VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ON-LINE KATALOG ZEMĚDĚLSKÉ TECHNIKY

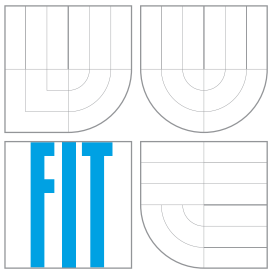
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN MILIČKA

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ON-LINE KATALOG ZEMĚDĚLSKÉ TECHNIKY

ON-LINE AGRICULTURE TECHNOLOGY CATALOGUE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MARTIN MILIČKA
AUTHOR

VEDOUCÍ PRÁCE — Ing. RADEK BURGET, Ph.D.
SUPERVISOR

BRNO 2008

Zadání bakalářské práce/6919/2007/xmilic01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Milička Martin**
Obor: Informační technologie
Téma: **On-line katalog zemědělské techniky**
Kategorie: Web

Pokyny:

1. Seznamte se s požadavky firmy P&L na aplikaci on-line katalogu zemědělské techniky.
2. Seznamte se se softwarovým vybavením serverů, které budou použity pro provoz aplikace a na základě toho zvolte implementační platformu.
3. Proveďte návrh dané aplikace. Zaměřte se na vhodnou reprezentaci parametrů prezentované techniky za účelem jejich možného srovnání a vyhledávání podle parametrů.
4. Implementujte aplikaci na zvolené platformě.
5. Vhodným způsobem realizujte uživatelské rozhraní aplikace tak, aby byly v co největší míře eliminovány nevýhody webového rozhraní.
6. Po dohodě se zadavatelem proveďte praktické nasazení aplikace.
7. Zhodnoťte dosažené výsledky a navrhněte pokračování projektu.

Literatura:

- Flanagan, D.: JavaScript: kompletní průvodce. Computer Press, 2002.
- Gilfillan I.: Myslíme v MySQL4. Grada, 2003.
- Welling, L., Thomsonová, L.: PHP a MySQL - rozvoj webových aplikací - 2. vydání. SOFTPRESS, 2003.
- Williams, H.E., Lane, D.: PHP a MySQL - Vytváříme webové databázové aplikace. Computer Press. 2002.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Burget Radek, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2007
Datum odevzdání: 14. května 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2
L.S.

doc. Ing. Jaroslav Zendulka, CSc.
vedoucí ústavu

Licenční smlouva je uvedena v archivním výtisku uloženém v knihovně FIT VUT v Brně.

Abstrakt

Hlavním úkolem této práce bylo vytvořit on-line katalog zemědělské techniky. Tento katalog umožňuje vkládat stroje do hierarchicky poskládaných kategorií. Kategorie nám umožňují seskupovat stroje s podobnými vlastnostmi a účelem použití. Každý stroj uložený v katalogu může obsahovat libovolné množství parametrů s příslušnými hodnotami. Předností tohoto katalogu je umísťování strojů na pakoviště a možné následné srovnávání strojů podle parametrů. Systém administrace je postaven na modulárním základu.

Klíčová slova

katalog, zemědělská technika, modularita, informační systém, XHTML, CSS, PHP, MySQL, SMARTY, SWFUpload, friendly URL

Abstract

The main goal of this project was to create an online agriculture technology catalogue. This catalogue allows managing machines in hierarchically organized categories. The categories allow grouping machines with similar parametres and purpose. Each machine stored in the catalogue can be assigned an arbitrary number of parameters with values. Another advantage of the catalogue is the possibility of placing selected machines on a parking place with the possibility of comparing their parameters. The system of administration has been designed at a modular principle.

Keywords

catalogue, agriculture technology, modularity, information system, XHTML, CSS, PHP, MySQL, SMARTY, SWFUpload, friendly URL

Citace

Martin Milička: On-line katalog zemědělské techniky, bakalářská práce, Brno, FIT VUT v Brně, 2008

On-line katalog zemědělské techniky

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Radka Burgeta. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Milička
14. května 2008

Poděkování

Rád bych poděkoval panu Radku Burgetovi za vedení, pomoc a připomínky týkající se této práce. Zaměstnancům zadavatele za odborné rady, které se vztahovaly k zadání. Velký dík patří také všem těm, kteří si přečetli tento text a odstranili v něm chyby.

Obsah

Kapitola 1

Úvod

1.1 Seznámení

Tato práce se zabývá tvorbou systému, jenž obsahuje universální modulárně rozšiřitelnou administraci. Celý systém, který je rozebírán v této práci, je použit pro katalog zemědělské techniky. Při návrhu tohoto systému bylo dbáno na možnou další funkční rozšířitelnost. To znamená, že byla snaha systém vytvořit tak, aby případné rozšíření systému nedalo velkou režijní práci. Díky této modulární vlastnosti je tedy systém možné dále rozšiřovat a pak používat pro různorodé webové aplikace.

1.2 Zadavatel

Zadavatelem této práce je společnost P & L spol. s r.o. Biskupice, zabývající se prodejem a servisem zemědělské techniky, která očekává vytvoření webové aplikace v pobobě katalogu zemědělské techniky pro doménu www.agrobazar.cz. Nová webová aplikace by měla nahradit stávající funkčně nevyhovující webovou aplikaci od společnosti Avonet[1].

1.3 Účel práce

Hlavní důvodem pro vytvoření takového katalogu byla skutečnost, že se na českém internetu nevyskytuje žádná ucelenější aplikace, která by se zabývala prodejem použité zemědělské techniky. Od tohoto katalogu se očekává schopnost konkurovat již existujícím zahraničním katalogům. Je nutné zmínit, že katalogy podobného typu se velmi osvědčily v automobilovém průmyslu, tudíž je od tohoto katalogu očekáváno zvýšení prodeje zmiňované *bazarové zemědělské techniky*. Informace o bližší specifikaci požadavků zadavatele je možné si přečíst v kapitole 4.1, která se tomuto tématu přímo věnuje.

[1] Jedná se o společnost působící v oblasti internetových služeb ve Zlínském kraji. Jejím pilotním produktem je modulární webový systém WEBSYSTEM, který vzhledem k tomu, že je vytvořen na příliš obecné úrovni, není schopen vždy uspokojit speciální požadavky zákazníků.

Kapitola 2

Základní pojmy

2.1 Webová aplikace

Pojem webová aplikace lze jednoduše chápat jako aplikaci pracující v prostředí sítě internet. K tomu, abychom mohli uživatelské rozhraní takové aplikace zobrazit, je nutné mít nainstalovaného webového klienta[1].

Jelikož je webová aplikace přístupná přes síť internet, je nutné si uvědomit, že uživatelské rozhraní zobrazené prostřednictví prohlížeče se na počítači nevyskytuje lokálně, ale při každém požadavku je stahováno prostřednictvím sítě internet z jiného počítače, kterému se říká server. U těchto aplikací se s výhodou využívá centrálního umístění, které při jakýchkoliv změnách zobrazí tyto změny okamžitě všem svým uživatelům vyskytujícím se kdekoliv na světě. Je zde pouze jedna podmínka a tou je přístup na internet.

Mezi standardní aplikace, které pracující na tomto principu, patří různé redakční systémy, internetové obchody a podobně.

Tento návrh aplikace se sebou nese několik podstatných výhod a také nevýhod. Jednou z hlavních výhod je možnost přístupu k těmto aplikacím v okamžiku existence připojení k internetu kdekoliv na světě, bez ohledu na fyzické umístění této aplikace.

Nevýhodou webových aplikací vzhledem ke klasickým aplikacím, které pracují lokálně na počítači je, nutnost existence již zmiňovaného připojení k internetu. Nebereme-li v potaz nějakou firemní síť s vlastním serverem, na kterém by jsme takovou aplikaci měli umístěnu. Obecně tedy bez připojení není aplikace dostupná. Při práci se rychlost odezvy aplikace odvíjí od použitého internetového připojení. Tvorbou webových aplikací se zabývá zdroj [1].

2.2 Architektura klient-server

Z hlediska centrální správy se při tvorbě rozsáhlých aplikací, které používá současně několik lidí, dospělo k následujícímu názoru. Jde o to, že existuje počítač (server), na kterém se vyskytují všechny aplikace, které jsou společné pro skupinu počítačů. Tyto aplikace se pak stanou díky síti přístupné z ostatních počítačů, přičemž instalace jednotlivých aplikací se bude vyskytovat pouze na jedné stanici, která k nim bude poskytovat přístup. Výhodou této architektury je, že se nemusíme zvlášť starat o několik stejných instalací určité aplikace

[1] Speciální program, který graficky zobrazuje a zpracovává HTTP požadavky. V běžné praxi se tomuto programu přezdívá webový prohlížeč. Mezi nejčastější prohlížeče patří Microsoft Internet Explorer, Mozilla Firefox, Opera atd.

na více stanicích. Obecně platí, že služby serveru můžeme využívat z libovolného počtu počítačů připojených v jedné síti. Omezením zde může pouze být počet možných licencí.

Pojem architektury klient-server je inspirován zpracováním požadavků. Předpokladem je, že požadavky klienta jsou odeslány a zpracovávány na straně serveru. Klient pouze zasílá požadavky na server, který je poté zpracovává. Výsledeky požadavků jsou po zpracování serverem zasílány klientovi, jenž je zobrazí uživateli.

Funkce klienta je v této architektuře pouze dotazovací a interpretovací. Naopak funkce serveru se stává vykonávací. Všechny funkční úkony jsou řešeny na straně serveru.

2.3 HTTP protokol

Jedná se o protokol pracující na aplikační vrstvě. Transport tohoto protokolu zajišťuje transportní protokol TCP/IP. Protokol HTTP patří do skupiny textových protokolů. Jedním z jeho výhod je, že příkazy tohoto protokolu nejsou zbytečně složité. Můžeme mluvit o odlehčenosti protokolu – existují zde pouze podstatné příkazy, bez kterých by protokol nebyl schopen spolehlivě fungovat.

Jednou z věcí, která není při činnosti protokolu řešena, je fakt, že je HTTP protokol bezstavový. Tudíž při jeho použití musíme počítat s tím, že si nějakými podpůrnými prostředky musíme zajistit schopnost neustále udržovat kontext. Protokol je využíván na architektuře klient-server. Klient zašle požadavek, server jej zpracuje a vrátí odpověď. Protokol obsahuje několik příkazů. Mezi nejčastější příkazy protokolu patří POST a GET, na něž server podle očekávání odpoví. V protokolu jsou implementovány dva základní typy zpráv. První je *Request* – žádost o data ze serveru. A druhá je *Response* – odpověď ze serveru.

Protokol při odkazování na dokumenty v rámci sítě internet používá URL[2] adresy.

V dnešní době protokol HTTP při své činnosti využívá formát MIME[3], který je schopen přenášet přes protokol HTTP téměř jakýkoliv formát souboru. Část těchto informací byla převzata ze zdroje [2].

[2]Uniform Resource Locator - jedná se o způsob jednoznačného určení dokumentu v rámci sítě internet

[3]Multipurpose Internet Mail Extensions – jedná se o standard rozšiřující možnost přenosu diakritiky, obrázků, souborů atd.

Kapitola 3

Použité technologie a techniky

3.1 HTML, XHTML

Jazyk HTML se používá ke tvorbě hypertextových dokumentů. Můžeme jej zařadit mezi značkovací jazyky. Jedná se o základní jazyk na tvorbu dokumentů prezentovaných v prostředí sítě internet. Jeho původ je v universálním značkovacím jazyku SGML. Vlastnost značkovacích jazyků je taková, že nám umožňují logicky označovat jednotlivé části textu zpracovávaného dokumentu.

Od nasazení své první verze jazyka HTML do reálné praxe, prošel tento jazyk řadou změn a vylepšení, které jsou popsány v příslušných specifikacích. V současnosti nejvyšší specifikace tohoto jazyka je HTML 4.01.

Standardy v prostředí internetu má na starosti konsorcium W3C[1].

Kromě jazyka HTML je při tvorbě hypertextových dokumentů využíváno také jazyka XHTML. Jeho původ je v XML. XHTML byl upraven tak, aby se svou syntaxí blížil jazyku HTML. Díky specifikaci XML je tento jazyk mnohem striktnější. Více informací ve zdroji [3].

Při tvorbě dnešních hypertextových dokumentů je hlavním cílem oddělit specifikaci grafické části dokumentu od samotné části obsahu. Za tímto účelem vznikly kaskádové styly CSS.

3.2 CSS

Aby se v dokumentech publikovaných v prostředí internetu oddělila grafická část od textové, vznikly kaskádové styly. Řešení této problematiky obvykle aplikujeme prostřednictvím nového souboru, který nazýváme stylový dokument. Pokud chceme, aby byl hypertextový dokument ovlivněn tímto stylovým dokumentem, musíme provést jeho explicitní připojení.

Bez stylového dokumentu se musí HTML dokument zobrazit bez námi vytvořeného formátování. Nesmí se u něj projevit žádné speciální projevy jiného než standardního formátování.

