UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

WWW aplikace s využitím relační databáze pro elektronický obchod s hasičskou výzbrojí

Jan Čáslava

Bakalářská práce
2010

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2009/2010**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jan ČÁSLAVA**
Osobní číslo:	**I07875**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**www aplikace s využitím relační databáze pro elektronický obchod s hasičskou výzbrojí**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Úkolem práce bude návrh a tvorba internetové aplikace elektronického obchodu určeného k prodeji vybavení pro hasiče.
Teoretická část se bude zabývat návrhem vhodné databáze a možnostmi zabezpečení dat uložených v databázi.
Aplikace bude umožňovat:
- registraci uživatelů,
- vyhledávání zboží,
- objednávání zboží,
- sledování expedice a plateb za objednávky,
- sledování stavu objednávek pro registrované zákazníky,
- zasílání informací o objednávce prostřednictvím e-mailu.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

Castagnetto, J. a kol. Programujeme PHP profesionálně. Computer Press, 2004.
Kout, P. Praktický JavaScript. Zoner Press, 2004.
Oppel, A. Databáze bez předchozích znalostí. Computer Press, 2006.
Lacko, L. . Oracle, správa, programování a použití databázového systému. Computer Press, 2007.

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

Datum zadání bakalářské práce: **15. ledna 2010**

Termín odevzdání bakalářské práce: **14. května 2010**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Cegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2010

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 14. 5. 2010 Jan Čáslava

Poděkování

Tímto bych chtěl poděkovat vedoucí práce RNDr. Ivě Rulicové za odborné vedení a za cenné rady, které mi poskytla v průběhu vypracovávání práce.

Dále bych rád poděkoval své rodině za výraznou podporu během celého studia.

Anotace

Tato práce se zabývá návrhem a implementací internetového obchodu. Předmět prodeje je soustředěn na požární ochranu budov a vybavení pro hasiče, záchranáře i požární sport. Jsou zde využity principy relační databáze, jazyka SQL a PHP. Práce se zaměřuje především na návrh a zabezpečení databázového systému.

Klíčová slova

Internetový obchod, e-shop, Oracle, PHP, relační databáze, zabezpečení databáze, návrh databáze

Title

E-shop – web application (using relation database) with equipment for firemen.

Annotation

This work is about design and implementation of internet shop. The subject of business is fire protection of buildings and equipment for fighters, rescuers and fire sport. There are used principles of relation database, programming language SQL and PHP. It’s focused on database project and security.

Keywords

Internet shop, e-shop, Oracle, PHP, relation database, database design, database security

Obsah

Seznam zkratek

SQL	Structured Query Language
PHP	Hypertext Preprocessor
E-R diagram	Entity - Relationship diagram
UML diagram	Unified Modeling Language
CSS	Cascading Style Sheets
HTML	HyperText Markup Language
WYSIWYG	What You See, Is What You Get SQL
SMTP	Simple Mail Transfer Protocol
OCI8	Oracle Connect Interface
MD5	Message-digest algorithm
SEO	Search Engine Optimization
URL	Uniform Resource Locator

Seznam obrázků

Seznam tabulek

1 Úvodní informace

Svět výpočetní techniky se od své původní podoby značně rozvinul. Pod vlivem změn, mezi které patří zejména rozvoj Internetu, globalizace, fenomény, dochází ke kladení neustále vzrůstajících nároků na shromažďování velkých objemů sdílených dat, které je potřeba vhodným organizovaným způsobem zpracovávat, uchovávat a zabezpečit přístup k nim. Od původních kartoték, souborových struktur a hierarchického modelu [1] jsme se dostali do období, kdy se převážná většina zařízení a aplikací neobejde bez kvalitně zpracované relační databáze.

Tato bakalářská práce se zaměřuje na vývoj a vlastní implementaci internetové aplikace s využitím relační databáze. Konkrétně se jedná o internetový obchod, jehož sortimentem je vybavení pro požární ochranu. V tomto tématu jsem tak spojil své zájmy – webové aplikace, databázové systémy a hasičství.

Aplikace jako taková by měla poskytovat funkci internetového obchodu a částečně i redakčního systému. Cílem je navrhnout skutečně jednoduchý přístup a zajistit tak dostatečný komfort pro jeho uživatele.

Úvodní část práce se bude zabývat logickým návrhem databáze. Budou charakterizovány fáze návrhu relační databáze a normalizace tabulek.

Druhá část je věnována zabezpečení dat v tabulkách. Budou zde popsány možné hrozby a obrana před neoprávněnou manipulací s informacemi.

Ve třetí části bude vysvětlena samotná implementace internetového obchodu.

V závěru dojde ke zhodnocení vytvořené aplikace a porovnání se zadáním.

2 Návrh databáze

Návrh databáze je velice důležitou součástí tvorby jakékoliv aplikace. [14] Nesprávný nebo nepřesný model má za následek nutnost upravit část kódu, nebo dokonce celou aplikaci. Na návrhu databáze se proto nevyplácí šetřit časem. I zdánlivě „dokonalé“ řešení může vést ke špatné implementaci. Programátor navíc na nedostatky většinou přichází až v době vývoje samotné aplikace, takže mu nakonec zabere mnohem více času oprava nepřesností než samotný návrh.

Je proto na místě stanovit si postup, během kterého se od shromažďování informací dostává ke zdokonalujícímu se reálnému návrhu.

2.1 Analýza

Než vývojář začne tvořit, musí si být dopředu jist, jaký je přesný účel aplikace. Během setkání se zadavatelem dochází k výměně upřesňujících informací a požadavků na aplikaci. Vývojář si tak dokáže udělat představu, do jaké hloubky bude potřeba aplikaci rozpracovat. Velice důležitá je i konzultace problému s cílovou skupinou, tedy budoucími uživateli systému. Od nich se může vývojář dozvědět velice zajímavé upřesňující postřehy a návrhy na vylepšení. Mnohdy se lze totiž odlišit od konkurenčního řešení právě maličkostí.

2.2 Konceptuální model

Po analýze požadavků na aplikaci nastává čas na vytvoření konceptuálního modelu. V tomto procesu dochází k přeměně získaných informací na schéma, které reprezentuje jednotlivé prvky zadání.

Konceptuální model znázorňuje entity s jejich vlastnostmi a vztahy mezi dalšími entitami v projektu. Model se už nezabývá daty na fyzické úrovni.

2.2.1 Návrh datových struktur

Stěžejním prvkem databáze je tabulka. Entity vystupující v projektu musí vývojář vhodně přetransformovat do tabulek včetně jejich vlastností.

Základem je vhodné rozvržení struktury tabulek s jejími atributy.

Vzhledem k potřebě jednoznačného identifikátoru dochází v tomto procesu i k návrhu kandidátních klíčů.

Kandidátní klíč jednoznačně identifikuje záznam v tabulce. [13] Nutně platí, že:

- klíč musí být unikátní,
- množina sloupců tvořících tento klíč musí být minimální,

- tabulka musí obsahovat alespoň jeden kandidátní klíč.

Základní jednoduchý návrh by mohl vypadat například následovně:

Tabulka 1, 2, 3 – Konceptuální návrh tabulek

Zákazníci
č. zákazníka
jméno
příjmení
adresa
přihl. jméno
heslo

Objednávky
č. objednávky
zákazník
zboží
cena za kus
množství
dodací adresa
datum objednání
způsob dodání
způsob platby

Zboží
kód zboží
název
popis
kategorie
cena za kus
skladem

2.2.2 Definice vztahů

Tak jako prvky reálného světa mají mezi sebou určité vztahy, i entity jsou navzájem propojeny. [10] Tyto souvislosti je proto potřeba zahrnout do výsledného modelu. Díky postupnému vymezování relací může docházet ke změnám atributů v již navržených tabulkách.

Bližší upřesnění vztahu lze definovat pomocí kardinality. Ta vyjadřuje kolik záznamů z jedné tabulky může vstoupit do vztahu s jedním záznamem tabulky druhé.

Kardinalita může nabývat těchto hodnot:

- 1:1 – jeden záznam jedné tabulky je ve vztahu s jedním záznamem tabulky druhé př.: osoba – rodné číslo (osoba může mít pouze jedno rodné číslo a naopak)
- 1:N – jeden záznam jedné tabulky je vztahu s jedním nebo více záznamy tabulky druhé př.: zaměstnanec – oddělení (zaměstnanec spadá pod jedno oddělení, ale oddělení může mít více zaměstnanců)
- M:N – více záznamů jedné tabulky je ve vztahu s více záznamy druhé tabulky př.: zákazník – zboží (zákazník může nakoupit více druhů zboží a jeden druh zboží může být koupen více zákazníky)

Se vztahy jednotlivých záznamů souvisí i **parcialita**. Ta vypovídá o nutnosti účastnit se vztahu.

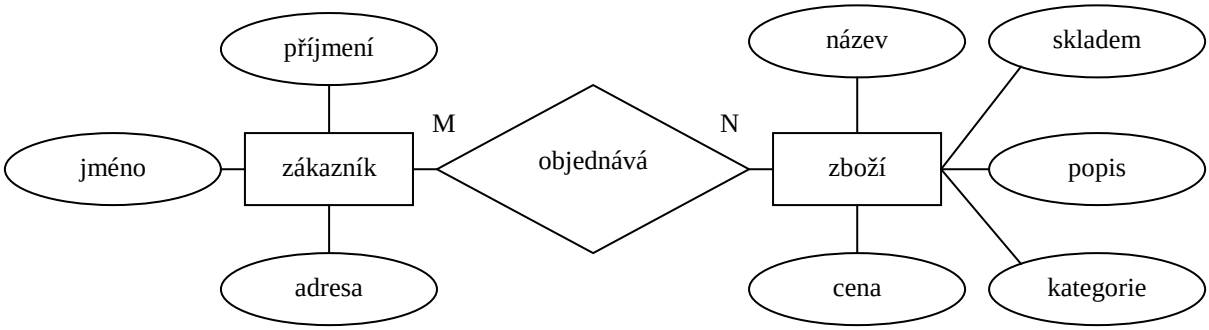
- jednostranně parciální – př.: zboží musí náležet určité kategorii, ale kategorie nemusí obsahovat žádný produkt
- oboustranně parciální – př.: zboží nemusí náležet určité kategorii a kategorie nemusí obsahovat ani jeden produkt

2.2.3 E-R diagram

Každá vývojová část aplikace je reprezentována různými modely (Use Case diagram, UML aktivity diagram, Rich Picture). Pro modelování konceptuálního návrhu databáze se využívá E-R digramu, který je v tomto odvětví standardem.

