděděny.

**Konfigurační soubor aplikace config.ini**

```
[common]
set.date-timezone = "Europe/Prague"
set.iconv-internal_encoding = "%encoding%"
set.mbstring-internal_encoding = "%encoding%"

service.Nette-Security-IAuthenticator = UserModel
service.Nette-Security-IAuthorizator  = Permission

[production < common]
database.driver = mysql
database.host = localhost
database.username = root
database.password = root
database.database = swshop
database.charset = utf8
```

```
database.lazy = TRUE

[development < production]
database.profiler = TRUE
```

## 6.5 SEO adresy a routování

Pro vytváření a následné zpracování SEO adres je možné využít služeb modulu mod_rewrite, který je standardně dodáván s webovým serverem Apache. Tento modul umí přepisovat používané URL adresy na adresy, které jsou snadněji v aplikaci zpracovatelné. Využívaná přepisovací pravidla jsou pomocí regulárních výrazů napsána a uložena v uživatelském konfiguračním souboru webového serveru (soubor ‚.htaccess' v kořeni aplikace). Tento způsob má ale několik nevýhod. Zaprvé neexistuje automatická synchronizace mezi adresami zapsanými v jednotlivých šablonách a pravidly v souboru ‚.htaccess'. Při změnách je velká pravděpodobnost, že se někde něco zapomene upravit. Druhou nevýhodou je skutečnost, že pokud webový server nepodporuje nebo má vypnutý modul mod_rewrite, tak aplikace prostě nebude fungovat.

Nette Framework proto přichází s řešením, které odstraňuje obě dvě výše zmíněné nevýhody. Do souboru ‚.htaccess' je zapsáno pouze jednoduché pravidlo přepisující všechny požadavky, které nejsou určeny pro konkrétní soubor (obrázek, soubor s kaskádovými styly, …), do souboru index.php. V inicializační části aplikace jsou následně vytvořena routovací pravidla, která by odpovídala pravidlům v souboru ‚.htaccess'. Tato pravidla jsou obousměrná, tedy umějí adresy jak zpracovávat, tak i generovat.

**Soubor ‚.htacces' obsahující pouze jednoduché přepisovací pravidlo**

```
RewriteEngine on
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule !\.(js|ico|gif|jpg|png|css|zip)$ index.php [L]
```

**Příklad jedné routy pro obsluhu požadavku**

```
$app = Environment::getApplication();
$router = $app->getRouter();

$router[] = new Route('<category>/<software>/', array(
   'module' => 'Front',
   'presenter' => 'Default',
   'action' => 'software'
));
```

Následné vytvoření adresy, která bude například využita pro odkaz v HTML šabloně, se již přenechá na frameworku. Ten obsahuje metodu, které je parametrem předána informace o části aplikace, která se bude vykonávat po kliknutí na odkaz, a tato metoda poté vrací odpovídající URL adresu. Pokud by bylo potřeba někdy

v budoucnu změnit podobu URL adres, tak se tato změna provede pouze na jednom místě v definici rout a aplikaci to žádným způsobem neovlivní. Navíc při nepodpoře přepisovacího modulu na serveru je možné aplikaci stále využívat, pouze nebude používána definovaná podoba URL adres a všechny informace budou předávány v parametrech za otazníkem.

**Příklad vytvoření odkazu ve stránce ($sw je objekt obsahující informace o jednom softwaru)**

```
<a href="{link Default:software $sw->seo_url}"
   title="Přejít na produkt: {$sw->name}">{$sw->name}</a>
```

## 6.6 Zpracování stránky

Pro zpracování stránek se využívají Controllery, což jsou třídy implementující určité rozhranní. V Nette Frameworku se tyto třídy nazývají presentery. Presenter poskytuje přístup k událostem, které se vyvolávají v určitých fázích obsluhování požadavku. Do jednotlivých fází je možné zasáhnout přepsáním určité zděděné metody, do které se implementuje vlastní požadovaná funkčnost. Mimoto presenter poskytuje metody pro přístup k často používaným operacím, které byly v rámci frameworku nejen objektově zapouzdřeny, ale obsahují i dodatečnou funkčnost.