Pouze díky změně stylového souboru máme možnost kompletně změnit podobu vytvářeného webu. Můžeme tedy vytvořit několik grafických šablon pro jeden obsahově stejný web. Pro uživatele, který si prohlíží stránky, to může být příjemné překvapení.

Nezmíněnou skutečností, která byla také jedním z hlavních důvodů nasazení tohoto jazyka, bylo centralizování nastavení vzhledu do jednoho souboru. Změna grafického vzhledu

[1] World Wide Web Consortium - mezinárodní konsorcium, které vzniklo za účelem sjednocení norem pro dokumenty publikované v rámci sítě internet.

byla před nasazením tohoto jazyka velkým problémem. Bylo totiž nutné projít všechny dostupné zdrojové kódy a případné úpravy vzhledu v nich změnit. Díky kaskadovým stylům není zásah do zdrojových souborů nutný. Stačí pouze upravit stylový soubor, který bývá společný pro celý web. Celá úprava grafického vzhledu je tedy podstatně jednodušší a také značně rychlejší.

Jako u všech standardů, tak i v tomto případě dochází postupem času k různým změnám ve specifikaci. V době psaní tohoto textu existuje standard pro CSS verze 2, objevují se však zmínky o brzkém nasazení verze 3.

3.3 JavaScript

Protože se dříve při tvorbě webových aplikací objevila potřeba zpracovávat některé události na straně klienta, vznikl za tímto účelem jazyk JavaScript. Tento jazyk patří do rodiny interpretovaných programovacích jazyků. Ve svém základu se jedná o jazyk pracující na straně klienta, ale podle informací publikovaných na internetu je možné jej použít také na straně serveru. Více informací o této problematice jsou v [4].

Jazyk Javascript patří do skupiny objektově orientovaných programovacích jazyků. Jeho charakteristickým rysem je potlačená typová kontrola.

Používaným pojmem ve spojení se jménem JavaScript je tzv. *klientský JavaScript*. Jedná se o integraci JavaScriptu do internetového prohlížeče. Klientský JavaScript používá objektový model dokumentů DOM. Díky této vlastnosti je možné dosáhnout dynamického chování webových stránek. Tato vlastnost se označuje pojmem DHTML (Dynamic HTML).

Skripty napsané v jazyce JavaScript se obvykle vyskytují v externím souboru. Připojení tohoto souboru se provádí v části HEAD dokumentu HTML. Tyto skripty mohou být samozřejmě také přímou součástí HTML dokumentu. Pokud bychom však měli rozsáhlý skript v jazyce JavaScript, musíme si uvědomit, že se při každém načítání stránky kromě užitečných dat budou muset také vždycky stahovat zbytečná nedůležitá data. V případě externího souboru se tedy jedná pouze o jedno stáhnutí tohoto souboru do počítače. Znalostí této problematiky můžeme urychlit zobrazování stránek.

V dnešní době je JavaScript a jeho objekt *XMLHttpRequest* často spojován s technikou AJAX. Popis chování této techniky bude příblížen v kapitole 3.4

Při používání jazyka JavaScript bychom neměli zapomínat na skutečnost, že ne všechny prohlížeče jsou schopny interpretovat tento jazyk. Z tohoto důvodu bychom se ho měli snažit využívat pouze minimálně. Informace o tomto jazyku je možné najít v publikaci [5].

3.4 Ajax

Jedná se o nově používanou techniku při tvorbě webových aplikací. Slovo AJAX je zkratkou anglického slova *Asynchronous JavaScript and XML*. Už z názvu je zřejmé jaké technologie jsou s touto technikou spjaty.

V souvislosti s touto technikou se často mluví o používání asynchronních dotazů na server. K tomuto účelu se využívá objektu *XMLHttpRequest* z jazyka Javascript, který musí být samozřejmě touto technikou podporován. Výsledkem takového dotazu obvykle bývá odpověď ve tvaru standardního XML dokumentu, který pak můžeme dále zpracovávat. Není to však nutnou podmínkou. Při zpracování odpovědi ze serveru se obvykle využívá schopnosti práce s objektovým modelem DOM. Díky němu jsme pak schopni přijatou odpověď zpracovat a zakomponovat do již existujícího HTML dokumentu.

Ajax nám umožňuje modifikovat objektový model HTML dokument bez nutnosti obnovovat celou webovou stránku.

Hlavní výhodou této techniky je, že můžeme tvořit mnohem interaktivnější weby, které nám okamžitě zobrazují výsledek. Na základě předešlého textu si musíme uvědomit tu výhodu, že předností této techniky je práce s mnohem menšími objemy dat, než v případě získávání data pro celou stránku. Daleko více informací o této problematice je možné nalézt v publikaci [6], kde bylo také čerpáno pro tento text.

3.5 PHP

Tento jazyk je oproti jazyku JavaScript využíván ke skriptování na straně serveru. V současné době se jedná asi o nejpoužívanější programovací jazyk v oblasti webových aplikací. Této skutečnosti také přeje fakt, že je serverová aplikace pro tento programovací jazyk OPEN SOURCE[2].

Užití PHP je následující. Klient zašle určitý požadavek na server, který jej zpracuje příslušným PHP skriptem. Výsledek po zpracování je zasílán ve formě HTML dokumentu klientovi. Ten pak případný výsledek zobrazí uživateli.

V současné době je nám jasný rozdíl mezi chováním skriptovacího jazyka na straně klienta a na straně serveru. Jelikož jsou data při zpracovávání skriptu na straně serveru přeposílána mimo prohlížeč, je nutné, aby se při volání nového skriptu vždy provádělo obnovení stránky. Tato povinnost nám při zpracování na straně klienta odpadá.

Programovací jazyk PHP nám umožňuje procedurální a od verze PHP 5 také objektově orientované programování. Programování pomocí objektů se může značně projevit v efektivitě výsledného kódu. Jazyk PHP patří mezi jazyky, kde není nutné předem definovat typy proměnných. Proměnné mohou během svého života několikrát měnit svůj typ. Můžeme tedy říct, že se jedná o jazyk s dynamicky typovanými proměnnými.

3.6 MySQL

Databáze MySQL patří do kategorie relačních databází. Pojem databáze můžeme chápat jako místo, jehož cílem je systematicky ukládat data. Ty pak můžeme díky speciálnímu dotazování zpracovávat. Databáze MySQL je multiplatformní. Je možné ji provozovat na různých počítačových platformách.

Tato databáze je obvykle používána ve spojení s programovacím jazykem PHP. Tento jazyk dokáže přistupovat k datům uloženým v této databázi, vkládat do ní nová data a provádět také jejich editaci. Databáze obsahuje tabulky tvořené sloupci, které obsahují názvy ukládaných proměnných. Řádkům tabulky odpovídají jednotlivé záznamy. Data pro příslušné sloupce jsou na každém řádku uložena podle datových typů jednotlivých sloupců.

Komunikace s databázovým serverem probíhá na základě jazyka SQL[3]. Pro snažší správu databáze MySQL vznikla webová aplikace *phpMyAdmin* napsaná v jazyce PHP. Jedná se o volně šířitelnou, hojně využívanou aplikaci.

[2]Tímto termínem bývá označován počítačový software s volně přístupnými zdrojovými kódy, které lze prohlížet a za splnění určitých podmínek upravovat.

[3]Structured Query Language - dotazovací jazyk pro práci s daty v relačních databázích. Bližší informace se můžeme dočíst v [7]

Aby bylo možné používat databázi ve webových aplikacích, je nejprve nutné, abychom se připojili k databázovému serveru. Po úspěšném připojení pak můžeme posílat dotazy do databáze a očekávat od ní patřičné odpovědi či chování.

3.7 Sessions

Protože nejsme schopni pomocí protokolu HTTP udržovat kontext při prohlížení stránek, vznikly za tímto účelem *session*. Díky nim si udržujeme kontext mezi prohlíženými stránkami. Session můžeme chápat jako textový soubor, v němž jsou uloženy zaregistrované proměnné a jejich hodnoty. Soubor s těmito hodnotami je uložen na straně serveru. Session jsou identifikovány použitým webovým prohlížečem, proto se nemůže stát, že by někdo zneužil toto řešení.

Abychom mohli využívat session na stránce, je nutné nejprve aktivovat jejich použití pomocí *session_start()*. Proměnné se do session registrují pomocí *session_register("promenna")*. Zaregistrované session můžeme zrušit pomocí *session_destroy()*. Předpokladem fungování těchto funkcí je povolení *register_globals* na straně serveru na hodnotu *On*.

Pomocí session můžeme v rozsáhlých webových aplikacích udržovat informaci o dostupném přihlášení uživatele. Jedná se o jeden ze způsobů, jak řešit takové přihlašování. Session nám kromě jiného dovolují nastavit její životnost. Jedná se o dobu, po kterou má daná proměnná platnost.

Další charakteristickou vlastností pro životnost session je fakt, že v okamžiku ukončení činnosti prohlížeče jsou všechny registrované session zrušeny – jejich životnost je ukončena. Při novém spuštění prohlížeče se vytvoří nové identifikace session pro použitý prohlížeč.

3.8 Cookies

Jedná se o soubory, které jsou automaticky prohlížečem webových stránek ukládány na pevný disk počítače. V tomto souboru můžeme nalézt příslušné proměnné, jejichž vlastností je existence minimálně po dobu prohlížení webu. Je možné nastavit dobu platnosti těchto proměnných. Jedná se o to, že proměnná bude existovat po dobu, kterou si nastavíme. Tady se nám projevuje jedna z charakteristických vlastností této proměnné. Výhodou je, že i po ukončení činnosti prohlížeče a pak jeho případném spuštění, jsou tyto proměnné uchovávány. Tudíž je možné s nimi později znovu pracovat. Tento způsob práce s potřebou zpětně evidovat obsah nějaké proměnné je z hlediska práce nejefektivnější.

Aby bylo možné používat tyto proměnné, je nutné vytvořit instanci takové cookie proměnné, kterou máme v plánu používat. Po vytvoření instance proměnné je nutné pouze definovat její název a hodnotu. Pokud však máme nějaký rozsáhlý web, je třeba, abychom nastavili také parametry nepovinné. Tato záležitost se nás bude týkat převážné v okamžiku, kdy chceme, aby existence proměnné nebyla pouze v rámci jednoho adresáře webu a nebo třeba pak v rámci jedné domény. Kromě specifikace rozsahu platnosti proměnné, je také možné nastavit dobu její platnosti. Po uplynutí definované doby obsahuje proměnná nedefinovanou hodnotu.

Je třeba ještě zmítit chování *cookies*, které vychází z jejich umístění. Vzhledem k tomu, že je soubor s cookies ukládán na pevném disku, je možné jej volně editovat nebo případně mazat. Musíme si uvědomit, že nemůžeme při použití těchto proměnných vždy spoléhat na jejich nedotknutelnost a stoprocentní pravost. Velikost souboru s proměnnými cookies

je pro jeden web omezena na 4kB s tím, že je možné uložit pouze dvacet takových různých proměnných.

3.9 mod_rewrite

Můžeme říct, že se jedná o techniku pracující na straně serveru. Konkrétně se jedná o modul, který je v současné době standardně součástí serveru Apache. K tomu, aby bylo možné používat tento modul, je nutné, aby byl na serveru povolen. Toto povolení je třeba provést v konfiguračním souboru serveru.

Definice, podle které se provádí přepisování URL adres, se obyčejně ukládájí v kořenovém adresáři webu. Soubor pro tyto definice se jmenuje *.htaccess*. Platí zde pravidlo, že definice umístěná v určitém adresáři se automaticky aplikuje na všechny podadresáře.

Tato technika má uplatnění v případě, kdy chceme vytvářet tzv. *Cool URIs*. Využívá se to převážně při tvorbě optimalizovaných webů pro vyhledávače – SEO[4]. Vycházíme ze schopnosti dostat důležitá klíčová slova také do adresy webu, což u vyhledávačů zvyšuje relevanci daného klíčového slova vzhledem k odkazu.

Dalším důvodem proč vytvářet takové odkazy je fakt, že návštěvník je schopný mnohem snáze si zapamatovat adresu webu. Odpadá zde tedy nutnost pamatování případných číselných parametrů, které bývávají předávány prostřednictvím URL, a které se stávají hůře zapamatovatelné. Výhodou je, že tato adresa webu obvykle dává smysl vzhledem k jeho obsahu. Jak se dají přepisovat adresy se můžeme dočíst v kapitole [8].

3.10 Smarty

Aby se dosáhlo oddělení programové části katalogu od grafického vzhledu, používá se šablonovací systém. Výhodou je, že je možné oddělit práci kodéra od práce programátora. Z volně dostupných aplikací byl pro vytvářený katalog vybrán šablonovací systém pro PHP – *Smarty*. Práce se šablonovacím systémem je založena na tvorbě nazávislé šablony, jež odpovídá nějakému standardu hypertextového dokumentu. V této šabloně se pak vyskytují speciální proměnné, které jsou při použití nahrazovány skutečnými hodnotami. Při činnosti tedy dochází k substituování proměnných za skutečné hodnoty.