Návrh tabulek a vztahů mezi nimi můžeme nyní znázornit do tohoto modelu. Jednotlivé prvky schématu mají svůj význam:

- obdélník = entita
- elipsa = atribut
- kosočtverec = vztah



Obrázek 1 - E-R diagram

2.3 Logický návrh

E-R diagram nyní poslouží pro vytvoření relačního modelu, který umožní odstranění nadbytku dat a stanovení klíčových hodnot.

Proces, který definuje postup pro dosažení těchto vlastností, se nazývá normalizace.

2.3.1 Normalizace

Jak už bylo zmíněno výše, vhodná implementace tabulek je nejpracnější částí návrhu databáze. [3][7] Nepřesné rozvržení zapříčiňuje zbytečnou složitost a ztížené podmínky pro vývoj a další rozšíření.

Cílem normalizace je navrhnout takové tabulky, které budou logicky a jednoduše postavené, snadno zpracovatelné, eliminují redundanci dat a získávání informací z nich bude dostatečně efektivní.

Postupy normalizace stanovují pravidla, která se nazývají normální formy. Postupným porovnáváním tabulek s různými úrovněmi normálních forem získáváme dokonalejší návrh databázové struktury.

Tabulka 4 – Tabulka *Zákazníci* bez normalizace

č. obj.	datum	jméno	příjmení	adresa
111	29.3.2010	Pavel	Novák	Poděbradská 126, 12345 Praha
112	30.3.2010	Jaromír	Havel	Hradecká 76, 12345 Praha
113	31.3.2010	Lukáš	Plaček	Korunovační 12, 12346 Liberec

2.3.2 Bezztrátová dekompozice

Relační databáze jsou založeny na spojování atributů tabulek. Výhodou tohoto řešení je, že výsledkem jsou původní data a to bez ztráty jakýchkoli informací. Bezztrátová dekompozice znamená rozmělnění tabulek na přehlednější tabulky se vzájemně souvisejícími atributy. Příkladem je dekompozice Tabulky 4.

Tabulka 5 – Tabulka *Objednávk*y po dekompozici

č. obj	datum	č. zákazníka
111	29.3.2010	1
112	30.3.2010	2
113	31.3.2010	3

Tabulka 6 – Tabulka *Zákazník*i po dekompozici

č. zákazníka	jméno	příjmení	adresa
1	Pavel	Novák	Poděbradská 126, 12345 Praha
2	Jaromír	Havel	Hradecká 76, 12345 Praha
3	Lukáš	Plaček	Korunovační 12, 12346 Liberec

2.3.3 Klíče

K tomu, abychom mohli dekomponované tabulky spojovat, musíme definovat jedinečné atributy – klíče. [14] Kdybychom tento krok neprovedli, dostávali bychom na výstupu nesmyslné výsledky (více zákazníků by mohlo mít stejné zákaznické číslo, a tak by docházelo k „pomíchání“ objednávek).

Primární klíč je jeden z kandidátních klíčů (v našem případě č. obj.), který byl navržen během konceptuálního návrhu. Všeobecně se doporučuje volit kratší klíče z důvodu úspory místa, zejména v případech, kdy se primární klíč objevuje jako cizí v dceřiné tabulce.

Vzhledem k tomu, že ne vždy obsahuje tabulka jednoznačný atribut, je zapotřebí zvolit tzv. klíč umělý. Z pravidla to bývá ID (v našem případě č. zákazníka).

Cizí klíč je primární klíč cizí tabulky, který je atributem dané tabulky. Platí, že klíč nabývá hodnot primárního klíče cizí tabulky nebo hodnoty NULL.

Cizím klíčem je v našem případě č. zákazníka v tabulce objednávek.

Při spojení tabulek objednávek a zákazníků podle č. zákazníka dostaneme na výstupu správná požadovaná data. Byla tak splněna počáteční podmínka.

2.3.4 Referenční integrita

Při manipulaci s cizími klíči je potřeba dodržovat pravidla referenční integrity. [14] Musí platit pravidlo, že žádný řádek v podřízené tabulce nesmí obsahovat hodnotu cizího klíče, která nenabývá hodnot primárního klíče v nadřazené tabulce.

Vkládání – do hodnoty cizího klíče nemůžeme vkládat jiné hodnoty než hodnoty primárního klíče nadřazené tabulky.

Odstraňování – při odstraňování záznamu může dojít k odstranění dat, na něž cizí klíč odkazuje. V praxi na to databázový systém upozorní a nedovolí data smazat. Vývojář tak musí nejprve odstranit data, která jsou podmínkou závislosti a poté samotný záznam.

Úprava – při aktualizování údajů může dojít k nekonzistenci dat, kdy se na související data zapomene.

2.3.5 Nultá normální forma

Po dekompozici a stanovení primárních klíčů můžeme přistoupit k samotné normalizaci relací. Prvním pravidlem je nultá normální forma.

„Tabulka je v nulté normální formě právě tehdy, existuje-li alespoň jedno pole, které obsahuje více než jednu hodnotu.“ [3]

Důsledkem tohoto postupu může být neefektivní a složité vyhledávání dle hodnoty v tomto poli.

Příkladem mohou být struktury Tabulky 4.

2.3.6 První normální forma

Pro odstranění nedostatků nulté normální formy přistoupíme k vyššímu stupni normalizace.

„Relace je v první normální formě, pokud každý její atribut obsahuje jen atomické (dále nedělitelné) hodnoty.“ [3]

Tabulka 7 – Tabulka *Zákazníci* v 1. normální formě

č. zákazníka	jméno	příjmení	ulice	č.p.	město	PSČ
1	Pavel	Novák	Poděbradská	126	Praha	12345
2	Jaromír	Havel	Hradecká	76	Praha	12345
3	Pavel	Novák	Korunovační	12	Liberec	12346

Toto pravidlo může být někdy velice objektivní a záleží tak na konkrétní implementaci aplikace a potřebám vývojáře. Příkladem může být atribut s datem. Tuto hodnotu by bylo možné dále rozložit na položku dne, měsíce a roku. V případě, že se bude v systému pracovat s datem jako s jedinou hodnotou, je vhodné hodnotu realizovat jako dále nedělitelnou.

Druhou možností je časté využívání jednotlivých atributů data. To vede vývojáře k rozložení hodnoty na jednotlivé prvky.

2.3.7 Druhá normální forma

„Tabulka je ve druhé normální formě, jestliže je v první a navíc platí, že existuje klíč a všechna neklíčová pole jsou funkcí celého klíče, a tedy ne jen jeho částí.“ [3]

Z tohoto lze jednoduše odvodit, že pokud je tabulka navržena pouze s jednosložkovým primárním klíčem, podmínka druhé normální formy je již splněna.

Problém nastává u složeného primárního klíče a závislostí atributů na něm. Je proto zapotřebí uvědomit si vztahy mezi atributy a tabulku vhodně rozložit.

Tabulka 8 – Tabulka *Zboží* před normalizací

název zboží	kategorie	cena	skladem
Fireman III	Obleky	13600	10
Tajfun Profi	Armatury	7600	2
Zahas V	Obleky	12300	5

Primárním klíčem v tomto případě mohou být hodnoty *název zboží* a *kategorie*. Opět lze odvodit, že pole *cena* a *skladem* jsou závislé pouze na klíči *název zboží*, nikoli už na *kategorii*. Proběhla proto transformace tabulky do těchto relací:

Tabulka 9 – Tabulka *Zboží* ve 2. normální formě

č. zboží	název zboží	cena	skladem
1	Fireman III	13600	10
2	Tajfun Profi	7600	2
3	Zahas V	12300	5

Tabulka 10 – Tabulka *Kategorie* ve 2. normální formě

č. kategorie	název kategorie
1	Obleky
2	Armatury

2.3.8 Třetí normální forma

„Tabulka je ve třetí normální formě, jestliže každý neklíčový atribut není tranzitivně závislý na žádném klíči schématu neboli jestliže je ve druhé normální formě a zároveň neexistuje jediná závislost neklíčových sloupců tabulky.“ [3]

To znamená, že každý neklíčový atribut je závislý pouze na primárním klíči.

Pro vysvětlení lze využít tabulku č. 6 bez atributů ulice a č.p., které nejsou v tomto případě tolik podstatné. V původním schématu platí, že existuje závislost mezi *PSČ* a *město*. Toto ovšem neodpovídá pravidlu třetí normální formy. Musíme proto vytvořit novou relaci, kde budou prvky závislé výhradně na hodnotě primárního klíče.

Tabulka 11 – Tabulka *Zákazníci* ve 3. normální formě

č. zákazníka	jméno	příjmení	č. města
1	Pavel	Novák	1
2	Jaromír	Havel	1
3	Pavel	Novák	2

Tabulka 12 – Tabulka *Města* ve 3. normální formě

č. města	město	PSČ
1	Praha	12345
2	Liberec	12346
3	Ostrava	12347

Třetí normální forma odstraňuje redundanci dat. Dochází tak k efektivnějšímu využívání databázového serveru. Výhodou 3. NF je i skutečnost, že můžeme vkládat hodnoty, které nebyly prozatím použity (např. tabulka s městy, položka *Ostrava*)

2.3.9 Další typy normálních forem

V pravidlech pro návrh databázové struktury jsou k dispozici ještě další typy normálních forem. Jsou to Boyce-Coddova, čtvrtá a pátá normální forma, které se zabývají převážně problematikou složených klíčů.

V praxi je standardem normalizace tabulek do třetí normální formy, která postačuje pro převážnou většinu aplikací. Jedním z důvodů je i to, že čím vyšší je stupeň normalizace, tím větší nároky jsou kladeny na výkon databázového serveru.

2.3.10 Denormalizace

Ne vždy je vhodné data normalizovat do co nejvyššího stupně. [14] Výsledkem je velké množství tabulek a vztahů mezi nimi. Jak již bylo řečeno výše, může to mít výrazný vliv na výkon databázového serveru.

Pro dosažení rychlejšího zpracovávání dotazů je vhodné využít denormalizaci, tedy opak normalizace. Cílem je na vhodných místech, kde nebude tolik podstatné zavedení redundance dat, tabulky sloučit. Tento proces ve výsledku sníží počet entit a vztahů, což vede k jednodušší struktuře databáze.

3 Zabezpečení obsahu databáze

Databáze shromažďuje velké množství informací. [9] Některé z těchto dat mohou být velice citlivé údaje. Vývojář se proto musí v další části návrhu aplikace zabývat i zabezpečením přístupu k těmto datům.

Vhodná implementace jednoduchých, ale efektivních opatření ve výsledku tvoří odolnou zabezpečenou aplikaci.