## 6.7 Autorizace

Autorizace, kam smí který uživatel vstoupit je v aplikaci řešena globálně za využití Nette Frameworku. Při inicializaci aplikace jsou nejdříve vytvořeny role a zdroje. Následně jsou jednotlivým rolím povoleny akce, které smějí s danými zdroji provádět. V aplikaci jsou pouze využívány akce view označující, že uživatel s danou rolí může k určitému zdroji přistoupit a vykonávat na něm veškeré operace.

**Definice přístupových práv**

```
$acl  = Environment::getService('Nette\Security\IAuthorizator');

$acl->addRole('guest');
$acl->addRole('customer', 'guest'); // role customer dědí práva
$acl->addRole('admin', 'customer');

$acl->addResource('resFront');
$acl->addResource('resCustomer');
$acl->addResource('resAdmin');

$acl->allow('guest', 'resFront', 'view');
$acl->allow('customer', 'resCustomer', 'view');
$acl->allow('admin', 'resAdmin', 'view');
```

Ovladače, ke kterým se omezuje přístup, obsahují kód, který ověřuje, zdali má daný uživatel právo vstoupit do této části aplikace. Při ověřování se již nemusí uvádět název role, jelikož tento název se bere automaticky z objektu uživatele. Při nesplnění minimálních oprávnění je uživatel přesměrován na přihlašovací stránku.

**Ověření přístupového práva v události ovladače stránky**

```
protected function startup() {
   parent::startup();
   $user = Environment::getUser();
   if (! $user->isAllowed('resAdmin', 'view')) {
      $this->redirect(':Front:Login:');
   }
}
```

## 6.8 Modely – objekty pro přístup k databázi

Pro manipulaci s daty se využívají objekty přímo k tomu určené, tzv. modely. Výhodou tohoto přístupu je oddělení práce s daty od okolního kódu, který je tak soustředěn pouze do jednoho místa aplikace. Ve většině případů obsahují modely pouze statické metody, což umožňuje jejich jednoduché použití v rámci celé aplikace.

**Ukázka modelu pro správu kategorií**

```
class CategoryModel extends Object {
   public static function getAll($visible = TRUE) {
      $sql = 'SELECT * FROM categories';
      $sql .= $visible ? ' WHERE visible = 1' : '';
      return dibi::dataSource($sql);
   }

   public static function get($id) {
      $sql = 'SELECT * FROM categories WHERE ';
      $sql .= is_numeric($id) ? 'id = %i' : 'seo_url = %s';
      return dibi::query($sql, $id)->fetch();
   }

   public static function add(array $data) {
      $sql = 'CALL category_add(%s, %s, %sn, %sn)';
      dibi::query($sql, $data['name'], $data['seo_url'],
         $data['seo_keywords'], $data['seo_description']);
   }

   public static function edit(array $data) {
      $sql = 'CALL category_edit(%i, %s, %i, %s, %sn, %sn)';
      dibi::query($sql, $data['id'], $data['name'],
         data['visible'], $data['seo_url'], data['seo_keywords'],
         $data['seo_description']);
   }

   public static function remove($id) {
      $sql = 'CALL category_remove(%i)';
      dibi::query($sql, $id);
   }

   public static function save($data) {
      if (empty($data['id'])) {
         self::add($data);
      } else {
         self::edit($data);
      }
   }
}
```

## 6.9 Formuláře

V aplikaci se velmi často objevují formuláře pro získávání nejrůznějších dat od uživatele. Například přihlašovací formulář, registrační formulář, formulář pro zaslání zapomenutého hesla nebo formulář pro získání fakturačních údajů při objednávce. Také administrační část obsahuje velké množství různých formulářů.

Nette Framework poskytuje objektové rozhranní pro práci s formuláři. Vytvoření formuláře je možné provést několika způsoby, například pomocí třídy implementující rozhranní AppForm, která následně definuje jednotlivé formulářové pole i s jejich validačními pravidly. Právě validační pravidla jsou silnou stránkou těchto formulářů. Framework umí automaticky validovat na straně serveru i na straně klienta podle našich předem vytvořených pravidel a předá řízení obslužnému kódu, který s daty manipuluje, pouze v případě, že formulář je validní.