3.11 SWFUpload

Jelikož není technicky možné provádět upload více jak jednoho souboru prostřednictvím jednoho formulářového prvku k tomu určenému, vznikly za tímto účelem aplikace, které tento problém řeší. Ve vytvářeném katalogu byla použita knihovna *swfupload* vyvinutá společností *Mambo.se*. Více informací je ve zdroji [9].

Jedná se o Javascript/Flash aplikaci, která se nám postará o zmiňovaný multiupload. Tuto aplikaci s výhodou využijeme v tomto projektu při uploadu fotografií produktů. Jednou z hlavních výhod této aplikace je schopnost uživatele pomocí jednoho formulářového prvku vybrat více než jeden soubor pro upload. Díky tomu, že aplikace při své činnosti využívá Flash, celý upload probíhá na pozadí, odpadá tedy nutnost obnovování stránky. S využitím objektového modelu dokumentu se po uskutečnění uploadu změna okamžitě projeví na aktuální stránce zobrazováním úspěšně uploadovaných fotek.

[4] Search Engine Optimalization – cílem této techniky je dosáhnout co nejlepšího umístění adresy webu pro dané klíčová slova na prvních pozicích ve vyhledávačích.

3.12 RTE editor

Využití tohoto editoru nastává v případech, kdy chceme mít prostřednictvím administrace možnost formátovat určité informativní textové části aplikace, jenž chceme zobrazovat návštěvníkům. Tento editor je převážně využíván z toho důvodu, aby se uživatelé administrace nemuseli učit základy formátování HTML dokumentů. Díky tomuto editoru není tato znalost tedy nutná. Pouze stačí mít znalost základní práce s nějakým jednoduchým textovým editorem.

Tento editor můžeme zařadit do kategorie tzv. WYSIWYG[5] editorů. Jejich charakteristikou vlastností je, že vzhled, který vidí uživatel administrace se zobrazí i návštěvníkům.

Informace o tomto editoru můžeme nalézt na [10].

[5] What You See Is What You Get – Co vidiš, to dostaneš. Jedná se o formátování HTML dokumentů.

Kapitola 4

Analýza

4.1 Požadavky zadavatele

Cílem této práce není zabývat se vyjmenováním a specifikací všech požadavků, proto zde pouze pro představu zmíním pouze ty nejdůležitější, které si zadavatel této práce přál.

- Vytvořit on-line katalog zemědělské techniky
- Ukládat u strojů různý počet parametrů
- Podpora několika jazykových mutací
- Efektivní editace a vkládání nových strojů do katalogu
- Správa uživatelů, možnost přidělování přístupových práv
- Vkládání reklamních bannerů pro pozdější prodej reklamy
- Automatická úpravu velikosti fotografií nahrávaných do katalogu
- Možnost nahrávání více fotografií najednou
- Možnost vyhledávání strojů
- Srovnávání parametrů stroje – parking[1]
- Vytváření novinek a jejich zobrazování na úvodní straně
- Zasílání informačních zpráv registrovaným návštěvníkům na principu Email-info

Mezi cílové uživatele katalogu bude patřit široká veřejnost. Z tohoto důvodu je nutné se při tvorbě katalogu zabývat problémy kompatibility prohlížečů, což se jeví jako jeden z nejdůležitějších požadavků, aby nebyli potencionální návštěvníci odrazeni nefunkční a nevkusnou grafikou. Je očekáváno, že celá aplikace bude fungovat na serveru obsahující PHP s podporou mod_rewrite a databázi MySQL .

[1] Jedná se o vkládání strojů na virtuální parkoviště, které nám umožní zvolené stroje přímo porovnávat pomocí jejich parametrů. Podobné provedení je možné vidět na portálu www.mobilmania.cz k porovnávání mobilů.

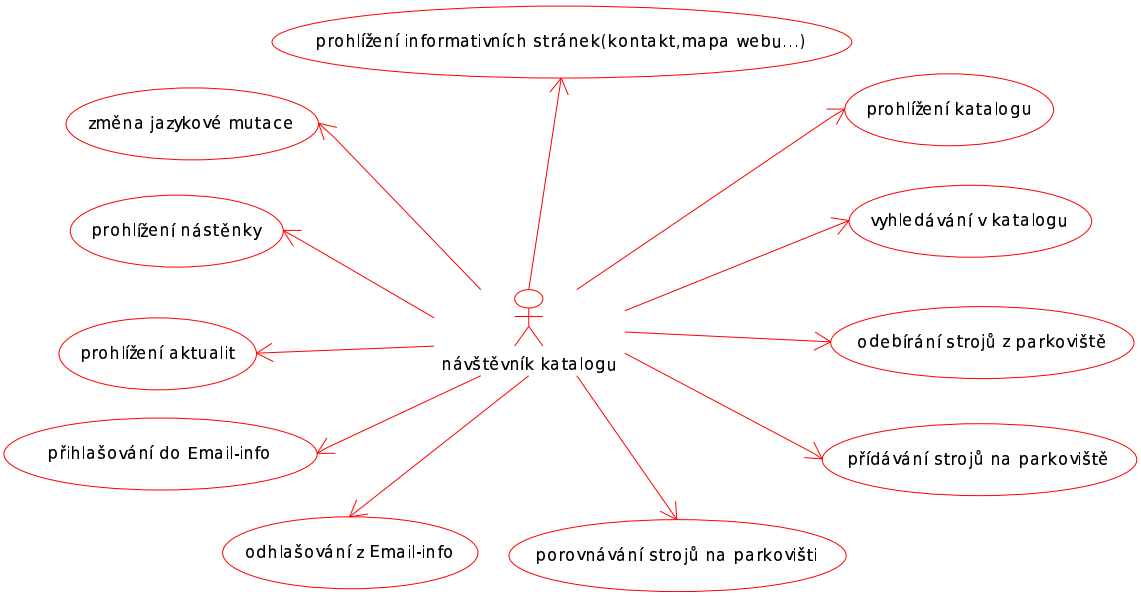
4.2 Návrh

4.2.1 Případy užití - USE CASE

V diagramu případů užití pro administraci na obrázku 4.1 můžeme vidět všechny aktéry administrace, kteří mohou přistupovat k jejím funkcím. V následujícím textu se dočteme a také to můžeme na obrázku vidět, že jedním typem uživatelů jsou administrátoři a druhým typem uživatelé s přístupovými právy. Jelikož je celý systém na principu přístupových práv postaven, je nutné, jak je zobrazeno na obrázku 4.1, tato přístupová práva kontrolovat.

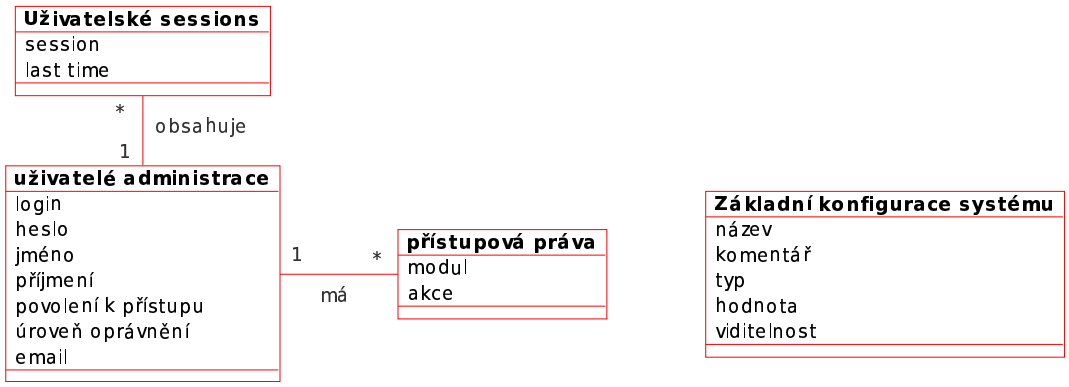
Obrázek 4.1: Diagram případu užití pro administraci

Na obrázku 4.2 si můžeme v kostce prohlédnout dostupné funkce, které má každý návštěvník katalogu. Tyto funkce jsou z hlediska pohledu na celý systém pouze informačního charakteru, návštěvník nemá možnost do jeho obsahu zasahovat. Jedinou možností zápisu a mazání je přihlašování a odhlašování z odběru Email-info.

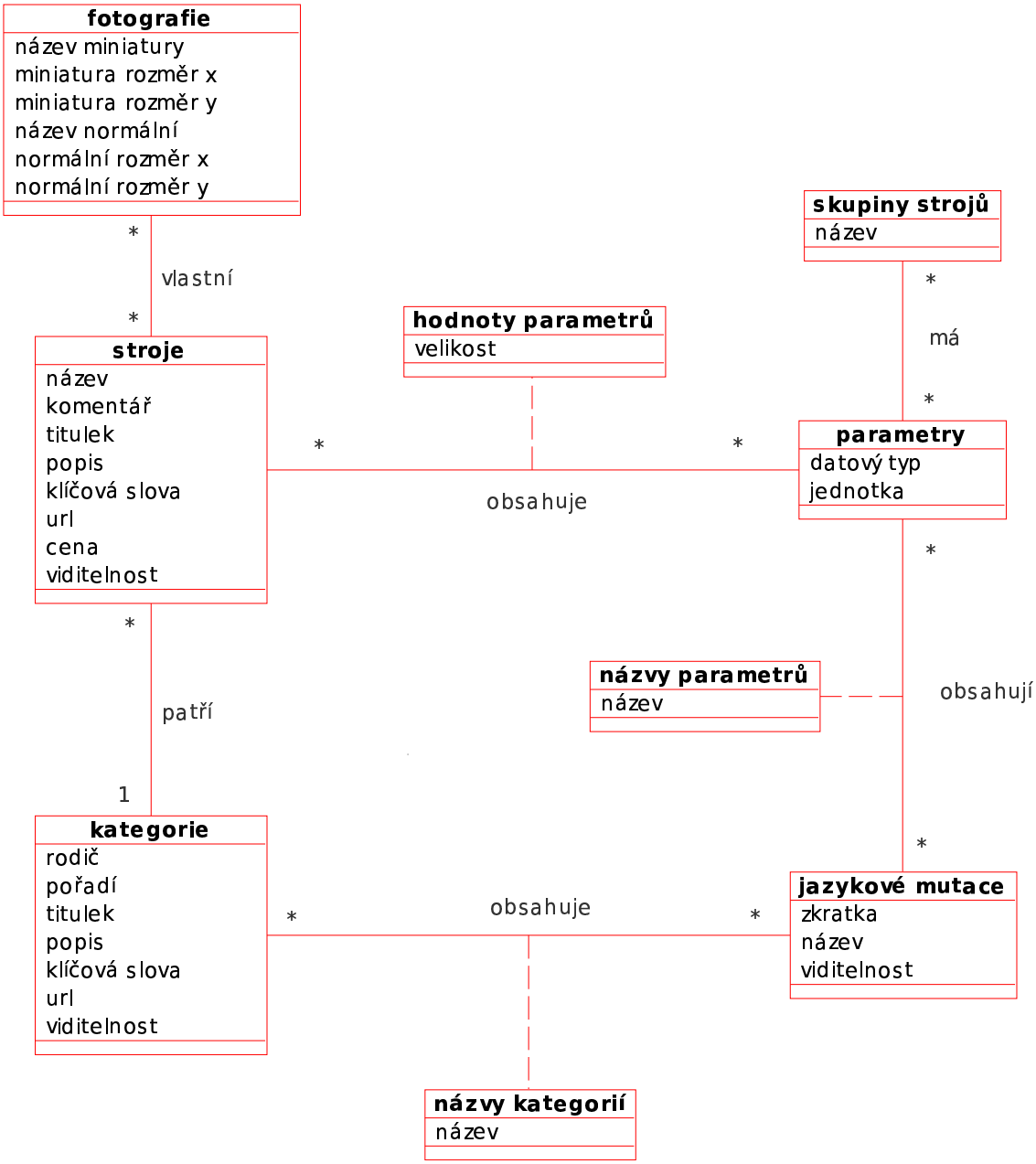


Obrázek 4.2: Diagram případu užití z pohledu návštěvníka

4.2.2 Konceptuální návrh katalogu – ER diagram



Obrázek 4.3: ER diagram administrace



Obrázek 4.4: ER diagram pro katalog strojů

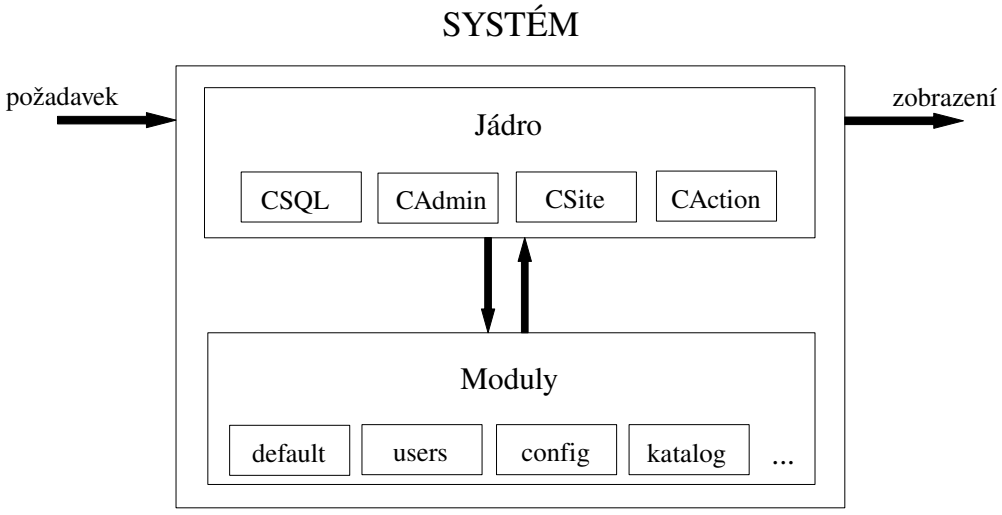
Kapitola 5

Implementace administrace

5.1 Architektura

Administrace je jednou z nejdůležitějších částí každého dynamického webu, u kterého máme v úmyslu zasahovat do jeho obsahu. Při tvorbě administrace bychom měli věnovat nemalý čas návrhu. Pokud je kvalitní návrh, programování složitějších funkcí v takové aplikaci se stává jednodušší.