3.1 Autentizace – přihlášení

Jedním ze základních a samozřejmých způsobu zabezpečení je autentizace. Tento pojem znamená proces přihlašování uživatelů. V podstatě se jedná o ověření identity uživatele na základě jeho údajů. Pokud tedy bude chtít využívat data poskytovaná databází, musí mít registrováno uživatelské jméno a heslo.

Přičemž platí, že čím složitější heslo se volí, tím hůře ho lze prolomit, a dostat se tak k citlivým datům.

Přidávat uživatele má povoleno pouze správce databáze.

```
CREATE USER frantisek IDENTIFIED BY 1234567890heslo0987654321;
```

3.2 Autorizace

Po úspěšné autentizaci dochází k tzv. autorizaci.

Jedná se o mechanismus přidělování rolí, podle nichž se kontroluje, zda má uživatel oprávnění přistupovat k danému soboru, objektu nebo vykonat určitou akci.

Lze tak jednoduše rozlišovat osoby pracující se systémem a zabránit jim např. mazat nebo modifikovat data.

3.3 Systémová práva

Systémová práva definují, jaké operace může uživatel vykonávat ve svém nebo cizím schématu. [12] Jedná se tak například o spouštění dávkových úloh, změny systémových parametrů, vytváření rolí nebo připojení k databázi.

Tato práva mohou být přidělována uživateli, roli nebo skupině PUBLIC (představuje všechny uživatele databáze) správcem databáze.

```
GRANT CREATE SESSION, CREATE TABLE TO frantisek;
REVOKE CREATE TABLE FROM frantisek;
```

3.4 Přístupová práva

Přístupová práva nad objekty definují dotazy, které může uživatel využívat při práci s objekty ostatních databázových schémat. [12]

Každý uživatel má práva nad tabulkami svého schématu. Různá oprávnění může potom přidělovat ostatním uživatelům, rolím nebo skupině PUBLIC. Pokud nepoužije klauzuli WITH GRANT OPTION, nemohou tito „obdarovaní“ poskytovat práva dále.

```
GRANT SELECT, INSERT, UPDATE ON tabulka1 TO frantisek;
GRANT EXECUTE, DEBUG ON funkce1 TO frantisek;
```

3.5 Role

Role obsahuje soubor práv, která mohou být přiřazována více uživatelům najednou. Tím se usnadňuje správa. Změnou oprávnění role se tak změní oprávnění všem podřízeným uživatelům.

Role mohou být přidělovány uživateli správcem databáze nebo uživatelem s oprávněním CREATE ROLE.

Role lze zabezpečit heslem identified by <heslo> a not identified.

```
CREATE ROLE zakaznici not identified;
GRANT SELECT, INSERT, UPDATE ON tabulka1 TO zakaznici;
GRANT zakaznici TO frantisek, petr;
```

Veškerá oprávnění databázových uživatelů jsou přístupná v systémovém katalogu.

3.6 SQL Injection

Významnou hrozbou pro data v databázi je tzv. SQL Injection. [5][6] Tento způsob útoku je založen na zranitelnosti databáze z pohledu možného přidávání příkazů k databázovému dotazu, který je odesílán zprostředkovatelskou aplikací.

3.6.1 Příklad zneužití

Mějme dotaz na přihlášení v následujícím tvaru:

```
SELECT * FROM uzivatele WHERE prihlasovaci_jm= ‘ “ & login & ” ’ AND
heslo= ‘ “ & pass & ” ’;
```

Pokud uživatel zadá prostřednictvím aplikace hodnoty frantisek a 1234, prohledá se příslušná tabulka a v případě nalezení této kombinace dat vrátí dotaz jeden záznam.

Pokud ale uživatel vloží např. hodnotu uživatelského jména ' or 'b'='b --, pozmění se význam celého dotazu.

```
SELECT * FROM uzivatele WHERE prihlasovaci_jm= ‘’ OR ‘b’=‘b’ -- AND
heslo= ‘ “ & pass & ” ’;
```

Výsledkem je zrušení vyhledávání hesla, jelikož znak -- znamená v SQL jazyce komentář. Testuje se tak pouze podmínka prihlasovaci_jm= " OR 'b'= 'b', přičemž 'b'= 'b' je pravdivé vždy, tudíž i celá podmínka vyhovuje. Útočník pravděpodobně tímto jednoduchým způsobem získá přístup do aplikace.

3.6.2 Hrozby

Vzhledem k možnosti takto upravovat dotazy kladené na databázi může útočník:

- získat citlivá data,
- přihlásit se jako administrátor aplikace,
- měnit, mazat data nebo databázové objekty.

Což je velice nežádoucí a nebezpečné.

3.6.3 Prevence

Velice důležité je tak vhodnou formou ošetřit celou aplikaci a tomuto typu útoku zabránit. [6][12] Mezi nejčastější postupy patří:

- Přidělit anonymním i autorizovaným uživatelům ta oprávnění, která jsou pro jejich roli nezbytnější.
- Ošetření uvozovek ve všech vstupech do aplikace. Vstupy se rozumí proměnné GET, POST, COOKIES, URI parametry.
- Testovat vstupy na správnost datových typů (zda číslo je skutečně číslem) a délku řetězců, u kterých je dán přený počet znaků. K tomuto účelu lze využít regulárních výrazů.
- Využívat procedury a funkce, které budou dotazy vykonávat a uživatel bude mít právo je spouštět.
- Lze využívat i funkcí oci_bind_by_name (vkládá PHP proměnné do dotazu, který se odesílá Oracle databázi) nebo mysql_real_escape_string (opatří speciální znaky zpětným lomítkem).

3.7 Pohledy

Pohledy jsou nástrojem pro omezení přístupu k informacím. Obecně je pohled pouze předpis, pomocí kterého se data z tabulek získávají. Můžeme tak nadefinovat

sloupce tabulky, které jsou podstatné pro danou akci. Ostatní sloupce uživatel nemá možnost vidět.

Další vlastností pohledů je možnost nastavit čtení, zápis nebo obojí. Jedná se tak o jisté omezení podobně jako u objektových oprávnění.

```
CREATE VIEW login AS
SELECT prihl_jmeno, heslo
FROM přihlazeni
WITH READ ONLY;
```

3.8 Procedury a funkce

Omezení přístupu k datům v databázi může být řešeno i pomocí procedur a funkcí, které vytvoří rozhraní mezi uživatelem a databází Jednoduše se zeptáme funkce na otázku a funkce nám odpoví. Uživatel tedy zadává pouze parametry a ke struktuře tabulky se nemůže dostat.

3.9 Chybová upozornění

Dalším drobným přínosem k zabezpečení může být zakázat výpis chybových hlášení. Případný útočník tímto způsobem nebude mít možnost zjistit podrobnosti o chybě a v některých případech i částečné struktury kódu aplikace.

Lze toho dosáhnout např. pomocí souboru .htaccess, který je umístěn do kořenového adresáře a definuje dodatečné direktivy pro webový server Apache.

```
#obsah souboru .htaccess
php_flag display_errors off
php_flag display_startup_errors off
```

4 Projekt – Internetový obchod

Tato část práce se bude zabývat demonstrací teoretických poznatků z předchozích částí. Pro konkrétní realizaci jsem se rozhodl věnovat návrhu a samotné implementaci internetového obchodu. Předmětem prodeje je vybavení pro požární ochranu.

4.1 Základní analýza

Cílem projektu je tvorba internetové objednávkové aplikace v kombinaci s jednoduchým redakčním systémem.

Přihlášený zákazník má možnost vyhledávat a vkládat zboží do nákupního košíku, a tak vytvářet objednávky, o kterých je informován prostřednictvím e-mailu.

Administrátor (zaměstnanec) poté tyto objednávky sleduje a zpracovává. V případě potřeby vkládá nebo edituje obsah seznamu produktů, doplňujících textů nebo kategorií.

V systému je zaveden i hlavní administrátor, který má možnost editovat zákaznické i administrátorské účty.

4.2 Použité technologie

4.2.1 Operační systém

Základem serveru se stal operační systém Ubuntu 9.10 na bázi Linuxu. Hlavním důvodem byla jednoduchá správa balíků a svobodné licencování jak tohoto systému, tak i veškerých komponent pro zprovoznění projektu.

4.2.2 Apache 2

Jako softwarový webový server byl použit Apache 2, v současné době světově nejvíce používaný. Jedná se o produkt šířený jako open-source[1].

4.2.3 PHP5

Skriptovací jazyk byl volen vzhledem k jednoduchosti, předchozím znalostem a možnosti komunikace s databázovým serverem Oracle. Výběr tak padl na PHP, které umožňuje programování dynamických stránek. Během tvorby aplikace se ukázalo, že je pro podmínky vypracování projektu zcela dostačující.

PHP v základní instalaci nepodporuje napojení na Oracle databázi. K tomuto účelu slouží knihovna oci8, kterou bylo nutno doinstalovat. [7]

[1] Software s otevřeným zdrojovým kódem

4.2.4 PHP PEAR

PHP PEAR je užitečný framework, který nabízí velké množství knihoven. Pro aplikaci internetového obchodu jsem využil třídu *Mail* pro odesílání e-mailů. Díky jednoduchému nastavení, které využívá nadřazeného SMTP serveru, odpadá nutnost instalace vlastního poštovního serveru.

Implementace této komponenty do systému je uvedena v příloze C.

4.2.5 CSS

K návrhu jednoduchého vzhledu bylo použito kaskádových stylů. Jedná se o jazyk, který definuje způsob zobrazování HTML stránek. Pro vývojáře to znamená zjednodušení práce, jelikož jednu vytvořenou třídu lze přiřadit více entitám HTML dokumentu.

4.2.6 Oracle Database 10g

Jako úložiště dat byl zvolen relační databázový systém Oracle 10g Express Edition. Jedná se o celosvětovou jedničku ve svém oboru. Výhodami jsou především vysoký výkon a rozšířené možnosti pro zpracování dat.

Vzhledem k tomu, že veřejné hostingové služby nenabízejí podporu Oracle databáze bez nutnosti zaplacení poplatků, nainstaloval jsem Oracle 10g XE, který má jistá především hardwarová omezení, na vlastní „server“. Ten společně s knihovnou OCI8 pro PHP5 dostačuje k vývoji a realizaci této bakalářské práce.

4.2.7 WYSIWYG editor

Aby bylo možné z pohledu administrátora jednoduše editovat obsah webové prezentace, byl využit WYSIWIG editor (v nezkráceném tvaru „What you see is what you get“ čili „co vidíš, to dostaneš“). Jedná se o jistý způsob textového editoru, díky němuž může uživatel formátovat text dle svých potřeb. Výstupem je text doplněný o HTML značky, který tvoří požadovaný, na vstupu nadefinovaný, vzhled.

V aplikaci internetového obchodu byl využit TinyMCE javascriptový editor.