Při vytváření formulářů bylo potřeba zajistit, aby všechny formuláře měli implementovánu nějakou vlastní základní funkčnost. K tomuto účelu byla vytvořená abstraktní třída BaseAppForm odvozená od třídy AppForm. Tato třída automaticky registruje každý formulář jako komponentu, kde název je odvozen od názvu implementující třídy, a vytváří callback na obslužnou událost, která nastane po odeslání a úspěšné validaci formuláře.

**Kód pro vytvoření registračního formuláře:**

```
class RegisterForm extends BaseAppForm
{
   public function __construct($parent)
   {
      parent::__construct($parent);
      $this->setRenderer(new ShopRenderer);

      $this->addText('email', 'E-mail:')
         ->addRule(Form::FILLED, 'Zadejte e-mail')
         ->addRule(Form::EMAIL, 'Zadaný e-mail není správný')
         ->addRule('UserValidator::uniqueEmail',
            'Na zadaný e-mail už je registrován jiný uživatel');
      $this->addPassword('password', 'Heslo:')
         ->addRule(Form::FILLED, 'Vyplňte heslo')
         ->addRule(Form::MIN_LENGTH,
            'Heslo musí mít nejméně 4 znaky', 4);
      $this->addPassword('password_check', 'Heslo znovu:')
         ->addRule(Form::EQUAL,
            'Hesla nejsou stejná', $this['password']);

      $this->addSubmit('submitButton', 'Registrovat');
   }
}
```

Renderování výsledného formuláře je automatické, ale dá se ovlivnit pomocí několika vlastností. Pokud by ale i někomu tohle nestačilo, je možné napsat si renderer vlastní, což je v podstatě třída implementující rozhranní IFormRenderer.

V aplikaci bylo využito obou způsobů úpravy výsledného vykresleného kódu. Pro administrační část stačilo pouze pomocí vlastností upravit HTML tagy, do kterých se formulář vykreslí. Pro veřejnou část aplikace byl napsán vlastní renderer z důvodů větší kontroly nad výsledným kódem, který zároveň přehledněji zobrazuje chybové hlášky. Bohužel tento renderer zatím neumí validovat formuláře na klientské straně.

**Obrázek 4: Registrační formulář**


## 6.10 Komponenty

Komponenta je část aplikace, která zapouzdřuje určitou funkčnost do jednoho balíčku. Je nezávislá na ostatním kódu a schopná pracovat samostatně. Výhoda komponenty je také v jejím opakovaném využití na více místech aplikace bez duplicitního psaní kódu.

V Nette Frameworku existují dva druhy komponent – nevykreslitelné, které pouze zapouzdřují určitou funkčnost, a vykreslitelné, které umí navíc zobrazit svůj stav (vygenerovat HTML kód). [15]

V implementaci aplikace je využito obou dvou typů. Mezi komponentu, která neumí zobrazit svůj stav, patří vnitřní implementaci nákupního košíku. Tato komponenta poskytuje definované rozhranní pro přidání položky, změnu jejího množství nebo smazání, pro výmaz všech produktů z košíku i pro získávání jednotlivých produktů.

Jako vykreslitelné komponenty byly realizovány všechny ostatní, mezi něž patří levé menu s výpisem kategorií, které umí automaticky vygenerovat URL adresu odkazu a také umí zvýraznit vybranou položku bez jediného vnějšího zásahu do komponenty. Dále zobrazení stavu o aktuálně přihlášeném, resp. nepřihlášeném uživateli. Tato komponenta umí pracovat s právy internetového obchodu, a proto zobrazuje jiné odkazy administrátorovi a jiné zákazníkovi. Další univerzální komponentou je zobrazení stavu nákupního košíku zobrazující počet položek a jejich celkovou cenu. A mezi poslední a velice užitečnou komponentu se řadí drobečková navigace, která zobrazuje aktuální úroveň zanoření ve struktuře webu.