Nyní si blíže popíšeme schéma návrhu implementovaného systému podle obrázku 5.1.



Obrázek 5.1: Schéma systému

Celý systém můžeme zjednodušeně chápat jako krabici, do které na vstupu vložíme určitý požadavek, ten se v systému podle jistých pravidel zpracuje a jeho výsledek, pokud bude mít grafický charakter, se zobrazí. Z toho vyplývá, že nás nezajímá samostatná implementace.

Pokud budeme chtít zjistit bližší informace o systému, můžeme jej rozdělit na dvě odlišné funkční části. První z nich je *jádro* systému, jež tvoří jeho základní kámen. Tato část je při rozšiřování systému neustále stejná. Vyskytují se zde základní třídy, které obsluhují funkčnost a možnou rozšířitelnost.

Do druhé části můžeme zařadit rozšiřující *moduly* systému. Jedná se o moduly rozšiřující funkčnost jádra systému. Jak můžeme na obrázku 5.1 vidět, jádro požaduje po určitém modulu provedení nějaké akce. Modul požadovanou akci zpracuje a výsledek se pak jádru předá.

5.2 Jádro

Při návrhu systému bylo nutné potýkat se s problémem oddělení samotného jádra systému, jež by obsluhovalo pouze funkční část administrace, a modulů, které se ke stávající administraci svou funkcí pouze připojují a doplňují její schopnosti.

K modularitě administrace bylo přistoupeno z důvodu možnosti systém bez velkých režijních zásahů dále rozšiřovat.

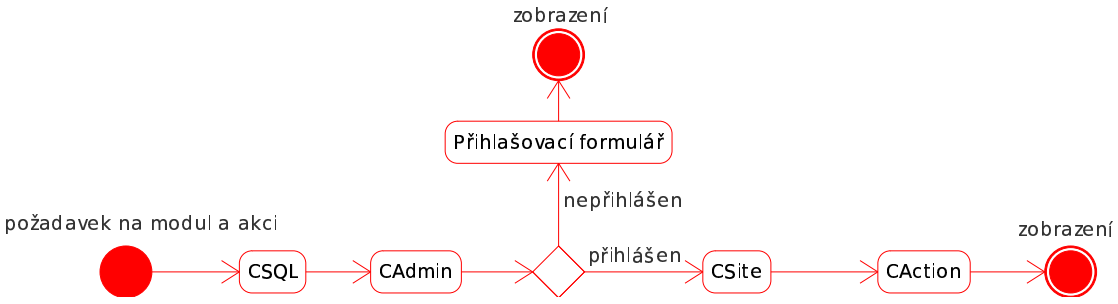
Funkčnost jádra administrace je postavena na výskytu několika tříd, přičemž každá zabezpečuje určitou funkční část z celé aplikace. Jako první musíme zmínit třídu na zpracování SQL dotazů, která kromě tohoto provádí také připojení k databázi. Bez takového připojení je celý systém nefunkční. Třída se jmenuje *CSQL* a najdeme ji v souboru *sql.class.php*.

Po úspěšném připojení k databázi je volán konstruktor třídy zabezpečující administraci webu. Jedná se o třídu *CAdmin* umístěnou v souboru *admin.class.php*. Tato třída kontroluje přihlášení. Pokud uživatel není přihlášen, je vyvolán přihlašovací formulář a skript je ukončen. V opačném případě se dále pokračuje ve vykonávání PHP skriptu.

Pro grafické zobrazení webu se používá třída *CSite* vyskytující se v souboru *site.class.php*. Ta obsahuje několik proměnných, ze kterých se po zavolání metody *show* sestaví výsledná webová stránka. Nejdůležitější proměnnou, kterou budeme při své činnosti hojně využívat, je proměnná *body*. Ukládají se zde zpravidla proměnné informace. Části systému, které se v závislosti na akci jednotlivých modulů mění. Do této části nepatří věci jako je hlavička stránky, menu a podobně. Více informací o této problematice se objeví v části 5.2.3.

V okamžiku, kdy je konstruktorem vytvořena třída *CSite*, můžeme přistoupit ke zpracování požadované akce. Zpravidla jsme schopni ji přečíst z adresy. Administrace rozlišuje celkem tři typy základních akcí. Podle jejího typu se pak také patřičně provádí náš požadavek. Třída obsluhující akce se jmenuje *CAction* a je umístěna v souboru *action.class.php*. Jedná se o třídu, bez které by systém nebyl schopen požadované akce zpracovávat.

Zpracovávání požadavku si můžeme znovu ujasnit na obrázku 5.2.



Obrázek 5.2: Zpracování požadavku

5.2.1 třída CSQL

Tato specifická třída nachází uplatnění jak na straně administrace, tak na návštěvnické straně. Je to proto, že v obou případech je nutné se připojovat k databázi a zpracovávat nad ní SQL dotazy. Výsledky těchto dotazů jsou obvykle data, bez jejichž přítomnosti nemá zobrazování webu smysl. Z tohoto důvodu se dospělo k názoru, že je nutné, aby se v okamžiku, kdy neexistuje spojení k databázi, přestaly zpracovávat další části skriptu a došlo k explicitnímu ukončení s vypsáním chybové hlášky.

Tato třída obsahuje metody, které nám zastřešují nejpoužívanější prováděné akce nad databází a podle nich vrací příslušné hodnoty. Názvy metod jsou podobné klasickým příkazům. Díky tomu, že je konstruktor této třídy obvykle volán na globální úrovni skriptu, je možné tuto třídu s výhodou připojovat a volat z dalších vytvářených funkcí za podmínky splnění odpovídajících volání globálních proměnných.

Po zavolání kontruktoru této třídy se provádí připojení k databázi podle definovaných parametrů. Pro informaci je dobré zmínit, že v této třídě existuje proměnná, která je schopná evidovat různé připojení k databázi. To znamená, že si můžeme předem nadefinovat více dostupných rozdílných variant tohoto připojení. Této vlastnosti můžeme s výhodou využít při překopírovávání webové aplikace z *localhostu* na existující server, přičemž při neustálém kopírování souborů na server a zpět nemusíme pořád kontrolovat a přepisovat připojovací parametry. Obecně si můžeme vytvořit libovolné množství takových variant.

Jednou z nejdůležitějších metod této třídy je *query*, která se chová stejně jako příkaz *mysql_query*. Výhoda tohoto řešení je v tom, že můžeme centrálně spravovat všechny požadavky, tudíž se vyvarujeme případného částečného ošetřování chyb, které bývá při nepoužití centrální správy častým problémem a vznikem chyb.

Za zmínku stojí také metoda *replaceData*. Ta umí sama rozpoznat, zda se jedná o vkládání nových dat do databáze a nebo pouze o opravu stávajících. Na vstupu očekává pole názvů sloupců, pole hodnot, které odpovídají názvům sloupců, název tabulky, nad kterou se bude provádět případný zápis nebo oprava dat, dále název sloupce, který je primárním klíčem a nakonec ještě hodnotu primárního klíče. Pokud je primární klíč roven nule, dojde automaticky k rozpoznání, že se jedná o insert do tabulky. V opačném případě se bude provádět update dat v tabulce.

5.2.2 třída CAdmin

Aby byla administrace katalogu přístupná pouze pro registrované uživatele, je nutné, aby byla zabezpečená. Za tímto účelem tudíž existuje tato třída, která nám bude řešit tento zabezpečený přístup k administraci webu. Tato třída se jmenuje *CAdmin* a je uložena v souboru *admin.class.php*. Tvoří samostatný funkční celek administrace. Pokud by použita nebyla, administrace webu by se stala nezabezpečená, tudíž by byla volně přístupná.

Klasické chování této třídy v okamžiku, kdy není uživatel přihlášen, je takové, že se zobrazuje přihlašovací formulář. Z toho můžeme sami usoudit, že všechny následující akce probíhající po zpracování této třídy jsou potlačeny a PHP skript je ukončen. V opačném případě, kdy je uživatel přihlášen, se ověřuje existence aktuální relace a její časové razítko. Každá relace přihlášeného uživatele je evidována také v databázi dostupných relací. Díky tomuto způsobu můžeme evidovat aktuálně přihlášené uživatele.

V okamžiku, kdy dochází k přihlašování, jsou údaje odeslané z přihlašovacího formuláře kontrolovány s daty v databázové tabulce dostupných uživatelů administrace. Kromě provedené kontroly těchto přihlašovacích údajů se kontroluje také povolení k přístupu. Jedná se

o to, že i když uživatel zná správné přihlašovací údaje, ještě ho to nemusí opravňovat k bezproblémovému přístupu do administrace webu. Příznak tohoto povolení musí být povolen v databázi u příslušného uživatelského účtu. Bez tohoto povolení se odeslání přihlašovacích parametrů chová stejně jako kdyby uživatel administrace webu vůbec neexistoval.

Díky tomu, že tato třída existuje jako samostatná část, můžeme ji libovolně upravovat, aniž by to mělo nějaký vážnější dopad na samostatnou funkční část webu. Realizaci tohoto modulu můžeme vždy upravit podle aktuálních požadavků. Další rozšíření, které by se dalo také realizovat v závislosti na požadavcích na administraci, je povolit přihlášení pouze jedné instance od jednoho uživatelského účtu.

5.2.3 třída CSite

Jelikož se při návrhu objevil požadavek nějakým způsobem řešit zobrazování administrace webu, vznikla za tímto účelem tato třída. Jejím úkolem je tedy zobrazovat kompletní grafický výstup webu, který je viditelný pro uživatel administrace webu. Jelikož je celý systém navržen tak, že je možné podle jednotlivých modulů přiřazovat uživatelská práva jednotlivým uživatelům administrace, je nutné, aby při volání konstruktoru této třídy byl vložen parametr obsahují identifikaci uživatele. Je to z toho důvodu, abychom si tento parametr nemuseli dodatečně zjišťovat z třídy řešící administraci, která pokud bude někdy upravována, nemusí již tento parametr mít v sobě uložen. Tímto způsobem zajistíme také nezávislost této části na jiné části skriptu a to nám také při ladění dovolí snáze hledat případné chyby.

Jak již bylo zmíněno, tato třída se nám stará o zobrazení požadovaného grafického výstupu. K provedení zobrazení se využívá metoda *show*. Aby však všechno fungovalo tak jak má, je nutné, abychom veškeré věci, které budeme chtít zobrazit v příslušnou chvíli na příslušném místě, tiskli (ukládali) také do příslušných proměnných, které nám tato třída nabízí. Bez dodržení těchto pravidel se pak nemůžeme divit výsledku.

Celkem můžeme provádět tisk do čtyř různých proměnných, které by se nám měly postarat o kompletní potřebný rozsah. V následujícím textu se o těchto proměnných dozvíme více infomací.

První proměnná třídy *CSite*, do které je možné provádět tisk, je *header*. Umožní nám přistupovat do části HEAD v HTML kódu. Hlavní výhodou této hlavičky je fakt, že můžeme dodatečně ovlivňovat její podobu a také třeba jen pro jednu unikátní zobrazovanou stránku webu můžeme specifikovat jednoznačnou akci v podobě nějakého Javascript kódu.

Další funkční proměnnou, do které můžeme ukládat, je *head*. V této proměnné jsou uložena data, která se budou vkládat bezprostředně po tagu BODY v HTML kódu. Podle pojmenování můžeme tuto proměnnou chápat jako jakousi hlavičku celého zobrazovaného dokumentu. V našem případě se do této proměnné bude obvykle ukládat modulové menu a menu příslušející zvolenému modulu.