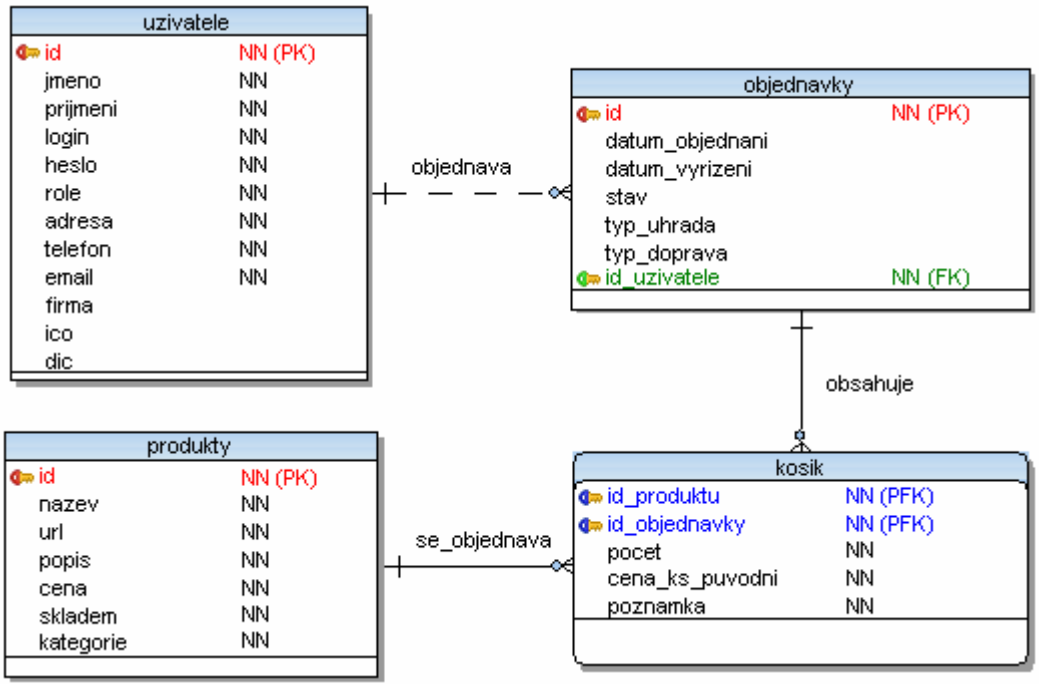
4.3 Návrh databáze

Vzhledem k tomu, že byl již vysloven seznam cílů, můžeme přistoupit k návrhu databázové části aplikace. Celý projekt je založen na způsobu propojení uživatelů, objednávek a produktů. Musíme proto aplikovat vhodnou dekompozici tabulek, aby nebyla zbytečně zaváděna redundance dat a získávání informací bylo dostatečně efektivní.

4.3.1 Konceptuální model

Na počátku musíme definovat entity reálného světa včetně jejich atributů a vztahů. V této části návrhu není podstatné řešení na fyzické úrovni (např. definice datových typů).

Nejjednodušším způsobem je využít E-R digramu, který nám napomůže získat přehled o již zmíněných vlastnostech.



Obrázek 2 - Konceptuální model

V diagramu jsou realizována základní spojení tabulek včetně znázornění důležitých atributů. Popis a funkce jednotlivých tabulek budou popsány po procesu normalizace.

4.3.2 Logický návrh

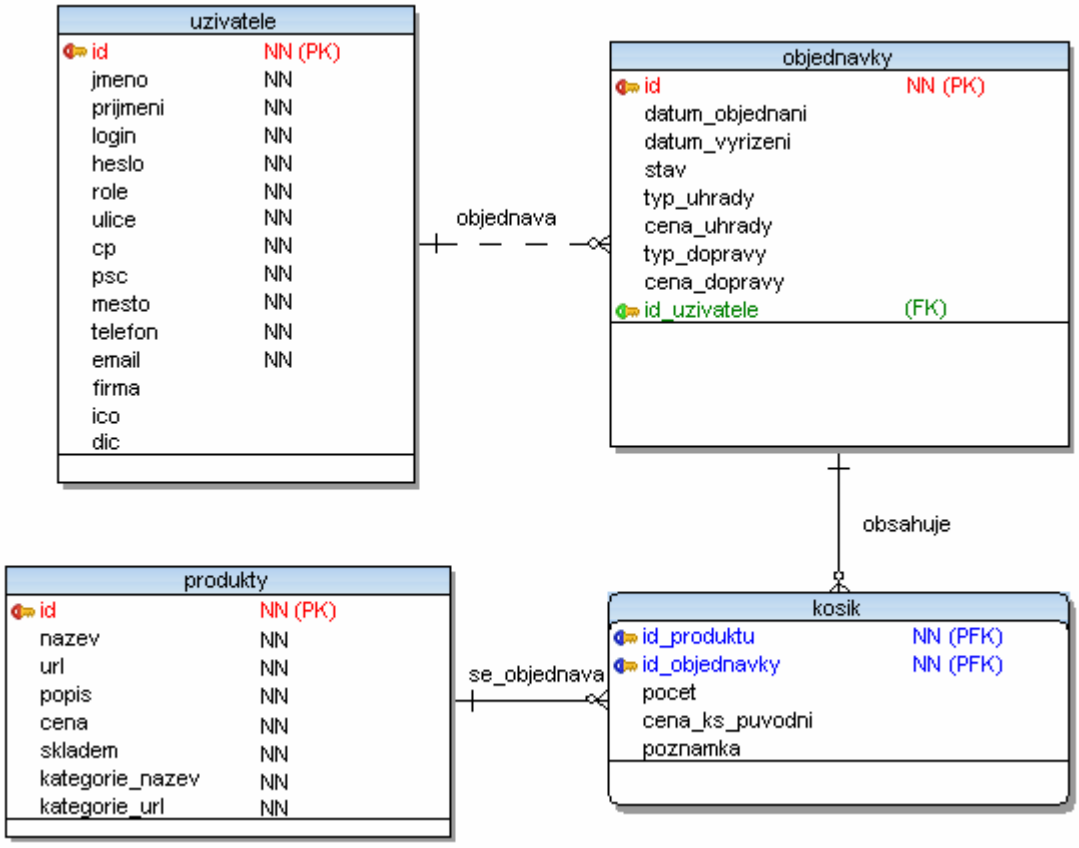
Hlavním úkolem logického návrhu je dekompozice tabulek do takové podoby, která odstraní nadbytečné hodnoty v důsledku rozmělnění tabulek na menší lépe zpracovatelné objekty.

Nultá normální forma

Příkladem podmínky nulté normální formy může být tabulka *uživatelé*. Platí, že atribut *adresa* obsahuje více než jednu hodnotu (ulice, číslo popisné, město a PSČ). Tento způsob nám ale neumožňuje vyhledat osoby např. dle konkrétního města, aniž bychom použili regulární výrazy.

První normální forma

První normální forma nám zajistí, že každý atribut obsahuje dále nedělitelné hodnoty.



Obrázek 3 - První normální forma

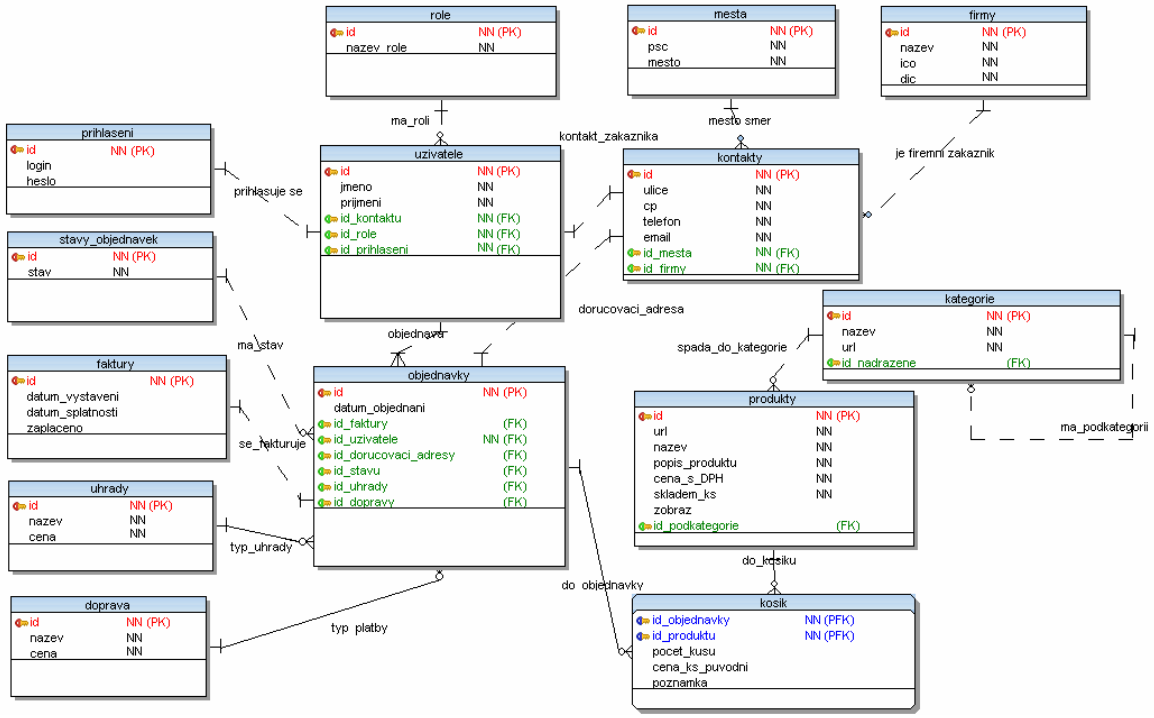
Druhá normální forma

Pravidlo druhé normální formy splňují všechny tabulky diagramu (platí 1. normální forma a neklíčové atributy jsou funkcí celého klíče).

Třetí normální forma

Pomocí třetí normální formy zajistíme, že neklíčové atributy v tabulce jsou vzájemně nezávislé.

Nastává tak rozpad tabulek na menší objekty, které obsahují navzájem související data a jejichž atributy jsou závislé výhradně na primárním klíči.



Obrázek 4 - Třetí normální forma

Denormalizace

Proces denormalizace jsem se rozhodl aplikovat na tabulku *kontakty*. Sjednocením s tabulkou *firmy* a *města* vznikne rozsáhlejší objekt, se kterým se bude v aplikaci lépe pracovat. Atributy této upravené tabulky odpovídají druhé normální formě.