**Obrázek 5: Drobečková navigace**


## 6.11 Šablony

Webové stránky byly vytvořeny ve striktní normě XHTML 1.1. Toto je zatím nejpřísnější norma, která nepovoluje využívání zastaralých HTML značek pro definici vzhledu dokumentu. Veškerý obsah je sémanticky vložen do správných tagů, čímž je umožněno vyhledávacím robotům správně rozpoznat obsah. Ve výsledku to může zajistit lepší pozici ve vyhledávačích. Pro definici vzhledu jsou použity kaskádové styly, které jsou většinou umístěny do zvláštního externě přilinkovaného souboru.

### CurlyBrackets filtr

Jednou z výhod použití MVC architektury je oddělení aplikačního kódu od HTML šablon určených pro výsledné zobrazení dat. PHP samo o sobě se také používá k tvorbě šablon, ale aby byla práce se šablonami ještě jednodušší a výsledný kód přehlednější, přichází Nette Framework s vlastním filtrem CurlyBrackets. Tento filtr má podobnou syntaxi jako šablonovací systém Smarty, je ale rozšířen o mnoho dalších vlastností. Veškeré šablony jsou při prvním požadavku „zkompilovány“ do klasického PHP kódu. Děje se tak díky tomu, že zpracování šablony při každém požadavku by velice snižovalo výkon celé aplikace.

Při vypisování jakýchkoliv proměnných je doporučeno tyto proměnné zbavit nežádoucích znaků. Při opomenutí nebo nedodržení tohoto principu je možné vpravit do našich stránek cizí HTML kód. Tato metoda útoku se nazývá XSS (Cross Site Scripting). Bohužel ošetřování každé proměnné poměrně prodlužuje výsledný kód šablony, šablona se stává nepřehlednou a je pravděpodobné, že se na nějakou proměnnou stejně zapomene [17].

**Část šablony vytvořená klasicky pomocí PHP:**

```
<ul>
<?php foreach ($items as $item): ?>
   <li><a href="<?php echo htmlspecialchars($item->href) ?>">
      <?php echo htmlspecialchars($item->name) ?></a></li>
<?php endforeach ?>
</ul>
```

Nette Framework proto přichází se speciální syntaxí využívající složené závorky. Jakákoliv proměnná, pokud není určeno jinak, je automaticky ošetřena proti XSS útoku. V šabloně je navíc možné jednoduše využívat základní řídící konstrukce, jako jsou podmínky a cykly. Některé z nich navíc mají rozšířené možnosti oproti použití v klasickém PHP.

**Stejná část šablony jako výše, pouze napsaná pomocí filtru CurlyBrackets v Nette:**

```
<ul>
{foreach $items as $item}
   <li><a href="{$item->href}">{$item->name}</a></li>
{/foreach}
</ul>
```

### Helpery

Občas je potřeba upravit výstup určité proměnné do formátu čitelnějšího pro lidi, například proměnnou obsahující měnu zformátovat podle národního prostředí a přidat k ní symbol této měny. Upravovat tyto proměnné již v presenteru není moc efektivní a navíc se tím kód jednotlivých metod zbytečně prodlužuje. Proto je ve frameworku implementována podpora pro tzv. helpery, neboli pomocníky. Helper je funkce nebo metoda objektu, která v argumentu přijme původní hodnotu upravované proměnné a vrátí hodnotu upravenou.

V implementaci aplikace jsou využity helpery především pro výpis jednotlivých cen a to jak bez DPH, tak s DPH. K cenám je taktéž přidán symbol měny pro českou korunu.