Jako u většiny dynamicky tvořených webů, tak i tady musíme mít část (v našem případě tedy proměnnou), která bude téměř skoro vždy stejná a pak část, jež se nám bude měnit v závislosti na modulu a akci v daném okamžiku požadovaných. Tato část bude reprezentována proměnnou *body*. Obsah této proměnné bude umístěn bezprostředně po zmiňované proměnné *head*. Teď je tedy jasné, že tato proměnná nás bude z hlediska změn obsahu webu zajímat. Jelikož se ale o zpracování webu stará jádro systému samo, nebudeme muset u každé funkce, která se nám bude starat o grafický výstup, vynucovat zápis do této proměnné. Bude pouze stačit, když v návratové hodnotě dané funkce vrátíme obsah, který budeme chtít zobrazit.

Poslední proměnnou, která nám chybí k tomu, abychom měli grafický obsah webu kom-

pletní, je *foot*. U této proměnné se ve svém obsahu vyskytují převážně uzavírací tagy náležící k hlavičce stránky, aby vše bylo korektní. Kromě toho se zde obvykle vyskytuje ještě patička informující o datu vytvoření a autorovi.

Zmíněné proměnné, ze kterých je celý web složen, a které ovlivňují také jeho obsah, zde byly jmenovány pro případ, že bychom chtěli někdy explicitně změnit kteroukoliv část webu.

O této třídě bychom měli také říci, že nám řeší modularitu systému. Při zavolání konstruktoru na tuto třídu dojde k načtení dostupných modulů a jejich konfigurací. Počítá se s tím, že každý modul obsahuje v konfiguračním souboru jak povinné konfigurační parametry, tak volitelné parametry, které se týkají pouze modulu samotného.

5.2.4 třída CAction

Jelikož ještě nebylo zmíněno kde a jakým způsobem budou zpracovávány jednotlivé akce, dozvíme se o tom blíže v této části textu, protože třída *CAction* nám tento problém řeší.

Tato třída byla vytvořena za účelem centrální správy všech prováděných akcí. Třída jako samotná pouze provádí kontrolu existence prováděné akce v závislosti na modulu a následně její volání. Všechny dostupné akce pro zadaný modul jsou specifikovány v konfiguračním souboru příslušného modulu. V tomto konfiguračním souboru můžeme kromě různých lokálních proměnných nalézt také celkem tři typy zpracovávaných akcí.

Prvním typem je P – *pages*, který se nám stará o stránky zobrazované na výstupu. Výsledkem tohoto typu akce je určitý grafický výstup, který se vkládá do funkční části webu. Konktrétně do již zmiňované proměnné *body* v třídě *CSite*. Z toho vyplývá, že z funkčního hlediska se po volání tohoto typu neprovede žádná akce, která by nám jakýmkoliv způsobem měla ovlivnit spravovaná data. Má pouze informativní charakter. Pokud není pro zvolený modul nastaven žádný typ akce, volí se automaticky základní stránka z konfiguračního souboru náležícího danému modulu.

Druhým typem je akce I – *interAction*. Je prováděna jako interní. Výsledkem volání tohoto typu není žádný grafický výstup. Obvykle provádí operace nad daty v databázi. V celém procesu zpracování akcí a jejich příkazů je tento typ volán ještě před zpracováváním akce typu P. Z toho vyplývá, že po provedení tohoto typu akce můžeme provést ještě úpravu volání akce typu P, která pokud není nastavena, volá se základní stránka pro grafický výstup aktivního modulu.

Posledním typem zpracovávaných akcí je akce typu B – *backgroundAction*. Jak lze z názvu vypozorovat, jedná se o akci, která se bude provádět na pozadí. Jelikož je dbáno o centrální správu akcí, bylo nutné, při používání techniky AJAX, tento typ akce vytvořit. Účelem této akce je zpracovávat požadavky objektu XMLHttpRequest. Výhodou tohoto zpracovávání je, že nemusíme kompletně obnovovat celé stránky, ale pouze modifikujeme objektový model stránek. Z tohoto důvodu také očekáváme na výstupu akce tohoto typu odpověď, která nám vrátí jen určitou část webu, kterou díky modifikace objektového modelu zakomponujeme do již existujícího modelu. Po ukončení zpracování akce tohoto typu je volán příkaz *exit*, který nám zabezpečí ukončení probíhajícího cyklu a také to, aby nedošlo k volání metody *show* třídy *CSite*.

Je nutné ještě říci, že zmiňované akce jsou předávány prostřednictvím URL metodou *GET* s příslušným typem akce a také hodnotou.

5.3 Modularita sytému

5.3.1 Vlastnost modularity

Aby byl tento systém co nejvíce universální, bylo rozhodnuto, že se vytvoří jako systém s modulární administrací. Pojem modularity můžeme chápat jako schopnost přidávání a odebírání funkcí systému bez náročného zásahu do jeho jádra. Také to můžeme chápat jako určitou okamžitou rozšiřitelnost již existujícího systému.

U systému se s výhodou využívá toho, že se skládá z jakéhosi jádra a na sobě funkčně nezávislých modulů, jejichž počet ani funkce nejsou omezeny. Vlastností modulů je jejich plná funkční nezávislost. To znamená, že tvorba modulů může probíhat nezávisle za předpokladu dodržení určitých pravidel, která platí pro konfigurační soubory modulů.

Díky té skutečnosti, že se v systému moduly chovají jako nezávislé části, je možné jejich vývoj rozdělit mezi více lidí s tím, že výsledek jejich práce bude vždycky použitelný a funkční s ostatními třeba již existujícími moduly. Z tohoto pohledu se jedná o velmi výhodnou vlastnost, která nám může mnohonásobně ovlivnit rychlost tvorby možných rozsáhlejších projektů, jelikož dostáváme možnost rozdělit projekt na další logicky nezávislé části.

5.3.2 Pravidla pro tvorbu modulů

Abychom dosáhli funkční modularity, bylo nutné nadefinovat několik pravidel, která je nutné pro správnou funkčnost modulů dodržovat. S takovými předem nadefinovanými pravidly se můžeme setkat u všech podobně založených systému. Bez nich by systém nemohl mít takové vlastnosti, které jsou pro něj v současné době charakteristické.

5.3.3 Tvorba modulů

O přidání modulu je základní povinností programátora o této změně informovat systém. Tato skutečnost se systému sdělí prostřednictvím základního konfiguračním souboru *config.php*, který se nachází v adresáří */etc/* v kořenové složce celého webu.

Úplně první úkol, který nás však potká, je zvolit vhodný název nového modulu. Tento název musí být v seznamu všech dostupných modulů unikátní. Pokud není unikátnost zaručena, není zaručen ani správný chod celého systému.

V konfiguračním souboru existuje globální proměnná definující všechny dostupné moduly systému. Pokud chceme přidat nový modul, je naší povinností zapsat jeho název do seznamu používaných modulů.

Existující funkční moduly jsou pak uloženy v adresáři *modules*, který se nachází v kořenovém adresáři celého webu. Zde se můžeme také přesvědčit o tom, kolik existujících modulů může náš systém při své činnosti využívat. Názvy existujících složek musí s názvy modulů korespondovat.

V následujícím textu se budeme věnovat vlastnímu nastavení modulu tak, abychom dosáhli jeho funkčního chodu. Dalším pro nás důležitým souborem je konfigurační soubor modulu – *config.php*. Ten se musí vždycky vyskytovat v adresáři */etc/*, jenž je umístěn v samotném modulu. Existence tohoto souboru je kontrolována při načítání modulů systému.

Nyní se podíváme na parametry, které se musí vyskytovat v konfiguračním souboru, aby nám systém volal požadované typy akcí podle našeho přání. Globálních proměnné jsou v jakési kaskádě, která nejprve vychází z klíčového slova *modul*, pak je specifikován název modulu a následují názvy jednotlivých proměnných modulu. V každém modulu se vyskytují povinné a nepovinné proměnné. Přičemž povinné proměnné nám zaručují správný chod

modulu. Nepovinné proměnné jsou takové, jenž jsou globálně využívané v rámci modulu samostatného. Obvykle tady můžeme nalézt definice cest k obrázkům, jisté speciální číselné konstanty a tak podobně.

5.3.4 Povinné proměnné v konfiguračním souboru modulu

V této části textu se budeme zabývat povinnostmi definovat určité typy proměnných, které jsou například kontrolovány při činnosti modulu.

Každý modul musí mít definován svůj název. Je nutné, pokud budeme chtít někde využívat specifičtější pojmenování modulu, abychom nadefinovali také jeho textový popis, jenž se bude zobrazovat. Mnohdy totiž nemusí být název modulu dostatečně vypovídající o jeho činnosti.

K tomu, abychom mohli provádět volání různých typů akcí, je nutné tyto akce nejprve v jednotlivém modulu nadefinovat, protože před skutečným voláním určitého typu akce, které nám provádí třída *CAction*, se provádí kontrola této definice. Bez její existence se daná akce neprovede. Ta je pak brána jako neplatná. Z určitého hlediska to můžeme brát jako ošetření případných nevyžádaných stavů.

Každá akce, která bude v modulu nadefinována, musí obsahovat řetězec, podle kterého dojde k jednoznačnému určení volané akce. V okamžiku, kdy budeme mít akci jednoznačně pojmenovanou v rámci modulu, je nutné jí přiřadit funkci, která danou akci bude vykonávat. Jelikož ale těchto funkcí může být obecně nekonečně mnoho, řeší se to tím způsobem, že ke jménu funkce se vloží ještě název souboru, ve kterém se daná funkce bude vyskytovat. Vychází se z přepokladu, že soubory pro akce se vyskytují v adresáři *admin* daného modulu. Jiné adresáře pro zdrojové soubory akcí nejsou z hlediska bezpečnosti přípustné.

Jak bylo v předešlém textu zmíněno, existují celkem tři typy zpracovávaných akcí, které jsou z logického hlediska zcela typově odlišné a řeší nám podle potřeb různé požadavky. Podle domluvených konvencí se tyto akce značí P, I a B. Nebudeme znovu vyjmenovávat jejich význam, ale pouze zmíníme, že akci P je možné explicitně nastavit na určitou výchozí hodnotu. Jedná se o nastavení proměnné *dafaultPages*, která odkazuje na akci typu P. Pokud není pro daný modul nastavena tato akce, je použita zmiňovaná výchozí hodnota.

Dalším typem proměnných, které je možné uvést v konfiguračním souboru modulu, jsou přístupová práva. Tyto proměnné ovšem nepatří do skupiny povinně definovaných. Její výskyt je v rámci modulu volitelný. Význam použití se projeví v okamžiku, kdy chceme přidělovat oprávnění k různým akcím.

Pokud tato proměnná není definována a systém administrace používá přístupová práva, automaticky se uvažuje, že je tento modul bez uživatelských práv a je tudíž plně přístupný bez jakýchkoliv omezení.

5.4 Uživatelská práva

5.4.1 Význam uživatelských práv

Jelikož se jedná o systém s administrací, který bude mít schopnost umožnit práci nejednomu uživateli, bylo nutné přemýšlet nad tím, jak zajistit, aby do určitých částí systému měli přístup pouze povolení uživatelé. Za tímto účelem vznikl systém uživatelských práv, který má v našem systému dvě úrovně.

První úroveň je *administrátor*. Tato úroveň oprávnění umožňuje uživateli administrace webu povolit přístup ke všem dostupným akcím administrace. Charakteristikou tohoto

přístupu je kompletní neomezený přístup, který nemůže být žádným způsobem omezen. Maximální počet takových uživatelů administrace není definován.

Druhou úrovní oprávnění je *uživatel*. Úroveň a množství oprávnění u tohoto typu účtu může být různé. Jedná se o užitečnou vlastnost systému, kdy můžeme různým uživatelům rozdělit správu webu na jednotlivé funkční celky. Množství uživatelských práv se pro jednotlivé moduly odvíjí od toho, jak moc chceme mít daný modul na uživatelských právech zabezpečen.

5.4.2 Nastavování uživatelských práv

Aby funkce uživatelských práv splnila svůj účel, je nutné tuto vlastnost pro přihlášeného uživatele zakomponovat také do zpracovávaného skriptu. Kontrolu uživatelských práv nám provádí metoda *enableAction* patřící do třídy *CAction*.

Zmiňovaná metoda provádí kontrolu pro aktuálně přihlášeného uživatel vzhledem ke zvolenému modulu a akci, kterou máme v úmyslu vykonat. Jak už bylo zmíněno, existují celkem dvě úrovně oprávnění.

Pokud se tedy při kontrole uživatelských práv zjistí, že aktuálně přihlášený uživatel má úroveň oprávnění *administrátor*, další kontrola se již neprovádí a přístup k akci daného modulu je bez další kontroly okamžitě povolen.

Jestliže se při kontrole uživatelských práv naopak zjistí, že se jedná o úroveň oprávnění *uživatel*, proces kontroly práv pokračuje do další fáze. V další části kontroly nastává práce s vyhledáváním povolení k akci daného modulu v databázi. Seznam uživatelských práv pro úroveň oprávnění *uživatel* je evidován ve speciální databázové tabulce k tomuto účelu určené. Celý systém správy těchto uživatelských práv funguje tak, že uživatel, který má mít přístup k akci určitého modulu, musí mít pro svůj uživatelský účet o tomto modulu a akci v této speciální databázové tabulce záznam. Pokud se však nenajde výskyt takové akce k modulu systému, přístup k akci modulu není povolen. Z textu tedy vyplývá, že existence záznamu v této tabulce opravňuje uživatele k přístupu.