Poslední úpravy návrhu

Na základě zamyšlení se nad problematikou možných změn sazeb za způsob dopravy a úhrady během právě zpracovávaných objednávek došlo k doplnění tabulky *objednávky* o atributy *cena_doprava* a *cena_uhrada*. Zákazník tak hradí cenu, která platila v době realizace objednávky.

Další úpravou je vytvoření tabulky *zpusoby_dodani*, která slouží jako shromaždiště kombinací způsobů dopravy a platby.

4.3.3 Fyzický datový model

Fyzický datový model byl vytvořen v programu Toad Data Modeler 3. Ten umožňuje vygenerovat SQL kód vytvořeného schématu, což zjednodušuje zavedení databáze na server. Fyzický návrh je součástí přílohy A.

4.4 Popis databázové struktury

4.4.1 Tabulka Texty

Tato tabulka slouží pro uchovávání doprovodných textových informací obchodu. Primárním klíčem je *id*, které je při vkládání vygenerováno ze sekvence *seq_id_textu* pomocí triggeru *trig_id_textu*.

Dle nastavení atributu *zobraz* je volena přístupnost daného textu.

```
CREATE SEQUENCE "OBCHOD"."SEQ_ID_TEXTU"
MINVALUE 1
MAXVALUE 9999999999999999999999999999
INCREMENT BY 1
START WITH 1
CACHE 20
NOORDER
NOCYCLE;
```

```
CREATE OR REPLACE TRIGGER "OBCHOD"."TRIG_ID_TEXTU"
BEFORE INSERT ON texty
FOR EACH ROW
BEGIN
  SELECT seq_id_textu.NEXTVAL INTO :new.id FROM DUAL;
END;
```

4.4.2 Tabulka Uživatelé

Tabulka slouží pro shromažďování informací o uživateli. Primárním klíčem je *id*, které je při vkládání vygenerováno ze sekvence *seq_id_uzivatele* pomocí triggeru *trig_id_uzivatele*. Cizí klíče odkazují na přihlašovací údaje, přidělenou roli a kontaktní informace.

4.4.3 Tabulka Přihlášení

Tabulka slouží pro uchovávání přihlašovacích údajů uživatelů systému. Primárním klíčem je *id*, které je při vkládání vygenerováno ze sekvence *seq_id_prihlaseni* pomocí triggeru *trig_id_prihlaseni*. Atribut *heslo* je zakódován pomocí metody MD5.

4.4.4 Tabulka Role

Tabulka obsahuje informace o uživatelských rolích. Nabývá konstantních hodnot *zákazník*, *administrátor*, *masteradmin*, které se nedají editovat.

4.4.5 Tabulka Kontakty

Tabulka slouží jako úložiště veškerých kontaktních i fakturačních údajů v aplikaci. Primárním klíčem je *id*, které je při vkládání vygenerováno ze sekvence *seq_id_kontaktu* pomocí triggeru *trig_id_kontaktu*.

Jedná se o jedinou tabulku databázového schématu, která je normalizována pouze do 2. normální formy.

4.4.6 Tabulka Objednávky

Tabulka slouží pro uchovávání informací ohledně objednávek. Primárním klíčem je *id*, které je při vkládání vygenerováno ze sekvence *seq_id_objednavky* pomocí triggeru *trig_id_objednavky*. Cizí klíče odkazují na objednávajícího zákazníka, informace o faktuře, doručovací údaje, stav vyřízení objednávky, způsob dopravy a úhrady.

Atributy cena_doprava a cena_uhrada, jsou zde uvedeny z důvodu možné změny sazebníku během již provedené, ale prozatím nevyřízené objednávky. Je tak zajištěno, že zákazník bude hradit původní částku.

4.4.7 Tabulka Stavy objednávek

Tabulka obsahuje informace o možných stavech objednávek. Nabývá konstantních hodnot *nevyřízeno*, *zpracovává se*, *vyřízeno*, které se nedají editovat.

4.4.8 Tabulka Faktury

Tabulka slouží pro uchovávání informací o fakturaci. Primárním klíčem je *id*, které nabývá totožných hodnot s *id* odpovídající faktury. Je tak zajištěna jedinečnost.

4.4.9 Tabulka Uhrady

Tabulka obsahuje informace o možných typech úhrady. Nabývá konstantních hodnot *dobírkou*, *převodem*, *osobně*, které se nedají editovat.

4.4.10 Tabulka Doprava

Tabulka obsahuje informace o možných typech dopravy. Nabývá konstantních hodnot *Česká pošta, zásilková služba, osobní odběr*, které se nedají editovat.

4.4.11 Tabulka Způsoby dodání

Tato tabulka reprezentuje vztah M:N mezi objednávkami a způsoby dopravy a úhrady. Je tak zajištěno že zákazník nemá možnost např. vybrat kombinaci *osobní odběr* a *platba dobírkou*.

4.4.12 Tabulka Produkty

Tabulka shromažďuje údaje o nabízených produktech. Primárním klíčem je *id*, které je při vkládání vygenerováno ze sekvence *seq_id_produktu* pomocí triggeru *trig_id_produktu*. Cizí klíč odkazuje na typ kategorie, které je v hierarchickém uspořádání na nejnižší úrovni.

Dle nastavení atributu *zobraz* je volena přístupnost a možnost objednání daného produktu.

4.4.13 Tabulka Kategorie

Tabulka je řešena jako ukazující sama na sebe. Lze tak vytvářet strukturu kategorií a podkategorií. Primárním klíčem je *id*, které je při vkládání vygenerováno ze sekvence *seq_id_kategorie* pomocí triggeru *trig_id_kategorie*.

4.4.14 Index idx_nazev_produktu

V aplikaci byl využit index nad tabulkou *produkty*, konkrétně sloupcem *název*. Důvodem je rychlejší zpracování dotazů při vyhledávání zboží prostřednictvím vyhledávacího formuláře aplikace.

```
CREATE INDEX idx_nazev_produktu ON produkty(nazev);
```

4.4.15 Pohled Login

Pohled *Login* slouží při procesu přihlášení pro zjištění kombinací přihlašovacího jména a hesla. Objekt je vytvořen tak, aby z něho bylo možné pouze číst.

```
CREATE OR REPLACE VIEW "OBCHOD"."LOGIN" ("ID", "LOGIN", "HESLO") AS
SELECT ID, LOGIN, HESLO
FROM PRIHLASENI
WITH READ ONLY;
```

4.5 Náhled na aplikaci

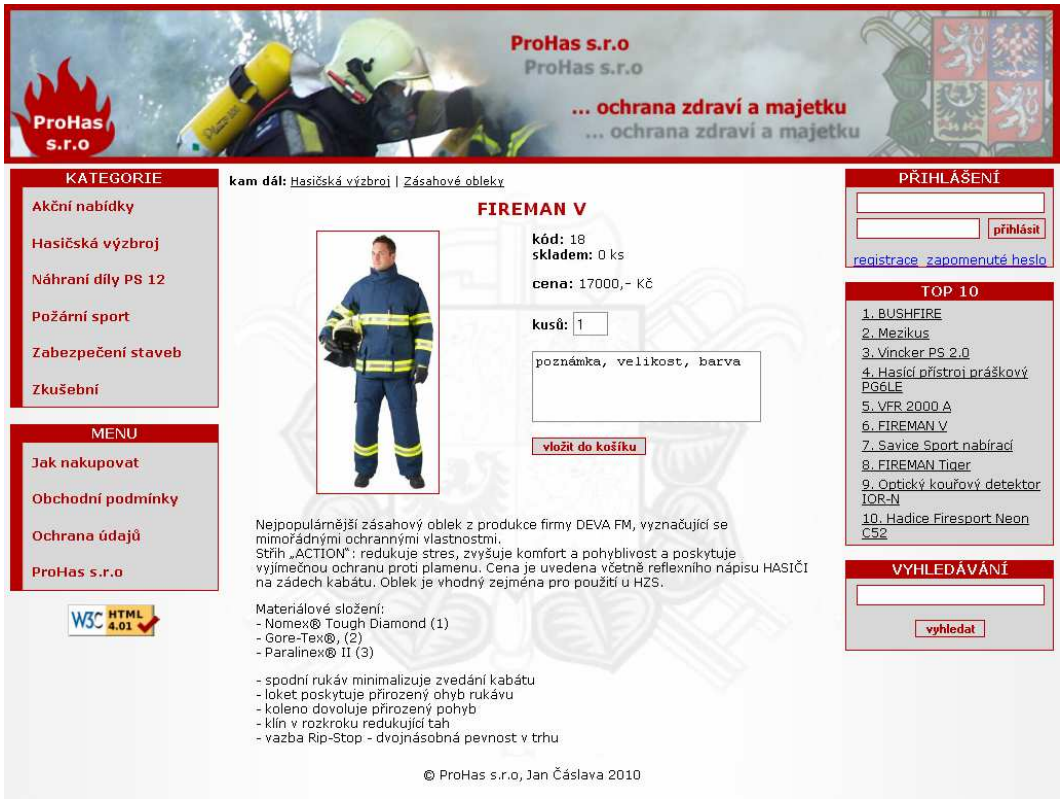
4.5.1 Layout

Layout je navržen jako přehledný jednoduchý třísloupcový (pravé menu, levé menu a obsah stránky). Součástí je i logo a na konci stránky „patička“. Pro realizaci vzhledu byly využity principy CSS (kaskádových stylů), které jsou definovány tak, aby se stránka zobrazovala shodně jak v prohlížeči Internet Explorer 6,7,8, tak i Mozille Firefox 3.6.3 a Opeře 10. Layout je HTML validní.

Levé dvouúrovňové menu je založeno na detekci proměnné reprezentující zkratku kategorie z URL adresy, podle níž se poté zobrazí i položky požadovaného podmenu.

Pravé menu slouží jako vstup do aplikace. Jedná se o prostor vymezený pro přihlašovací údaje, který se po úspěšné autentizaci a autorizaci změní na menu dle přidělené role. Dalším vstupním údajem může být název nebo část názvu zboží, které následně slouží pro vyhledávání produktů.

Část vymezená pro obsah je určena pro zobrazovaní veškerých produktů odpovídajících vybrané kategorii/podkategorii. O tom, v jaké kategorii se produkt nachází, informuje jednoduché menu nad popisem jeho detailu. To slouží zároveň pro lepší orientaci uživatele v aplikaci.



Obrázek 5 - Layout

4.5.2 Adresářová struktura

Adresářová struktura aplikace je rozvržena tak, aby bylo možné kdykoli dohledat zdrojové kódy. Názvy souborů vždy vystihují akci, kterou reprezentují.