**Helper pro výpis měny:**

```
class ShopHelper extends Object
{
   public static function currency($value)
   {
      // \xc2\xa0 je nedělitelná mezera v kódování UTF-8
      return str_replace(" ", "\xc2\xa0",
         number_format($value, 0, "", " ")) . ",-\xc2\xa0Kč";
   }

   public static function currencyVat($value)
   {
      $value *= 1.19; //DPH
      return str_replace(" ", "\xc2\xa0",
         number_format($value, 0, "", " ")) . ",-\xc2\xa0Kč";
   }
}
```

Helpery jsou následně využívány tak, že při výpisu proměnné v šabloně je za tuto proměnnou napsána svislá čára a dále jméno helperu. V případě že helper přijímá ještě nějaké další dodatečné parametry, tak tyto parametry jsou následně napsány za dvojtečku.

**Použití helperu pro upravení proměnné použité v šabloně:**

```
Cena bez DPH: <strong>{$price|currency}</strong>
Cena s DPH: <strong>{$price|currencyVat}</strong>
```

## 6.12 Administrační část

Administrační část slouží pro kompletní správu internetového obchodu. Tato část byla optimalizována tak, aby co nejvíce místa na stránce zabraly užitečné informace. Z tohoto důvodu hlavička pouze obsahuje nadpis a údaje o přihlášeném uživateli. Pod hlavičkou je nabídka s jednotlivými sekcemi, které lze spravovat a dále se už nachází přímo samotný obsah stránky.

**Obrázek 6: Layout administrační části**


V administrační části jsou ve velkém množství využívány tabulky, které přehledným způsobem zobrazují informace. Tyto tabulky jsou vylepšeny o podporu stránkování a řazení podle jednotlivých sloupců. Pokud navíc klientský prohlížeč podporuje JavaScript, tak je pro stránkování a řazení využívána technologie AJAX. Při použití této technologie se nenačítá celá stránka znovu, ale ze serveru je poslána pouze tabulka, která ve stránce nahradí tu stávající. Celý tento proces aktualizace stránky je navíc doplněn o krátkou grafickou animaci.

# 7 Závěr

Cílem této práce bylo vytvořit internetový obchod s prodejem software. Aplikace byla naprogramována s pomocí moderního a výkonného Nette Frameworku za využití jazyka PHP a databázového serveru MySQL. Design webu byl vytvořen s ohledem na přehlednost, přístupnost a především nenáročnost, který ale pro své použití zcela vyhovuje. Webová aplikace je validní podle dnešních nejpřísnějších standardů a je optimalizována pro vyhledávače. Bohužel tato optimalizace nebyla otestována v reálném provozu a tak se jenom můžeme domnívat, na jaké pozici by se internetový obchod ve vyhledávačích umístil.

Vytvořený internetový obchod by měl být po všech stránkách funkční a spolehlivý, tudíž by neměl být problém nasadit ho do reálného provozu. Umožňuje spravovat v administrační části vše, co je potřeba. V obchodě je implementováno zatřídění softwaru do jednotlivých kategorií, včetně možnosti stránkování a řazení podle různých kritérií. Dále je tu možnost použití snadného vyhledávání podle názvu zboží. Zákazníci mají možnost se zaregistrovat a následně po ověření pravosti účtu nakupovat. Po zaplacení převodem na bankovní účet a potvrzení této změny administrátorem v obchodu mají možnost přímého stažení objednaného software. V jejich zákaznické sekci se jim navíc evidují informace o jednotlivých objednávkách.

V dnešní době by ale tento internetový obchod asi nebyl konkurenceschopný v oblasti nabízených služeb a bylo by potřebné implementovat další funkčnost. Nejcitelnější chybějící částí je integrace platebního systému přímo do aplikace. Dále by se vyplatilo mít možnost označit software určitými stavy, jako v akci nebo ve výprodeji. Ideální také není pouze jednoúrovňová struktura kategorií a při velkém množství zboží v katalogu by se internetový obchod stal nepřehledný. V neposlední řadě by tu mohla být možnost informovat registrované zákazníky o novinkách a akcích v obchodě, a to buď pomocí implementace RSS služby anebo zasíláním informací přímo na jejich e-mail. Díky modulární struktuře by ale neměl být problém doimplementovat jakoukoliv funkčnost.