V současnosti už víme, jakým způsobem budou uživatelská práva ukládána, ale ještě nevíme, jak je můžeme nastavovat. K tomuto účelu nám slouží speciální modul *users*, který nám dovoluje kompletní nastavení uživatelského účtu. Kromě standardních informací můžeme pro uživatele také nastavovat úroveň oprávnění a pak případná uživatelská práva.

Seznam všech dostupných uživatelských práv se zjišťuje z konfiguračních souborů jednotlivých modulů. Tato práva jsou pak kontrolována s databází uživatelských práv a podle nastavení jsou jim přiděleny příslušné příznaky.

5.5 Modul KATALOG

5.5.1 Seznámení s problematikou

Pojem katalog je obvykle spojován s rozsáhlou databází určitých produktů, které z hlediska svého použití můžeme zařadit do různých kategorií. V případě tohoto systému, jak již bylo zmíněno, se bude jednat o bazarové zemědělské stroje, které budeme vkládat do katalogu. K tomu, abychom byli schopni tyto stroje do takového katalogu ukládat, je nutné provést několik režijních úkonů, které nám při vkládání pomohou ukládaný stroj přesně zařadit. Bližší informace o těchto úkonech nalezneme v následujícím textu, konkrétně v 5.5.2, 5.5.3, 5.5.4. Informace o vkládání stroje jsou pak popsány v 5.5.6.

5.5.2 Parametry stroje

Abychom byli schopni vždy jednoznačně identifikovat parametry stroje, bylo při řešení katalogu strojů přistoupeno k myšlence centrální správy parametrů. Tato myšlenka nám zjednoduší práci s parametry. Jedním z hlavních pozitiv tohoto řešení je skutečnost, že systém bude automaticky schopen k sobě řadit parametry různých strojů při jejich porovnávání.

Pokud by se takový problém neřešil tímto způsobem, ale u každého stroje separátně, jen prostřednictvím nějakého textového pole, nebyli bychom schopni vždy jednoznačně říci, že stroj, obsahující určité parametry, má shodné právě jen některé parametry se strojem jiným. Musíme si uvědomit, že tento problém se může objevit pokaždé, kdy máme možnost zapsat určitou věc více způsoby, což by v případě vždy unikátního textového zápisu parametru nebyl žádný problém. Proti takovému problému bychom pak museli zbytečně složitě bojovat.

Pro ukládání parametrů v databázi jsou vytvořeny celkem dvě tabulky. V první tabulce jsou ukládány názvy parametrů v různých jazycích, přičemž primárním klíčem do této tabulky je číselná identifikace parametru a jazyková verze.

Druhá tabulka obsahuje sloupce, které jsou společné pro všechny jazykové verze, jako je například jednotka parametru.

Propojení parametrů se strojem se děje prostřednictvím speciální tabulky, ve které jsou ukládány hodnoty. Primárním klíčem do této tabulky je číselná identifikace parametru a číselná identifikace příslušného stroje.

5.5.3 Skupiny strojů

Aby se usnadnila a zefektivnila práce se spoustou různých parametrů, které je možné vkládat ke strojům, bylo přistoupeno k řešení vytváření *skupin strojů*. Jedná se o interní pojem administrace. Díky skupinám strojů docílíme efektivnějšího zadávání parametrů.

Celý význam této funkce je takový, že si mohu pod jednotlivé skupiny strojů nadefinovat jejich nejpoužívanější parametry. Výhoda se projeví v okamžiku, kdy v systému existuje několik desítek parametrů a budu chtít vložit stroj obsahující například deset různých parametrů. Tyto parametry nemusím tedy složitě hledat a postupně přidávat. Pouze stačí, abych si při vkládání stroje zvolil příslušnou skupinu strojů, do které bych mohl stroj zařadit. V tom okamžiku se mi zobrazí parametry definované pro danou skupinu. Tím pádem odpadá čas strávený nad vyhledáváním a přemýšlením, jaké parametry se pro daný stroj vkládají.

5.5.4 Kategorie strojů

V předešlém textu bylo zmíněno, že každý katalog by měl mít určité členění do logických částí. V případě katalogu zemědělské techniky je tento problém pojmenován jako kategorie. Toto členění nám umožní jednoznačně logicky stroj v rámci katalogu umístit. Od strojů v kategorii obvykle očekáváme, shodu paramatrů. Toto tvrzení však nemusí být pravidlem, protože ne všichni výrobci udávají právě takové parametry, které se rozhodneme ukládat.

Kategorie patří do skupiny informací ukládaných u strojů, kde je nutné uvažovat s možným dalším jazykovým rozšířením. Z tohoto důvodu bylo třeba se tak zachovat vzhledem k ukládaným informacím o kategorii. Převážně se jedná o ty názvy kategorií, které je nutné mít přeložené ve všech definovaných jazykových verzích.

Podobně jako u parametrů strojů, i tady se tento problém ukládání kategorií v různých jazykových mutacích řeší vytvořením speciální tabulky, kde jsou specifikovány pouze názvy kategorií v jednotlivých jazycích. Podobně jako u parametrů, je klíčem do této tabulky

číselný identifikátor kategorie a označení jazykové verze. V hlavní tabulce příslušející kategoriím pak můžeme najít například identifikaci předka kategorie nebo pořadí v kategorii. V dalších sloupcích této tabulky jsou informace vztahující se k optimalizaci webu pro vyhladávače a s tím spojenému přepisování webových adres.

Pojem *předek kategorie* nebo také někdy označováno jako otec kategorie, je číselný identifikátor specifikující nadřazenou kategorii, ke které námi zpracovávaná kategorie náleží. Přičemž jedna kategorie může mít několik dceřinných kategorií. Pro každou úroveň je možné dále specifikovat pořadí na zpracovávané úrovni.

Ze zmíněného textu si musíme uvědomit, že díky ukládání hodnoty otcovské kategorie a také díky pořadí na dceřinné úrovni, jsme schopni vytvořit neomezenou hierarchii kategorií s vložením nové kategorie na libovolné místo.

Aby byly informace o kategorii kompletní, je nutné ještě říci, jak je řešena kořenová úroveň takového hierarchického stromu kategorií. Vychází se z předpokladu, že kořenem kategorií je kategorie s číselným identifikátorem nula. Od tohoto idenfikátoru se pak dále vytváří strom kategorií.

5.5.5 Obrázky strojů

Protože obrázky jsou schopné říci o stroji mnohem víc než nějaký obsáhlý text, je i tady možnost u vkládaných strojů obrázky ukládat. Byl to jeden z požadavků zadavatele. Dalším a to stěžejním požadavkem, který z neznámých důvodů nebyl implementován v konkurenčním systému, byla automatická úprava velikosti vkládaných obrázků. Z hlediska komfortu při uploadu, jsem přistoupil k dvojímu řešení uploadu. Jedním je multiupload fotografií pomocí knihovny *swfupload*, o níž bylo více napsáno v kapitole 3.11. Jelikož je podmínkou instalace *Flash Playeru*, existuje zde pro ty, kteří ho nemají nainstalován, ještě možnost klasického uploadu – po jedné fotografii.

Veškeré operace týkající se uploadu obrázků probíhají na pozadí, to znamená, že v okamžiku, kdy jsou obrázky uploadovány, nedochází k obnovení stránky. S výhodou se zde využívá objektového modelu stránky pro okamžité zobrazení již úspěšně uploadovaných fotografií.

5.5.6 Stroje

V následujícím textu bude zmíněno řešení ukládání strojů v katalogu. Informace zde publikované byly probrány se zadavatelem systému, bude snahou zmínit pozitiva a nebo případná negativa takových rozhodnutí.

Klíčovou informací o stroji, kterou vždy ukládáme, je její název. Skoro pokaždé se jedná o text, na základě kterého jsme schopni již odhadovat o jaký stroj, nebo obecně produkt, jde. Asi by se očekávalo, že je nutné název stroje ukládat v různých jazycích. Dle zadavatele není třeba tuto problematiku řešit po vzoru kategorií či parametrů, protože názvy strojů jsou od výrobců voleny tak, že jsou multilinguální, tudíž není nutné jejich názvy ukládat v duplicitách pro jednotlivé jazyky. Jednalo by se o zbytečnou redundanci.

Aby bylo možné stroj v katalogu najít, je další ukládanou informací u stroje jeho kategorie. Ta nám jednoznačně definuje umístění stroje v rámci katalogu. Bližší informace o ukládání strojů v rámci kategorií můžeme nalézt v kapitole 5.5.4.

Za účelem možnosti textově okomentovat vkládaný stroj, můžeme ke stroji také vložit komentář. Význam a ukládání tohoto komentáře vychází pro tento katalog ze vzoru katalogů zahraničních. Komentář je zadáván pouze v jednom a to libovolném jazyce. Počítá se s tím, že důležité informace o strojích se budou vyskytovat převážně v parametrech strojů. Ty

překlad v dostupných jazycích mít budou. Bylo mi řečeno, že se ve velké většině případů nebudou vkládat rozsáhlé texty. Řešení zobrazení této informace nemusí být vždy z pohledu zahraničního návštěvníka nejlepší. Pokud však budeme brát v úvahu informaci, že tento komentář bude vyplněn zřídkakdy, můžeme se s tímto řešením spokojit. Musíme si uvědomit, že se jedná o zjednodušení obsluhy katalogu.

Jelikož se čas od času objeví požadavek skrýt stroj s rámci katalogu, existuje zde příznak pro toto chování. Jedná se o to, že pokud stroj nechceme přímo mazat, můžeme jej umístit do mezi stavu – tedy ho skrýt.

Standardní informací ukládanou u stroje je jeho cena bez DPH. Bylo domluveno, že se bude zadávat pouze jedna cena v korunách českých s tím, že se by se v budoucnu případné další ceny pro jiné měny dopočítávali na základě přímo zadaného kurzu nebo prostřednictvím kurzovního lístku ČNB.

K nejstěžejnějším informacím, jež se u stroje ukládají, patří jeho parametry. Při zadávání parametrů se s výhodou využívá skupin strojů, jak bylo zmíněno v kapitole 5.5.3. Přestože se nám pro danou skupinu zobrazí předdefinové parametry, můžeme jejich výčet upravit. A to buď přidáním nového nebo smazáním stávajícího. Počet parametrů si tedy můžeme upravit podle svých potřeb.

K prvotním nejvíce vypovídajícím informacím o stroji patří jeho fotografie. Možnosti uploadu přikládaných fotografií ke strojům byly zmíněny v kapitole 5.5.5. K těmto informacím už jen zbývá dodat, že díky objektovému přístupu můžeme takové obrázky ze seznamu také mazat a tím rušit příslušnost obrázku ke stroji.

K posledním ukládaným informacím u stroje patří informace pro optimalizaci. Tyto informace jsou využívány při zobrazování detailu stroje v katalogu.

Kapitola 6

Implementace prezentační části

6.1 Seznámení

V následující části textu bude zmíněna implementace systému katalogu z prezentačního pohledu. To znamená, že se zde rozebere zpracování katalogu z pohledu, jenž vidí jeho běžný navštěvník. Rozmístění jednotlivých funkčních částí systému v rámci stránky můžeme vidět na obrázku 6.2.

Prezentační část systému je po grafické stránce zpracovávána pomocí šablonovacího systému *SMARTY*, o kterém bylo více zmíněno v kapitole 3.10. Jedná se o to, že z hlediska přehlednosti bylo třeba oddělit šablonu webu od programové části tohoto webového systému.

6.2 Jazykové mutace

Celý systém byl navržen s tou myšlenkou, že v budoucnu bude počítáno s rozšiřováním jazykových mutací. Aby bylo možné tímto způsobem korektně zobrazovat prezentační část systému, je nutné, abychom si nadefinovali texty, které budou neměnné, a texty, které bude možné měnit. Mezi neměnné patří například texty obsažené v menu a všechny další režijní texty, které jsou součástí jakéhosi jádra prezentační části systému. Všechny tyto texty jsou umístěny ve speciálním překladovém souboru. Pokud budeme tedy chtít přidat nový jazyk, je nutné vytvořit překlad těchto textů pro jazyk nový. O překlad neměnných (statických) textů pro příslušný jazyk se stará funkce *translate*.

Ostatní texty vztahující se ke kategoriím či parametrům jsou obyčejně již předdefinovány v rámci administrace, tudíž není nutné se touto problematikou při vytváření nového jazyka dále zabývat. Jen musíme mít jistotu, že výraz se v administraci pro nový jazyk vyskytuje.

6.3 Grafická šablona

Při vytváření systému bylo cílem, aby se nezměnila, ale případně ještě zlepšila, pozice ve vyhledávačích. Z pohledu zdrojových kódů současného webu jsem se rozhodl šablonu celého webu plně přepracovat po vzoru mně známých doporučení.