Kořenový adresář obsahuje především indexovou stránku, soubor kaskádových stylů, skripry pro přihlášení a další specifické soubory:

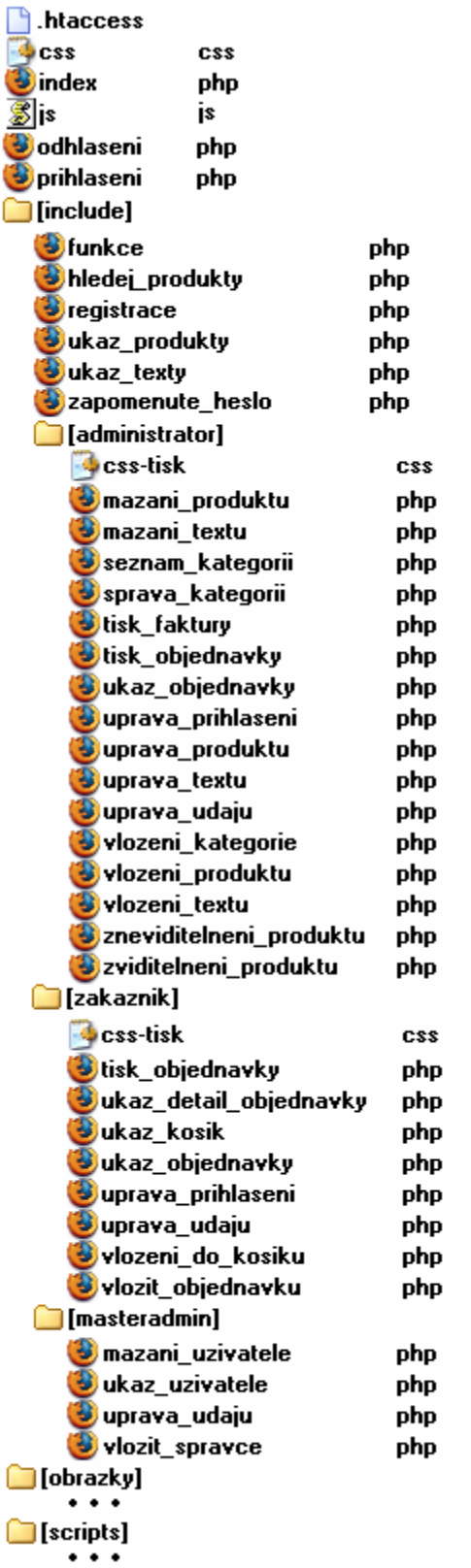
- *.httacess* – podstrkávání URL, vypnutí výpisu chybových hlášení
- *js.js* – definice WYSIWYG editoru

Adresář include obsahuje další podadresáře (administrátor, zákazník, masteradmin), které odpovídají jednotlivým uživatelským rolím a jejich možnostem, jak systém využívat.

Dále obsahuje i soubory skriptů, jež jsou důležitou součástí celé aplikace bez rozdílu přidělené role – vyhledávání zboží, zobrazování produktů a doprovodných textů, registrace, zaslání zapomenutého hesla a skriptovací soubor *funkce.php*, který souží pro definici funkcí určených pro spojení s databází, odesílání e-mailů a haléřového zaokrouhlování.

Adresář obrazky obsahuje veškerou grafiku, která je s touto aplikací spojena (prvky layoutu i obrázků nabízených produktů).

Adresář scripts obsahuje soubory WYSIWYG editoru TinyMCE.



Obrázek 6 - Adresářová struktura

4.6 Uživatelé v systému

Aplikace rozeznává čtyři typy uživatelů. Podle přidělené role mohou využívat systém různými způsoby.

4.6.1 Anonymní návštěvník

Anonymními návštěvníky jsou všichni nepřihlášení uživatelé systému. Kvůli tomu mají pouze omezená práva systém využívat. Smějí stránky prohlížet, vyhledávat produkty, zaregistrovat se nebo si nechat zaslat zapomenuté heslo k již existujícímu uživatelskému účtu.

4.6.2 Zákazník

Tato role již nabízí rozšířenou možnost využívat systém. Hlavní náplní přihlášeného zákazníka je vyhledávání zboží s možností vložení do nákupního košíku, tvorba, sledování a editace objednávek nebo změna osobních údajů.

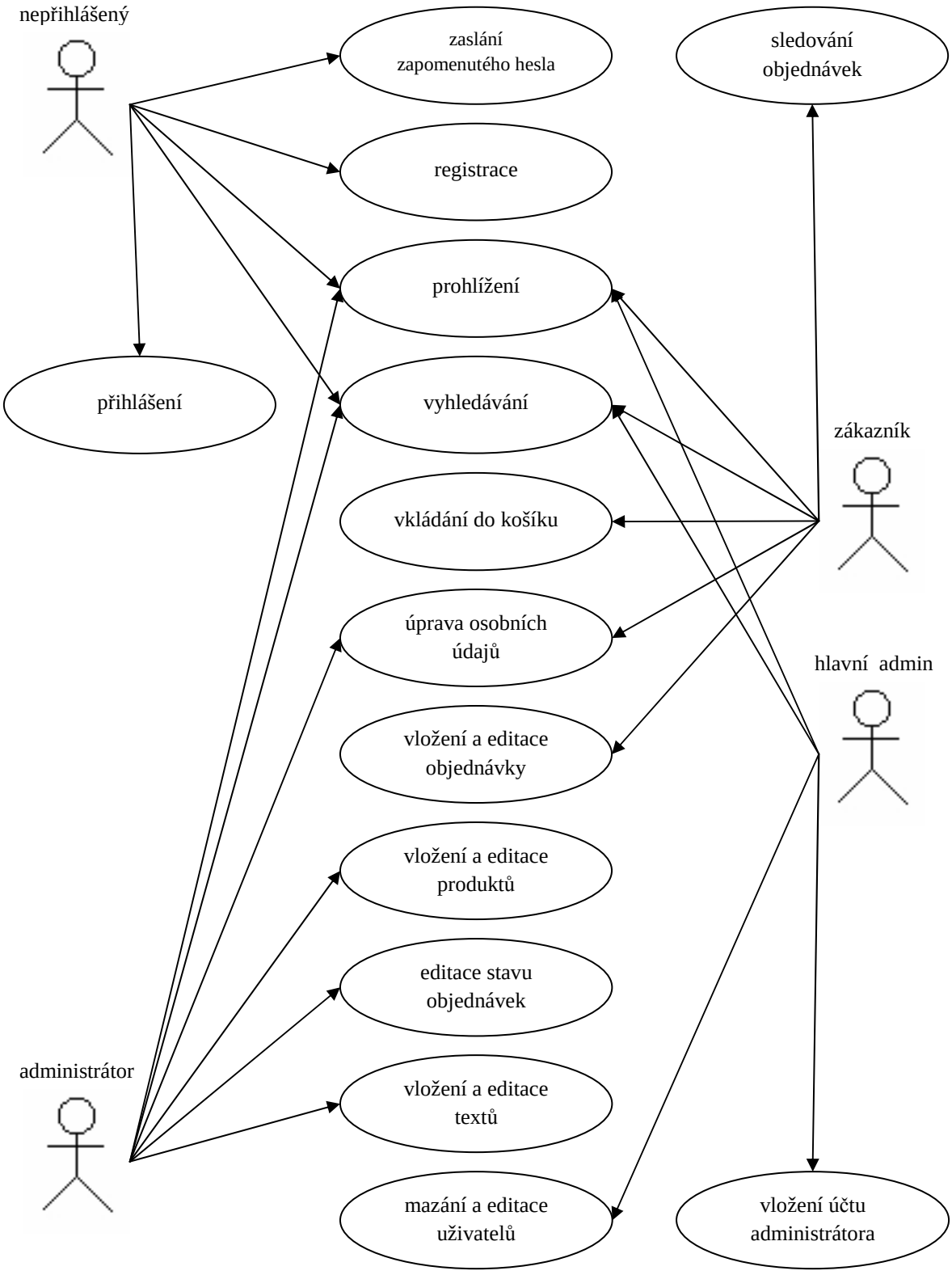
4.6.3 Administrátor

Obsluha internetového obchodu s administrátorskými právy vkládá a edituje produkty, vkládá a edituje textové části, spravuje kategorie a dohlíží na vyřizování objednávek.

4.6.4 Master administrátor

Nejvyšší stupeň oprávnění má role hlavního administrátora. Ten může editovat veškeré účty v aplikaci.

4.6.5 UML Activity diagram



Obrázek 7 - UML Activity diagram

4.7 Principy aplikace

4.7.1 Registrace

Každý anonymní uživatel má možnost se zaregistrovat. K tomuto účelu slouží registrační formulář. Tato část aplikace musí být velice dobře ošetřena z důvodu možnosti přístupu anonymních uživatelů, potažmo rizika napadení systému metodou SQL Injection.

Vstupní formulář obsahuje položky pro získání základních informací o zákazníkovi nutných pro přihlášení a objednání zboží. Z důvodu možného napadení musí vývojář zajistit, aby nebylo možné zadávat do formuláře hodnoty, které např. nekorespondují s datovými typy skutečných údajů. Lze toho dosáhnout pomocí regulárních výrazů v kombinaci s funkcí Oracle oci_bind_by_name, která specifickým způsobem vkládá PHP proměnné do dotazu, a tak zajišťuje vyšší bezpečnost dotazování.

Po odeslání formuláře se vytvoří záznam v tabulce uživatelé, který je dále propojen s vytvořeným záznamem v tabulce *přihlášení* a *kontakty*. *Id role* odpovídá statusu zákazníka z tabulky *role*. Po úspěšném uložení údajů je uživateli zaslán e-mail se zadanými přihlašovacími údaji. [4]

```
//PSČ smí obsahovat pouze 5 číslic
if ((strlen($_POST['form_psc'])==0) OR
    (!ereg("^[0-9]{5}$", $_POST['form_psc'])))
{
     echo „Hodnota PSČ není zadána správně!“;
}
```

4.7.2 Přihlášení uživatelů

Princip přihlášení je založen na dotazování se funkce fce_prihlaseni s parametry zadaného přihlašovacího jména a hesla. V případě úspěchu nalezení kombinace z pohledu *login* vrací funkce hodnotu 1, v opačném případě hodnotu 0 (počet nalezených záznamů).

```
CREATE OR REPLACE FUNCTION "OBCHOD"."FCE_PRIHLASENI" (moje_prihl_jm IN
NVARCHAR2, moje_prihl_heslo IN NVARCHAR2)
RETURN NUMBER
AS
 v_id NUMBER;
BEGIN
  SELECT COUNT(login.id) INTO v_id
  FROM obchod.login
  WHERE login.login=moje_prihl_jm AND login.heslo=moje_prihl_heslo;
  RETURN v_id;
END;
```

4.7.3 Rozpoznávání uživatelů

Princip přidělování rolí (a z toho vyplývající možnosti systém využívat) je velice důležitým prvkem aplikace. Musí být tedy vhodným způsobem zabezpečeno.