Pro mě, jako autora práce, bylo toto téma velice zajímavé. Dozvěděl jsem se mnoho nových informací a získal řadu zkušeností, které v budoucnu určitě využiji. Především téma SEO optimalizace stránek pro vyhledávače bylo pro mě nové. Naučil jsem se, jak by webové stránky měly vypadat z pohledu zdrojového HTML kódu a co bych naopak neměl používat. Při návrhu databázové struktury jsem využil mnoho znalostí získaných ve škole a jejich použití na jiném typu databáze bylo pouze otázkou naučení se pár nových pravidel a klíčových slov. Pro mě

nejzajímavější a zároveň také asi nejobtížnější bylo naučit se využívat Nette Framework. Tento moderní objektově orientovaný framework umožňuje tolik věcí, že jsem nebyl schopen za celou dobu vývoje internetového obchodu využít všech jeho vlastností. Vlastně si myslím, že jsem využil jen malou část toho, co skutečně nabízí. Nicméně jsem rozhodnut naučit se tento framework využívat v maximální možné míře a implementovat v něm velkou část v budoucnu vytvářených webových stránek

Tato práce mě ujistila v tom, že jsem schopný vytvářet kvalitní webové aplikace, které jsou navíc naprogramovány moderním a univerzálním způsobem za využití všech výhod objektově orientovaného programování.

# 8 Použitá literatura


[1] Internetový obchod. *Wikipedie, otevřená encyklopedie* [online]. 2009 [cit. 2009-05-09]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Internetový_obchod>.

[2] ŘÍHA, Ondřej. *E-shop jako alternativní distribuční kanál firmy*. [s.l.], 2008. 52s. Univerzita Pardubice, Fakulta elektrotechniky a informatiky. Bakalářská práce.

[3] SMIČKA, Radim. *Optimalizace pro vyhledávače - SEO : Jak zvýšit návštěvnost webu*. Dubany : Jaroslava Smičková, 2004. 120 s. Dostupný z WWW: <http://seo.jasminka.cz/seo-kniha.pdf>. ISBN 80-239-2961-5.

[4] Internetový vyhledávač. *Wikipedie, otevřená encyklopedie* [online]. 2009 [cit. 2009-05-09]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Vyhledávač>.

[5] MELAGOVÁ, Iva. *Podpora internetového obchodování – SEO optimalizace*. [s.l.], 2007. 70 s. Univerzita Pardubice, Fakulta ekonomicko-správní. Diplomová práce.

[6] *Seo optimalizace* [online]. c2007 [cit. 2009-05-09]. Dostupný z WWW: <http://www.seo-optimalizace.info/>.

[7] *PHP: Hypertext Preprocessor* [online]. c2001-2009 [cit. 2009-05-09]. Dostupný z WWW: <http://php.net/>.

[8] PHP. *Wikipedie, otevřená encyklopedie* [online]. 2009 [cit. 2009-05-09]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Php>.

[9] KOFLER, Michael. *Mistrovství v MySQL 5 : Kompletní průvodce webového vývojáře*. Brno : Computer Press, 2007. 808 s. ISBN 978-80-251-1502-2.

[10] MySQL. *Wikipedie, otevřená encyklopedie* [online]. 2009 [cit. 2009-05-09]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/MySQL>.

[11] CROFT, Jeff, LLOYD, Ian, RUBIN, Dan. *Mistrovství v CSS : Pokročilé techniky pro webové designéry a vývojáře*. Brno : Computer Press, 2007. 416 s. ISBN 978-80-251-1705-7.

[12] HOLZNER, Steven. *Mistrovství v Ajaxu*. Brno : Computer Press, 2007. 592 s. ISBN 978-80-251-1850-4.

[13] Framework. *Wikipedie, otevřená encyklopedie* [online]. 2009 [cit. 2009-05-09]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Framework>.

[14] Model-view-controller. *Wikipedie, otevřená encyklopedie* [online]. 2009 [cit. 2009-05-09]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/MVC>.

[15] Nette Foundation. *Nette Framework* [online]. c2008-2009 [cit. 2009-05-09]. Dostupný z WWW: <http://nettephp.com/cs/>.