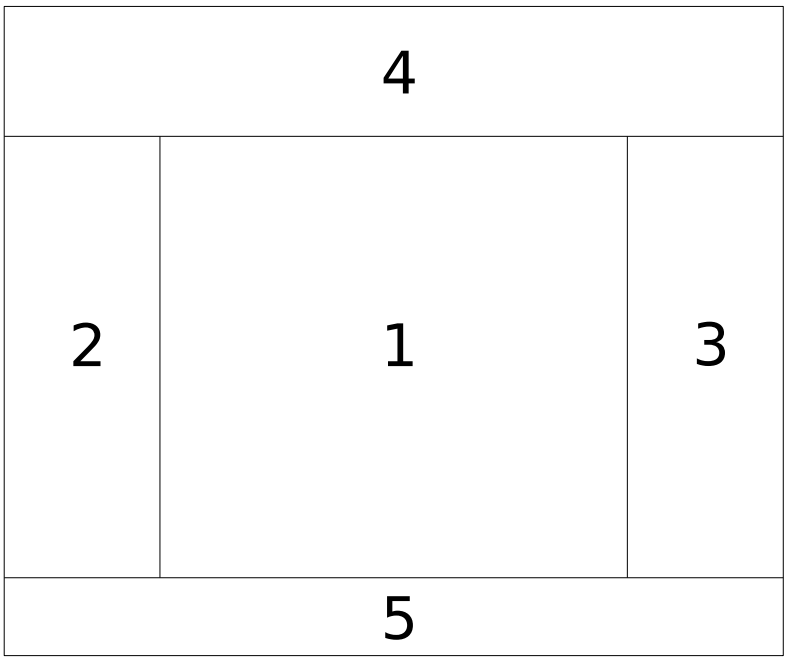
Jedním z hlavních úkolů bylo, aby se text, který plní informační část stránky, vyskytoval ve zdrojovém kódu na prvním místě ihned po nadpisu. To znamená, že je nutné všechny režijní texty, kam patří případná menu, hlavička či patička, vyskytovaly až po hlavním textu. Mezi další věci, které jsem se snažil dodržet, byla i schopnost mít na každé straně

rozdílný nadpis typu H1, což na současném webu zadavetele nebylo dodrženo. Vyskytoval se zde neustále stejný text uvozující název webu. V neposlední řadě bylo dbáno v rámci webu také na pořadí nadpisů, aby se nestala ta věc, že po nadpisu typu H1 se bude vyskytovat nadpis typu H3.

Abychom zvýšili relevanci hlavních nadpisů, je jeho text obsažen také v titulku stránky. Kromě toho se část z tohoto nadpisu vyskytuje také v adresách webu. O této problematice se více píše v kapitole 6.10.

Grafického vzhledu webu do jeho konečné podoby bylo dosaženo díky kaskádovým stylům, o nichž bylo psáno v kapitole 3.2. Návrhy takových stránek se zmíněným rozmístěním prvků stránky můžeme najít na internetu pod pojmem *SEO layout*. Jedná se o šablony navržené pro lepší optimalizaci webu. Bližší informace lze nalézt v [11].

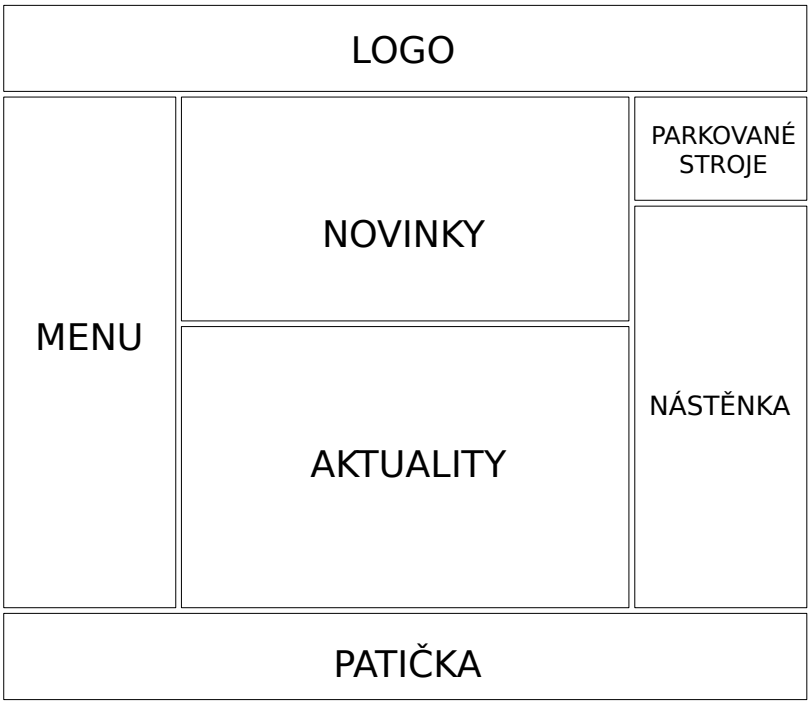
Pořadí jednotlivých částí webu v rámci stránky je vidět na obrázku 6.1



Obrázek 6.1: Pořadí částí stránky v rámci zdrojového kódu

6.4 Novinky

Část Novinky tvoří automaticky generovanou část stránek. Její zobrazení se provádí pouze na úvodní stránce celého webu. Hlavním účelem této části jsou informace o nových strojích uložených v katalogu. Katalog je v současné době nastaven na výpis posledních čtyř vložených strojů.



Obrázek 6.2: Rozmístění jednotlivých funkčních částí systému na stránce

6.5 Aktuality

Mezi požadavky zadavatele patřila možnost informovat návštěvníky webu o případných akcích či změnách prostřednictvím zpráv, které jsou označovány jako *Aktuality*. Tyto zprávy jsou přístupné v rámci úvodních stránek webu.

Jednotlivé zprávy jsou kromě nadpisu uvozeny krátkým textem a případným obrázkem. Při prokliku pak dojde k zobrazení detailu této zprávy, kde se nám ukáže kompletní text s příslušnou galerií.

6.6 Nástěnka

Název této části webu může být různě interpretován. V případě vytvářeného katalogu se jedná o část zobrazované stránky, která obsahuje krátké informativní zprávy, případně reklamní bannery. V podstatě se jedná o podobnou část jako jsou aktuality, ale s tím rozdílem, že mají podstatně kratší informativní charakter. Každý příspěvek je zobrazen ve vlastním boxu, jejichž pořadí je možné měnit.

Nutno zmínit, že tyto zprávy jsou součástí každé zobrazené stránky tohoto webu. Jejich informativní potenciál je tudíž velký. U zpráv, které jsou publikovány v nástěnce, se nastavuje datum, do kterého mají být publikovány. Což nám, pokud budeme poutat na nějakou akci, zajistí to, že se příspěvek nebude zobrazovat i po ukončení akce, aniž bychom do toho museli zasáhnout.

6.7 Agrobazar - katalog

6.7.1 Kategorie

Hierarchie vytvořeného katalogu je založena na vzájemně provázaných kategoriích. Tyto kategorie nám pak tvoří jakýsi ukazatel pozice v rámci prohlížení katalogu. Názvy kategorií obvykle blíže specifikují oblast strojů, jež můžeme v dané kategorii najít.

Přístup ke katalogu tohoto systému je prostřednictvím hlavního menu přes odkaz *Agrobazar*. Musím říct, že se z hlediska přehlednosti nejedná o nejlepší řešení. Ale jelikož jsem vycházel ze zadaného grafického layoutu, nemohl jsem do jeho návrhu zásadně zasahovat, spíše jen vznášet připomínky. V nejbližší době, kdy se bude pokračovat v práci na tomto katalogu, je už domluveno, že dojde ke zrušení podoby současného menu, které bude nahrazeno menu obsahující kategorie.

Z pohledu návrhu obsahu tedy nezbývalo nic jiného, než vymyslet, jak řešit při prohlížení katalogu přehlednost a umístění v dané kategorii. Bylo tedy rozhodnuto, že se při prohlížení katalogu, ihned po nadpisu kategorie nebo názvu stroje, zobrazí cesta, přes které kategorie je možné se do daného místa v katalogu dostat.

Jelikož při ukládání strojů do kategorií není přesně dáno, že stroj musí být umístěn vždy v nejnižší části náležící pro danou kategorii, můžeme se při prohlížení katalogu setkat s tím, že při zobrazení detailu kategorie se nám kromě podkategorií zobrazí také výpis nějakých strojů.

Výpis strojů je vždy uvozen takovou hlavičkou, v níž můžeme nastavavovat vlastnosti výpisu strojů. Lze například nastavit styl výpis. Můžeme si zvolit mezi tabulkovým, který obsahuje pouze název stroje a jeho cenu, nebo obrázkovým, který těchto informací o stroji obsahuje podstatně víc. Kromě jiného lze také nastavit, aby se výpis strojů řadil podle ceny či jména. Dále pak už jen na nás záleží, jaký styl zobrazení a řazení si při prohlížení katalogu zvolíme.

Na závěr této části bych ještě přiblížil tvar a řešení webových adres pro kategorie. Každá je ve tvaru *kategory-nazev_url_kategorie-id_kategorie*. Pomocí prvního řetězce před první pomlčkou určujeme jednoznačný typ adresy. Následuje upravený název kategorie, který nesmí obsahovat mezery, diakritiku a další nežádoucí znaky. A před poslední pomlčkou je číslo kategorie. Zdůvodnění tohoto zápisu se vyskytne v kapitole 6.10.

6.7.2 Karta stroje

Detaily stroje v katalogu budeme zobrazovat pomocí *karty stroje*. Jedná se o stránku, která má vždy stejnou kostru. Obsahuje název stroje, jeho umístění v rámci katalogu (pro lepší orientaci), textové informace o stroji, parametry stroje a v neposlední řadě také jeho příslušné fotografie.

Díky parametrům stroje, které se automaticky překládají do příslušné jazykové verze, je katalog schopen udělat navštěvníkovi o prohlíženém stroji bližší představu, která je pak ještě doplněna přiloženými fotografiemi.

U každého stroje existuje možnost ho vložit na parkoviště. Jedná se o podobný princip, který bývá obvykle aplikován na e-shopech. Možnosti zaparkovat stroj se věnuje kapitola 6.8, která tuto schopnost katalogu blíže přiblíží. Pokud máme aktuálně prohlížený stroj již na parkovišti, změní se tlačítko *zaparkovat stroj* na tlačítko *odporkovat stroj*, což nám ho umožní z parkoviště případně odstranit.

Jako bylo zmíněno řešení webových adres v kapitole o kategoriích 6.7.1, i v tomto případě tuto problematiku a její řešení nastíníme. U karet strojů se vyskytují adresy ve tvaru

machine-nazev_stroje-id_stroje. Popis jednotlivých částí adresy je úplně stejný jako v již zmiňované kapitole o kategoriích 6.7.1. Je zde pouze nepatrný rozdíl týkající se řetězce před první pomlčkou, který je v tomto případě pojmenován *machine*.

6.8 Parkoviště

Jelikož měl zadavatel této práce požadavek parkovat stroje, bylo mu v tomto případě vyhověno prostřednictví *parkoviště*, které ve spoustě webových aplikacích můžeme vidět v podobě košíku na zboží. Tímto umístěním dosáhne návštěvník toho, že si postupně vytváří vlastní virtuální parkoviště. Tady pak může neomezeně přistupovat a upravovat jej podle svých potřeb.

Ukládání strojů na parkoviště je řešeno pomocí cookie, jejichž chování bylo popsáno v kapitole 3.8. Pro ukládání byla zvolena právě tato metoda, protože si zadavatel přál, aby se stroje umístěné na parkovišti po uzavření prohlížeče nezrušily, ale byly po určitou dobu uložené.

Z výpisu strojů na parkovišti můžeme vidět, že zde jsou jen minimální orientační údaje o jednotlivých parkujících strojích. Je to z toho důvodu, že parkoviště jako samotné nám nemůže dobře posloužit k porovnávání strojů. Za tímto účelem existuje na parkovišti odkaz na zobrazení výpisu parkujících strojů. Tato problematika je probírána v kapitole 6.9.

Před přechodem na srovnávání strojů si můžeme na parkovišti zvolit, které stroje si přejeme do porovnávání zařadit. To se provádí pomocí zaškrtávacích polí, jež jsou vedle každého názvu stroje. Výhoda tohoto zpracování je v tom, že nemusíme nutně všechny parkující stroje na porovnávání posílat.

6.9 Srovnávání strojů

Aby byl celý katalog efektivní z funkčního hlediska, bylo nutné u něj aplikovat funkci pro porovnávání strojů. V našem katalogu se k porovnávání strojů můžeme dostat prostřednictvím parkoviště, o jehož činnosti jsme se více dozvěděli v kapitole 6.8.

Stroje, které byly z parkoviště poslány na porovnání, jsou následně zobrazeny v tabulkovém výpisu. Tabulka se skládá z několika sloupců a řádků, přičemž každému sloupci, kromě prvního, odpovídá jeden stroj. První sloupec obsahuje názvy parametrů pro jednotlivé řádky tabulky.