Při procesu přihlašování do samotné aplikace obchodu se uživateli přidělí ID role, kterou vrací úspěšná autentizace dle databáze. Na úrovni aplikace jsou samotné PHP skripty ošetřeny proti vykonání neoprávněným uživatelem dle tohoto ID. Čili „pokud nejsi administrátor, nemůžeš vykonat tento skript“.

Druhotným zabezpečením je i přístup k objektům v samotné databázi. K tomuto účelu byli vytvořeni uživatelé s různými druhy nejnutnějších oprávnění k databázovému schématu *obchod*, ve kterém jsou umístěny veškeré objekty. Způsob připojování k databázi je uveden v příloze B.

```
CREATE USER  anonymni IDENTIFIED BY a21c3443d3b4865ad487343cac6;
CREATE USER  zakaznik IDENTIFIED BY a2db4563efa3128abbd9013c5ce;
CREATE USER  spravce IDENTIFIED BY a1f3d4a6b58acd94d2d3cef617e;
CREATE USER  masterspravce IDENTIFIED BY a13C4763F4a7623bc486312fab5;
```

```
GRANT CREATE SESSION TO anonymni;
GRANT SELECT ON obchod.kategorie TO anonymni;
GRANT SELECT,INSERT ON obchod.kontakty TO anonymni;
...
```

4.7.4 Objednávka

Objednávání zboží z pohledu zákazníka je velice jednoduché (vkládání zboží do košíku, následné odsouhlasení dodacích údajů, obchodních podmínek a samotné odeslání objednávky).

Vzhledem k zaměření této práce je důležité tento proces popsat z pohledu databázového schématu. Uživatel má po přihlášení možnost vkládat požadované zboží do nákupního košíku – zapisování záznamů do tabulky *kosik*. Pokud se jedná o první vložený produkt, vytvoří se záznam v tabulce *objednavky*. Zde dojde pouze k vyplnění *ID* a *ID_uzivatele*, čímž je zajištěna referenční integrita. Vhodným spojením (nalezení záznamu s hodnotou *NULL* u položky *datum_objednani*) se tak dá zajistit, aby se zákazníkovi i po odhlášení a nedokončení nákupu přiřadil po opětovném přihlášení právě tento otevřený nákupní košík.

Teprve až po odeslání konečné podoby objednávky se doplní zbývající položky (tzn. *stav*, *typ úhrady* a *cena*, *typ dopravy* a *cena*).

Při realizaci každé objednávky je v tabulce *kontakty* vytvořen záznam (i v případě totožných údajů s kontaktními údaji v profilu zákazníka). Současně jsou zaručeny tyto principy a možnosti:

- zvolit si jiné fakturační údaje,
- při změně osobních údajů nedojde ke změně fakturačních informací.

4.7.5 Zobrazení skladovaného množství

V detailu každého produktu se zobrazuje aktuální skladované množství kusů. V databázi existuje funkce *fce_zjisti_pocet_skladem*, která přebírá *id* požadovaného zboží a vrací číselnou hodnotu, která odpovídá rozdílu počtu kusů na skladě a počtu kusů v objednávkách se statusem *nevyřízená* a *zpracovává se*.

Pokud je objednávka vyřízená, administrátor jí přidělí status *vyřízená*, přičemž dojde k definitivnímu odečtení množství z tabulky *produkty*.

4.7.6 Mazání vs. zneviditelnění

Produkty lze i mazat nebo znevititelnit. Aby bylo možné produkt zcela odstranit, nesmí se z důvodu referenční integrity nacházet ani na jedné objednávce (zboží by se z již provedených objednávek ztratilo, což nesmí nastat). Pokud není tato podmínka splněna, lze produktu přidělit status neviditelnosti a znemožnit tak jeho další objednávání. Na již provedených objednávkách se tudíž nic nezmění.

Zneviditelnění má i výhodu v tom, že vyhledávače tuto možnost nerozlišují a mohou do internetového obchodu nasměrovat např. potenciální zákazníky za účelem získání informací o starších, již nenabízených produktech – vstup do podvědomí zákazníka.

4.7.7 Správa obsahu

Aplikace umožňuje administrátorovi vkládat a upravovat detaily produktů včetně nahrávání obrázků zboží nebo editaci stromové struktury kategorií.

4.7.8 Správa objednávek

Administrátor obchodu přiřazuje objednávce status dle jejího aktuálního rozpracování (*nevyřízeno*, *zpracovává se*, *vyřízeno*) a platbě (*zaplaceno*, *nezaplaceno*). O každé změně tohoto stavu je informován zákazník prostřednictví e-mailu nebo jej lze sledovat v uživatelském účtu.

4.7.9 Správa uživatelských účtů

Přihlášený zákazník i administrátor mají možnost měnit své přihlašovací i osobní údaje. Vhodným ošetřením je zajištěno, že dva uživatelé nemohou mít totožné přihlašovací jméno. O změnách je opět uživatel informován prostřednictvím e-mailu.

Jediný, kdo může odstraňovat zákaznické účty a přidávat administrátorské pozice, je hlavní administrátor (role masteradmin). Před tím, než dojde ke konečnému smazání uživatelského profilu, je zjištěno, zda má uživatel realizované nějaké objednávky. Na základě těchto informací je hlavní administrátor upozorněn na skutečnost, že spolu s uživatelským účtem odebere i jeho dosavadní objednávky.

Zároveň je zajištěno, že nemůže být ze systému odebrán účet hlavního administrátora. Aplikace tak vždy obsahuje alespoň jednoho uživatele.

4.7.10 Vyhledávání

V aplikaci je kladen důraz i na vyhledávání. V rozsáhlé nabídce se může zákazník snadno ztratit. Na internetu se lze dost často setkat s aplikacemi internetových ochodů, které tuto problematiku nemají zcela domyšlenou. Vyhledávání vůbec nefunguje nebo je velice nepřesné. Zákazník tak obchod opouští v domnění, že zboží v nabídce není, i když je tomu ve skutečnosti naopak. Díky vhodnému algoritmu je ale možné zajistit si spokojeného a vracejícího se zákazníka.

Vytvořená aplikace je založena na vyhledávání zadaných slov v názvech nebo částech názvů zboží. Délka každého hledaného slova musí mít z důvodu přesnějšího vyhledávání alespoň 3 znaky. Z těchto řetězců se následně vytvoří pomocí PHP cyklu podmínka, která se spojí se SQL dotazem. Výsledkem je zobrazení produktů, které jsou sjednocením nalezených záznamů v databázi pro každý výraz.

4.7.11 Zapomenuté heslo

V systému je vyřešena i skutečnost, že registrovaný zákazník může zapomenout přihlašovací heslo. V případě, že do příslušného formuláře zadá uživatelské jméno, mu bude obratem na e-mail uvedený v profilu zaslána nově vygenerovaná hodnota hesla.

4.8 SEO optimalizace

Žádná internetová aplikace se neobejde bez SEO optimalizace. [13] Existuje celá řada řešení. V tomto internetovém obchodu jsou však implementovány jen základní možnosti optimalizace.

4.8.1 Základní pravidla

- vhodná definice <title>*název webu*</title> v HTML hlavičce (např. ProHas | Fireman III),
- vhodná definice <meta="keywords" content="*klíčová slova*"> (např. obchod hasiči požární sport zabezpečení staveb ochrana …),
- při použití kaskádových stylů využívat pro klíčová slova nadpisů <h1>, <h2>, <h3>, … (např. <h1>Fireman III</h1>),
- klíčová slova využívat často v textu, ovšem s mírou,
- vkládat klíčová slova do popisků ALT u definice obrázků (např. <img src="/obrazky/fireman-iii.jpg" alt=" Fireman III">).

4.8.2 Přepisovací pravidla

Další možností, jak se dostat do podvědomí vyhledávačů, je podstrkávání obsahu. [8] Toho se dá využít např. pro nahrazení URL adresy s předávanými proměnnými metodou GET na uživatelsky přívětivější a čitelnější.

vstup: www.prohas.cz/index.php?kat=vyzbroj&podkat=obleky&prod=fireman-III

výstup: www.prohas.cz/vyzbroj/obleky/fireman-III.html

```
# zapnutí mod_rewrite (přepisování)
RewriteEngine On
# podmínka existence souboru
RewriteCond %{REQUEST_FILENAME} !-f
# podmínka existence adresáře
RewriteCond %{REQUEST_FILENAME} !-d
# podmínka zadaného URL
RewriteCond %{REQUEST_URI} ^(.*)/(.*)/(.*)\.html
# podstrčení
RewriteRule ^(.*)/(.*)/(.*)\.html$ index.php?kat=$1&podkat=$2&prod=$3
[L,QSA]
```

5 Závěr

Cílem této práce bylo vytvořit funkční internetový obchod se sortimentem týkajícím se požární ochrany. Při vývoji jsem využíval své postřehy z toho prostředí a snažil se zakomponovat do aplikace co možná nejjednodušší a nejkomfortnější způsob pro orientaci zákazníka v obchodě s důrazem na správnou a logickou funkčnost .

Při výběru použitých technologií jsem vycházel ze znalostí, kterých jsem nabýval během studia a v zaměstnání u společnosti zabývající se poskytováním Internetu a doplňkových služeb. Jednalo se především o volbu skriptovacího jazyka PHP a databázového systému Oracle XE. V obchodu jsem využil i technologie, se kterými jsem se setkal poprvé. Jedná se především o knihovnu PHP PEAR, která mi výrazně zjednodušila problematiku odesílání e-mailů.

Ve výsledku byly splněny veškeré požadavky kladené v zadání. Projekt jsem se snažil rozpracovat do co největších detailů. Samozřejmě, že se nejedná o zcela dokonalé řešení. V případě dalšího rozšíření by byl kladen důraz na stornování již probíhajících objednávek či editace hodnot týkajících se způsobu úhrady, dopravy, stavu objednávek, změny sazby DPH a také zdokonalení zázemí obchodu tzn. vyřešení způsob objednávání zboží u dodavatelů nebo napojení na výpis z bankovního účtu. Vhodné by bylo rovněž implementovat logovací systém pro uchovávání informací o veškerém dění v aplikaci.

Nad rámec zadání byl obchod opatřen jednoduchou optimalizací pro vyhledávače.

Literatura

[1] Databáze. Wikipedie, otevřená encyklopedie [online]. 2010 [cit. 2010-02-12]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Databáze>.

[2] Chez Xaver, Ubuntu, PHP5, OCI8 and PDO_OCI : the perfect install [online]. 2010 [cit. 2009-12-03]. Dostupný z WWW: <http://www.lacot.org/>.