[16] GRUDL, David. Nette Framework: zvyšte svoji produktivitu. *Zdroják* [online]. 2009 [cit. 2009-05-09]. Dostupný z WWW: <http://zdrojak.root.cz/clanky/nette-framework-zvyste-svoji-produktivitu/>.

[17] GRUDL, David. Nette Framework: Chytré šablony. *Zdroják* [online]. 2009 [cit. 2009-05-09]. Dostupný z WWW: <http://zdrojak.root.cz/clanky/nette-framework-chytre-sablony/>.

[18] Nette Foundation. *Dibi* [online]. c2008-2009 [cit. 2009-05-09]. Dostupný z WWW: <http://dibiphp.com/cs/>.

[19] GRUDL, David. Dibi - pokrokový databázový layer. *La Trine* [online]. 2006 [cit. 2009-05-09]. Dostupný z WWW: <http://latrine.dgx.cz/dibi-pokrokovy-databazovy-layer>.

[20] GRUDL, David. Téměř v cíli: dibi 0.9b. *La Trine* [online]. 2007 [cit. 2009-05-09]. Dostupný z WWW: <http://latrine.dgx.cz/temer-v-cili-dibi-0-9b>.

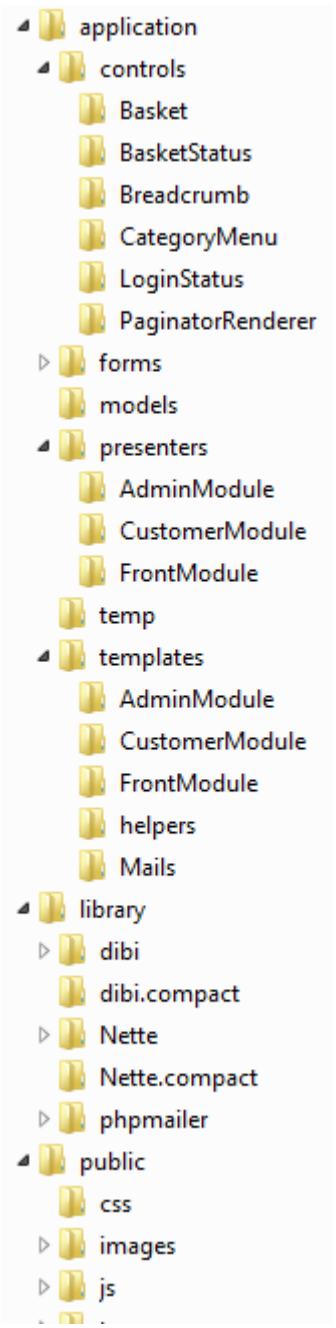
[21] GRUDL, David. MVC paradox a jak jej řešit. *PhpFashion* [online]. 2009 [cit. 2009-05-09]. Dostupný z WWW: <http://phpfashion.com/mvc-paradox-a-jak-jej-resit>.

[22] *JQuery: The Write Less, Do More, JavaScript Library* [online]. c2009 [cit. 2009-05-09]. Dostupný z WWW: <http://jquery.com/>.

[23] *Codeworx Technologies : PHPMailer* [online]. c1999-2009 [cit. 2009-05-09]. Dostupný z WWW: <http://phpmailer.codeworxtech.com/>.

## Příloha A – Struktura adresářů na přiloženém CD

Přiložené CD obsahuje přímo v hlavní struktuře celou aplikaci, kterou je možné zkopírovat do produkčního prostředí. Mimoto tu najdeme soubor readme.txt, který obsahuje stručný popis aplikace, soubor install.txt popisující instalaci a soubor install_db.sql, obsahující vyexportovanou strukturu databáze včetně několika ukázkových dat.