Jak je asi jasné, je možné porovnávat rozdílné stroje, které nemusí mít žádný společný parametr nebo mohou mít společnou pouze část parametrů. Aby nebyl výpis porovnávaných parametrů nesystematický, řeší se řazení porovnávaných parametrů následujícím způsobem. Na první řádky se vkládají parametry, jejichž výskyt je v nejvíce porovnávaných strojích. Tak je to stupňováno až po stav, kdy v jednotlivých řádcích můžeme najít pouze jednu hodnotu parametru charakteristickou právě pro určitý stroj.

Aby bylo prohlížení jednotlivých parametrů příjemnější, je každý řádek při přejetí kursoru myši zvýrazněn.

6.10 Přepisování adres

Protože spousta internetových vyhledávačů při indexování internetových stránek do svých databází zvyšuje relevanci stránky i na základě webové adresy, na které se stránka vyskytuje. Musí být v našem zájmu, abychom do této adresy dostali co nejvíce klíčových

slov vztahujících se k vlastnímu obsahu stránky. Při optimalizaci stránek pro vyhledávače je často přistupováno k tomuto úkonu. Klíčová slova, která máme umístěna v titulku stránky, v meta tazích a hlavním nadpisu stránky, nám mohou značně ovlivnit pozici v umístění ve vyhledávačích. Samozřejmě se nejedná o jistou věc. Optimalizace pro vyhledávače tvoří svůj vlastní obor. Jak se můžeme dočíst ve spoustě článků na internetu, neexistují přesně daná pravidla, která by vždy byla schopná posunout naše stránky na první pozice ve vyhledávačích. Abychom na tyto pozice stránky dostali, je třeba provést ještě několik dalších věcí, jež tady nemá smysl rozebírat.

Důvod, proč přepisovat adresy, už známe. Nyní se podíváme, jakým způsobem je řešena tato problematika ve vytvářeném katalogu. Na začátek je třeba zmínit, že zápis adres už vychází z jazykových verzí, které jsou pro katalog dostupné. Každá adresa ihned po doméně obsahuje jazykovou verzi, ve které stránky prohlížíme. Po této jazykové mutaci se mohou vyskytovat názvy stránek. Kromě jiného se na konci adresy může objevit ještě číslo, které označuje číslo strany. Zpracovávané adresy neobsahují žádné další rozšíření. Chceme-li docílit toho, aby každá adresa byla z hlediska svého obsahu unikátní, nesmí se zde vyskytovat nadřazené kategorie, které nám při prvním pohledu mohou vytvářet dojem systematického zanořování. Adresa bude mít následující tvar:

http://domena/jazyk/zobrazovana-stranka/cislo-strany

Nyní se podívejme na řešení vytváření tvaru adresy pro část *zobrazovaná stránka*. Obecně to můžeme zapsat následujícím způsobem.

[kategory|machine] - nazev_stranky - identifikátor

K tomuto tvaru bylo přistoupeno z důvodu řešení unikátních adres pro celý katalog. Před první pomlčkou se vyskytuje řetězec specifikující skupinu stránek, do níž náleží zobrazovaná stránka. V rámci našeho katalogu se bude jednat o skupinu stránek *kategory* vztahující se ke kategoriím a pak skupinu stránek *machine* vztahující se ke strojům. Řetězec, který se vyskytuje mezi první a poslední pomlčkou, blíže specifikuje stroj či kategorii, jež je aktuálně zobrazena na stránce. Za poslední pomlčkou pak můžeme nalézt číselný identifikátor vztahující se k záznamu v databázi.

Při zpracovávání adres se z nich získává jazyk, jenž je aktuálně používán, typ prohlížené stránky, upravený tvar názvu kategorie či stroje a identifikátor. Všechny tyto hodnoty jsou kontrolovány, tudíž se nemůže stát, že by název neodpovídal identifikátoru v databázi. V případě zadání neplatné adresy je zaslána hlavička s odpovídajícím chybovým kódem a v prohlížeči je zobrazena chybová stránka.

Kapitola 7

Plány na další rozšíření katalogu

Jak už to bývá u všech rozsáhlých projektů, i tady platí, že se v této práci bude nadále pokračovat a postupně hledat nejlepší řešení pro jednotlivé části systému. Některé tyto problémy již byly v předešlém textu zmíněny.

Na poslední poradě se dospělo k názoru, že by bylo dobré, aby celý katalog fungoval po vzoru katalogů již dobře fungujících. Zjistilo se, že vzor zahraničních katalogů není z hlediska navigace dostatečně vyhovující a bylo by lepší řešit tento problém zrušením stávajícího a nahrazením nového menu, jež by obsahovalo kategorie katalogu.

Další naše myšlenka, která se pomalu začíná ve větších projektech objevovat, je sloučení části administrace s prezentační částí. Je nutné dodat, že v dnešní době komerčních informačních systému se to objevuje jen zřídkakdy. Jednalo by se o to, že na stránkách by byl přihlašovací formulář pro editaci. Po přihlášení by se nám pak grafika webu nezměnila, pouze by se stalo to, že by se nám případně objevily funkce pro ovládání nastavení webu, samozřejmě na základě přístupových práv. Prohlížení katalogu by probíhalo standardním způsobem s doplněním nových odkazů pro úpravu katalogu. Dalo by se říci, že by bylo možné povolit také přístup registrovaným uživatelům, na které v současném systému není pamatováno, protože se uvažovalo nad možností, používat tento katalog pouze soukromě. Tací noví uživatelé by mohli do systému vkládat nové inzeráty na bazarové stroje. V tomto případě by bylo samozřejmostí schvalování nově vložených inzerátů administrátorem webu.

Pro registrované uživatele s právy návštěvníka by mohla existovat také možnost zasílání přímých poptávek či rezervací na zvolený stroj.

Z hlediska přístupu do klienské části webu se uvažuje nad možností, že by mohla být možnost pouze jednoho přihlášení od jedné instance uživatele. To znamená, že by připadalo v úvahu, aby se na jeden uživatelský účet mohl v jednu chvíli přihlásit pouze jeden uživatel. Pro jiné návštěvníky, využívající stejné přihlašovací údaje, by se tento účet stal zamčený.

Kapitola 8

Závěr a hodnocení práce

Výsledkem této práce, jenž byla popisována v tomto textu, je použitelný katalog zemědělské techniky, u kterého se počítá s reálným nasazením. Díky vyvinutému systému této aplikace, je možné jeho jádro použít také pro zcela odlišný projekt, který s katalogem nemusí mít nic společného.

Jak již bylo zmíněno v kapitole 7, ostré spuštění aplikace se očekává až po dokončení zmiňovaných změn, se kterými bohužel na počátku projektu nebylo počítáno. Aplikaci, která je popsána v této práci, je možné spatřit na dočasné adrese http://agrobazar.zahorovice.cz.

Při vývoji systému bylo dbáno na použitelnost a využívání nejnovějších technik. Snahou bylo, aby se odstranily problémy vycházející z používání aplikace pod webovým prohlížečem.

Jsem rád, že jsem svou bakalářkou práci mohl postavit na reálném projektu, který nebyl vytvářen bezúčelně. Jedná se o velmi dobrou zkušenost. Pro takovou práci, o které člověk ví, že bude mít jisté uplatnění, se mnohem snadněji hledá motivace, než by tomu mohlo být v případě opačném.

Výhodou tohoto projektu byla představa, že podobná aplikace již existuje. Sice jsem neměl možnost přístupu do administrace existujícího webu, ale zařídil jsem se tak, aby nový systém významně usnadnil práci s katalogem. Mohlo by se zdát, že práce s návrhem grafické šablony se dala zkopírovat z existujících stránek, ale z důvodu jiného názoru na zdrojový kód byla tato šablona vytvářena úplně od začátku za účelem dosažení co nejlepší optimalizace pro vyhledávače.

Musím říci, že prací na tomto projektu jsem se naučil systematicky rozebrat složité problémy na menší, které jsem pak řešil separátně. Díky tomuto systému jsem si osvojil schopnost přemýšlet nad možným rozšiřováním složitých systémů, řešit tak možnou modularitu a s ní spojené problémy.

Neměl bych zapomenout ani na to, že jsem si vyzkoušel práci s přepisováním adres, které jsou pro optimalizaci jedním z důležitých faktorů. K výborným zkušnostem z tohoto projektu lze také zařadit práci s dnes prosazující se technikou AJAX a její schopností dynamicky pracovat s celou stránkou. Za zmínku asi stojí jeden z nejpěknějších způsobů editace textu, kdy editujeme text na místě samém a to jen díky tomu, že například dvakrát poklikáme na text, který chceme editovat.

Kdybych měl zhodnotit tuto práci, myslím si, že určitě splnila očekávání zadavatele. Jak již bylo dříve zmíněno, měl jsem snahu zachovat rozsah funkcí stávajícího komerčního systému. Mým úkolem bylo tyto funkce naopak obohatit a rozšířit tak schopnosti celého katalogu. Tyto schopnosti je možné zhlédnout jak na straně administrace, tak na straně prezentace katalogu a to konkrétně v podobě, v tomto textu již několikrát zmiňované, možnosti parkování a srovnávání strojů podle jejich parametrů.

Dodatek A

Instalace a první spuštění

Na CD, které je přiloženo k této práci, můžeme najít aplikaci připravenou k instalaci a použití na libovolném webovém serveru, jenž obsahuje podporu PHP s mod_rewrite a MySQL. K tomu, aby byla aplikace funkční, je nutné před jejím prvním spuštěním provést počáteční inicializaci, která se skládá z následujících kroků.

1. Zkopírování zdrojových souborů na webový server.
2. Spuštění serverové aplikace *phpMyAdmin* a následná inicializace databáze MySQL pomocí SQL skriptu *katalog.sql*, který je uložen v kořenovém adresáři webu.
3. Nastavení přístupových práv zápisu pro adresáře, do nichž se budou ukládat soubory při nahrávání v administraci. Jedná se o následující složky, jenž se vyskytují v kořenovém adresáři webu.
   - /obrazky-aktualita/
   - /obrazky-nastenka/
   - /obrazky-produktu/
4. Aby aplikace katalogu pracovala správně s databází, je nutné nastavit připojovací údaje k ní v konfiguračním souboru *config.php*. Ten se nachází ve složce */etc/* uložené v kořenovém adresáři webu. Při nastavování se musíme řídit pokyny, které jsou uvedeny u jednotlivých parametrů.
5. Poslední duležitou věcí, kterou je nutné udělat, aby katalog správně pracoval, je nastavit soubor zajišťující přepis URL adres. Soubor se jmenuje *.htaccess* a je uložen v kořenové složce webu. Editace tohoto souboru se provádí pouze případě, kdy chceme, aby katalog nepracoval přímo v základní doméně, ale byl například zanořen v nějakém adresáři. Bližší informace jsou uvedeny ve zmiňovaném souboru.
6. Po provedení předešlých kroků by měla být aplikace katalogu funkční, avšak neobsahuje žádná uložená data. Abychom mohli modifikovat obsah katalogu, přistoupíme přes adresu *doména/admin* do administrace. Je nutné se přihlásit. Aktivní je pouze jediný uživatel, jenž má následující přihlašovací údaje.
   - login: admin
   - heslo: aaaaaa

Literatura

[1] Naramore, E., aj. *PHP5, MySQL, Apache: vytváříme webové aplikace*. Computer Press, a.s., 2006. ISBN 80-251-1073-7.

[2] WWW stránky. HTTP - Wikipedie, otevřená encyklopedie. Dokument dostupný na URL http://cs.wikipedia.org/wiki/Hyper_Text_Transfer_Protocol [online] (duben 2008).

[3] Burget, R., Zeman, D. Tvorba webových stránek. Dokument dostupný na URL https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/ITW-IT/texts/opora_itw_061020.pdf [online] (duben 2008).

[4] WWW stránky. JavaScript - Wikipedie, otevřená encyklopedie. Dokument dostupný na URL http://cs.wikipedia.org/wiki/JavaScript [online] (duben 2008).

[5] Škultéty, R. *JavaScript: Programujeme internetové aplikace*. Computer Press, a.s., 2001. ISBN 80-7226-457-5.

[6] Holzner, S. *Mistrovství v AJAXu*. Computer Press, a.s., 2007. ISBN 978-80-251-1850-4.

[7] WWW stránky. SQL - Wikipedie, otevřená encyklopedie. Dokument dostupný na URL http://cs.wikipedia.org/wiki/SQL [online] (duben 2008).

[8] WWW stránky. Mikův weblog | mod_rewrite a hezké url. Dokument dostupný na URL http://mike.webzdarma.cz/mod_rewrite-a-hezke-url/ [online] (duben 2008).

[9] WWW stránky. SWFUpload. Dokument dostupný na URL http://swfupload.org/ [online] (duben 2008).

[10] WWW stránky. Cross-Browser Rich Text Editor (RTE). Dokument dostupný na URL http://www.kevinroth.com/rte/demo.htm [online] (duben 2008).

[11] WWW stránky. YAML | Example 3col_fixed_seo. Dokument dostupný na URL http://www.yaml.de/fileadmin/examples/05_layouts_advanced/3col_fixed_seo.html [online] (duben 2008).