[3] Teorie relačních databází: Normalizace. Manualy.net [online]. 2010 [cit. 2010-04-01]. Dostupný z WWW: <http://www.manualy.net/article.php?articleID=13>.

[4] Kontrola formulářových údajů v PHP. Interval.cz [online]. 2010 [cit. 2010-04-08]. Dostupný z WWW: <http://interval.cz/clanky/kontrola-formularovych-udaju-v-php/>.

[5] SQL Injection. Wikipedie, otevřená encyklopedie [online]. 2010 [cit. 2010-04-10]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/SQL_injection>.

[6] SQL injection: Princip a ochrana. Investice - Ihned [online]. 2010 [cit. 2010-04-10]. Dostupný z WWW: <http://digiweb.ihned.cz/c4-10122900-19732020-i00000_d-sql-injection-princip-a-ochrana>.

[7] Normalizace. BI Experts [online]. 2010 [cit. 2010-04-11]. Dostupný z WWW: <http://www.biexperts.cz/index.php/component/content/article/18-ctsql/31-arnormalization.html>.

[8] Mod_rewrite a hezké url díl II. Mikův Weblog [online]. 2010 [cit. 2010-04-11]. Dostupný z WWW: <http://mike.treba.cz/mod_rewrite-a-hezke-url-dil-ii/>.

[9] Databázová bezpečnost. NaWEBka [online]. 2010 [cit. 2010-04-12]. Dostupný z WWW: <http://www.rydval.cz/phprs/view.php?cisloclanku=2006082701>.

[10] RIORDAN, Rebecca. Vytváříme relační databázové aplikace. Praha: Computer Press, 2000. 280 s. ISBN 80-7226-360-9.

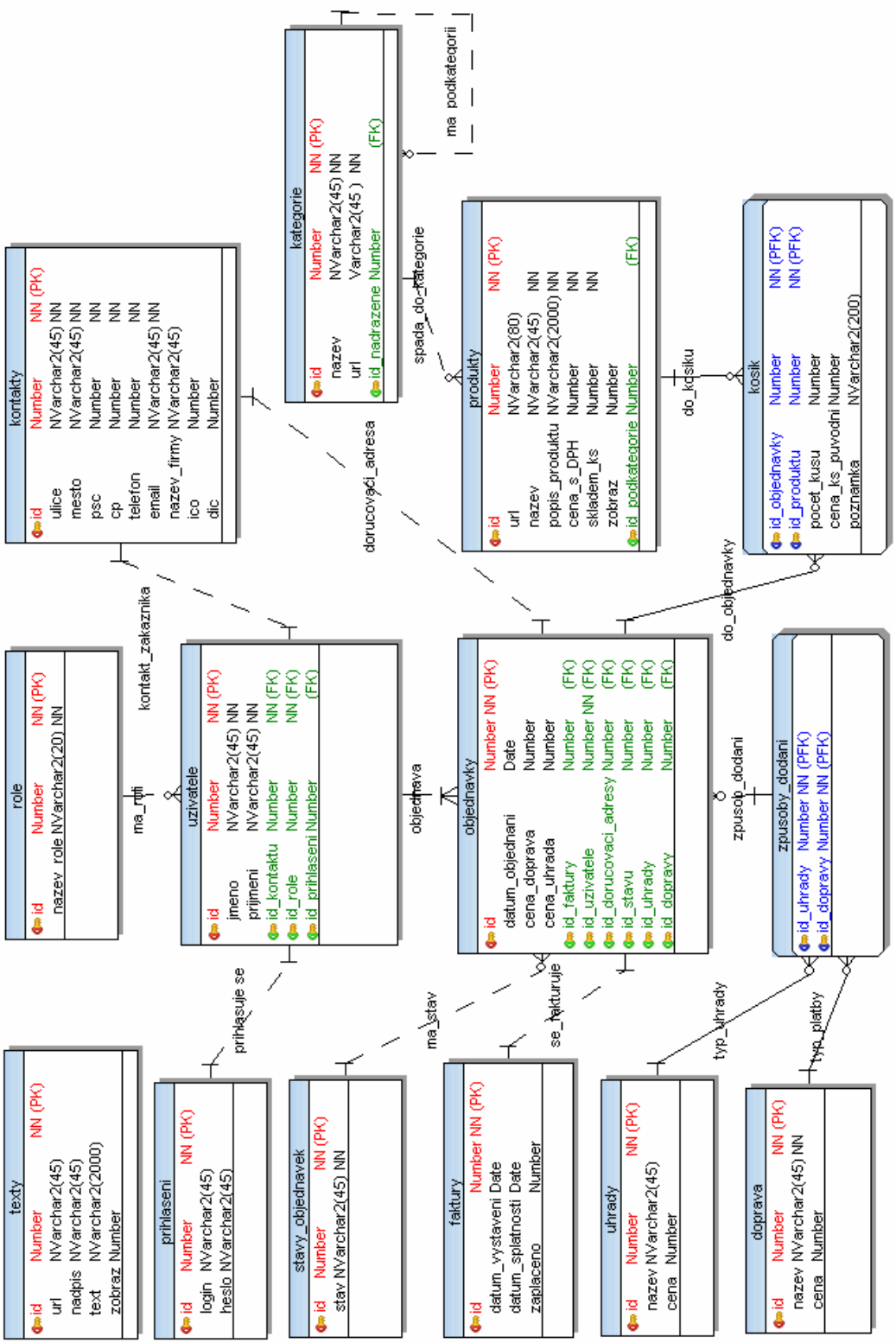
[11] HERNANDEZ, Michael. Návrh databází. Praha: Grada, 2006. 408 s. ISBN 80-247-0900-7.

[12] THERIAULT, Marlene, NEWMAN, Aaron. Bezpečnost v Oracle. Brno: Computer Press, 2004. 515 s. ISBN 80-722-6979-8.

[13] SEO - Search Engine Optimization. Interval.cz [online]. 2010 [cit. 2010-04-15]. Dostupný z WWW: <http://interval.cz/clanky/seo-search-engine-optimization/>.

[14] OPPEL, Andrew.Databáze bez předchozích znalostí. Brno: Computer Press, 2006. 318 s. ISBN 80-251-1199-7.

Příloha A – Fyzický databázový model



Obrázek 8 - Fyzický datový model

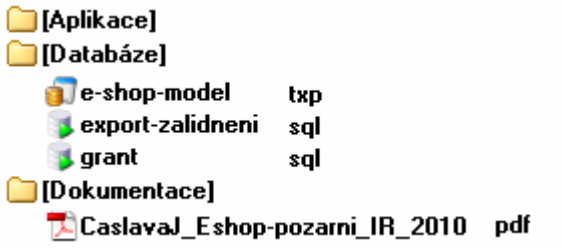
Příloha B – Způsob připojování k databázi

```
function zjisti_kdo(){
if (isset($_SESSION['ses_uz_id'])) {
    if ($_SESSION['ses_uz_role_id']== 1)
      $connect =oci_new_connect('spravce',
      'a1f3d4a6b58acd94d2d3cef617e','//localhost/XE', 'utf8');
    if ($_SESSION['ses_uz_role_id']== 2)
      $connect = oci_new_connect('zakaznik',
      'a2db4563efa3128abbd9013c5ce','//localhost/XE', 'utf8');
    if ($_SESSION['ses_uz_role_id']== 3)
      $connect = oci_new_connect('masterspravce',
      'a13C4763F4a7623bc486312fab5','//localhost/XE', 'utf8');
  }
  else
      $connect = oci_new_connect('anonymni',
      'a21c3443d3b4865ad487343cac6','//localhost/XE', 'utf8');
  return $connect;
}

function dotaz_inject($sql, $bind) {
    $connect = zjisti_kdo();
    $s = oci_parse($connect, $sql);
    foreach ($bind as $k => $v) {
      oci_bind_by_name($s,$k,$bind[$k]);
    }
    oci_execute($s);
    oci_fetch_all($s,$res);
    oci_close($connect);
    return $res;
}
```

Příloha C – Funkce MAIL

```
function posli_email($adresat, $retezec_html, $predmet){
    include('Mail.php');
    include('Mail/mime.php');
    $recipients = $adresat;
    $headers['From'] = 'st18921@student.upce.cz';
    $headers['To'] = $adresat;
    $headers['Subject'] = "ProHas - ".$predmet;
    $param['text_charset'] = 'utf-8';
    $param['html_charset'] = 'utf-8';
    $param['head_charset'] = 'utf-8';
    $params["host"] = 'IP SMTP SERVERU';
    $params["port"] = 'PORT SMTP SEVERU';
    $params["auth"] = TRUE; //vyžaduje SMTP server přihlášení?
    $params["username"] = 'LOGIN SMTP SERVERU';
    $params["password"] = 'HESLO SMTP SERVERU';

    $text = 'Zpráva internetového obchodu ProHas.cz.';
    $html = $retezec_html."<p>S přátelským pozdravem Jan Čáslava, Tým
            obchodu ProHas s.r.o.</p>";
    $crlf = "\n";

    $mime = new Mail_mime($crlf);
    $mime->setTXTBody($text);
    $mime->setHTMLBody($html);

    $body = $mime->get($param);
    $headers = $mime->headers($headers);

    $mail_object = & Mail::factory('smtp',$params);
    $send = $mail_object->send($recipients, $headers, $body);
    if (PEAR::isError($send)) echo "Email se nepodařilo odeslat";
}
```

Příloha D – Adresářová struktura přiloženého CD



Obrázek 9 - Adresářová struktura přiloženého CD

Adresář *Aplikace* obsahuje soubory aplikace.

Adresář *Databáze* obsahuje soubory pro vytvoření základních tabulek včetně podstatných dat a soubor pro vytvoření uživatelských práv včetně přidělení práv.

Adresář *Dokumentace* obsahuje dokumentaci projektu.

Příloha E – Instalace

Pro zprovoznění aplikace je potřeba mít nainstalován webový server např. Apache, PHP s knihovnou OCI8 a databázový server Oracle.

Po přihlášení do správy databázového serveru je potřeba vytvořit uživatele *obchod*. Do tohoto schématu poté importujte databázové objekty pomocí souboru */Databáze/export-zalidneni.sql* a */Databáze/grant.sql*, který nám zajistí vytvoření uživatelů databáze.

Následuje zkopírování obsahu adresáře *aplikace* do kořenového adresáře webového serveru. Pozor, neopomeňte soubor *.htaccess*.

Aplikaci lze spustit zadáním URL adresy do webového prohlížeče. Nyní se můžete přihlásit jako *masteradmin* s heslem *hesloheslo* a přidat administrátorský účet.

Instalace je tímto dokončena.