Složka application obsahuje logiku a implementaci internetového obchodu. Tato složka dále obsahuje podsložky controls s komponentami, forms s definicemi všech používaných formulářů a models se třídami komunikujícími s databází. Dále zde najdeme podsložku presenters s ovladači jednotlivých stránek, která je ještě dále rozdělena podle modulů, podložku temp, kde se uchovávají cachované informace, především „zkompilované“ šablony, a jako poslední tu najdeme složku templates s vlastní definicí šablon, která je taktéž dále rozdělena podle modulů a logických částí

Složka library obsahuje knihovny použité při implementaci aplikace. Jedná se o Nette Framework, databázovou vrstvu Dibi a knihovnu pro odesílání emailů PHPMailer. Složky s koncovkou compact obsahují ty samé knihovny jako bez této přípony, pouze jsou určeny pro produkční prostředí. Veškeré soubory konkrétní knihovny jsou totiž minimalizovány do jednoho kompaktního souboru.

Poslední složka s názvem public obsahuje soubory dostupné z prohlížeče. Tedy kaskádové styly, obrázky a skripty. Dále zde najdeme ještě složku temp, do které se ukládají obrázky a soubory jednotlivých produktů, které byly skrze administraci přidány do obchodu.

## Příloha B – komponenta Basket (nákupní košík)

Toto je ukázka nevizuální komponenty, která spravuje nákupní košík a veškeré činnosti jako je přidání, odebrání nebo změna množství probíhají právě skrz ni. Tato komponenta využívá návrhový vzor singleton, aby se zamezilo vícenásobnému vytvoření této třídy během obsluhy požadavku.

```
<?php

/**
 * Nákupní košík
 * @author Radek Prokeš
 */
class Basket extends Object
{
   /** @var Basket */
   private static $instance = NULL;

   /** @var Session */
   private $basket;


   /**
    * Konstruktor
    * @param IComponentContainer rodič
    */
   private function __construct()
   {
       $this->basket = Environment::getSession('basket');
       if (empty($this->basket->items)) {
           $this->basket->items = array();
       }
   }


   /**
    * Singleton
    * @param $parent
    * @return unknown_type
    */
   public static function getInstance()
   {
       if (self::$instance === NULL) {
           self::$instance = new self();
       }
       return self::$instance;
   }


   /**
    * Vrací pole všech položek v nákupním košíku
    * @return array
    */
   public function getItems()
   {
       return $this->basket->items;
   }
```

```
    /**
     * Vrací počet položek v nákupním košíku
     * @return int
     */
    public function getItemsCount()
    {
        $count = 0;

        foreach ($this->basket->items as $item) {
            $count += $item->quantity;
        }

        return $count;
    }


    /**
     * Vrací celkovou sumu všech položek v košíku
     * @return int
     */
    public function getItemsPrice()
    {
        $price = 0;

        foreach ($this->basket->items as $item) {
            $price += $item->quantity * $item->price;
        }

        return $price;
    }


    /**
     * Přidá položku do košíku
     * @param int identifikátor položky
     * @param int cena položky
     * @param int množství
     * @return void
     */
    public function addItem($id, $price, $quantity = 1)
    {
        foreach ($this->basket->items as $item) {
            if ($item->id == $id) {
                $item->quantity++;
                return;
            }
        }

        if ($quantity > 0) {
            $this->basket->items[] = (object) array(
                'id' => $id,
                'price' => $price,
                'quantity' => $quantity
            );
        }
    }
```

```
    /**
     * Změní množství určité položky
     * @param int identifikátor položky
     * @param int množství
     * @return void
     */
    public function changeQuantity($id, $quantity)
    {
        foreach ($this->basket->items as $index => $item) {
            if ($item->id == $id) {
                if ($quantity > 0) {
                    $this->basket->items[$index]->quantity = $quantity;
                } else {
                    unset($this->basket->items[$index]);
            }
            break;
            }
        }
    }


    /**
     * Smaže položku z košíku
     * @param int identifikátor položky
     * @return void
     */
    public function deleteItem($id)
    {
        foreach ($this->basket->items as $index => $item) {
            if ($item->id == $id) {
                unset($this->basket->items[$index]);
            }
        }
    }


    /**
     * Vysype nákupní košík
     * @return void
     */
    public function clear()
    {
        unset($this->basket->items);
        $this->basket->items = array();
    }
}
```