UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

WWW aplikace pro inzerci motocyklů

Vojtěch Pešl

Bakalářská práce
2011

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2010/2011**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Vojtěch PEŠL**
Osobní číslo:	**I07750**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**WWW aplikace pro inzerci motocyklů**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Analýza: Srovnání stávajících informačních systémů pro inzerci vozidel a jejich porovnání se systémem pro inzerci motocyklů.

Teoretická část: Představení technologií sloužících k tvorbě webových aplikací. Dále se bude zabývat návrhem vhodné databáze a možnostmi zabezpečení dat včetně problematiky SQL Injection.

Praktická část: Návrh architektury webové aplikace pro inzerci motocyklů a návrh databáze MySQL. Praktické řešení aplikace bude umožňovat vkládání inzerátů od registrovaných uživatelů, při vkládání bude normovat velikost fotografií a hlídat povinné údaje pro vložení, editaci inzerátů, pokročilé vyhledávání podle více kriterií, přístup dle práv (administrátor, registrovaný uživatel).

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

[1] BRÁZA, Jiří. PHP 5 : začínáme programovat. 1. Praha : Grada Publishing, 2005. 244 s. ISBN 80-247-1146-X.

[2] KOSEK, Jiří. PHP - tvorba interaktivních internetových aplikací : podrobný průvodce. 1. Praha : Grada Publishing, 1999. 492 s. ISBN 80-7169-373-1.

[3] GROFF, James R.; WEINBERG, Paul N. SQL : kompletní průvodce. 1. Brno : Computer Press, 2005. 936 s. ISBN 80-251-0369-2.

Vedoucí bakalářské práce: **prof. Ing. Karel Šotek, CSc.**
Katedra softwarových technologií

Datum zadání bakalářské práce: **17. prosince 2010**

Termín odevzdání bakalářské práce: **13. května 2011**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2011

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 2. 5. 2011 Vojtěch Pešl

Poděkování

Zde bych chtěl poděkovat prof. Ing. Karlovi Šotkovi, CSc. za pomoc a cenné informace při vypracování bakalářské práce.

Anotace

Bakalářská práce se v teoretické části zabývá popisem technologií sloužících pro tvorbu webových aplikací a možnostmi zabezpečení dat včetně problematiky SQL Injection. Praktická část popisuje vytvoření informačního systému pro inzerci motocyklů, jeho hlavní funkce a způsob implementace.

Klíčová slova

webová aplikace, inzerce, MySQL, PHP, zabezpečení

Title

WWW applications for advertising motocycles

Annotation

In the theoretical part, the bachelor thesis deals with the description of the technologies used for creating the web applications and data security including problems with SQL Injection. The practical part describes creation of the information system for advertising motocycles, it is main functions and the way of implementation.

Keywords

web aplication, advertising, MySQL, PHP, security

Obsah

Seznam zkratek

AJAX	Asynchronous JavaScript and XML
ASP	Active Server Pages
CSS	Cascading Style Sheets
GUI	Graphical User Interface
HTML	HyperText Markup Language HTTP HyperText Transfer Protocol
PHP	Hypertext Preprocessor
SGML	Standard Generalized Markup Language
SQL	Structured Query Language
URL	Uniform Resource Locator
WWW	World Wide Web
XHTML	eXtensible HyperText Markup Language
XML	eXtensible Markup Language

Seznam obrázků

Seznam tabulek

1 Úvod

Cílem této bakalářské práce je vytvoření informačního systému pro inzerci motocyklů. V teoretické části bakalářské práce jsou popsány technologie k tvorbě webových aplikací, návrh vhodné databáze a možnosti zabezpečení dat včetně problematiky SQL Injection.

V praktické části bakalářské práce jsou srovnány stávající informační systémy pro inzerci motocyklů, dále je zde popsáno vytvoření informačního systému, jeho hlavní funkce a způsob implementace.

Teoretická část

2 Technologie k tvorbě webových aplikací

2.1 World Wide Web

Je označení pro aplikace internetového protokolu HTTP. Je tím myšlena soustava propojených hypertextových dokumentů. Dokumenty umístěné na počítačových serverech jsou adresovány pomocí URL, jehož součástí je i doména a jméno počítače. Název naprosté většiny těchto serverů začíná zkratkou www, i když je možné používat libovolné jméno vyhovující pravidlům URL. Protokol HTTP je dnes již používán i pro přenos jiných dokumentů, než jen souborů ve tvaru HTML [1].

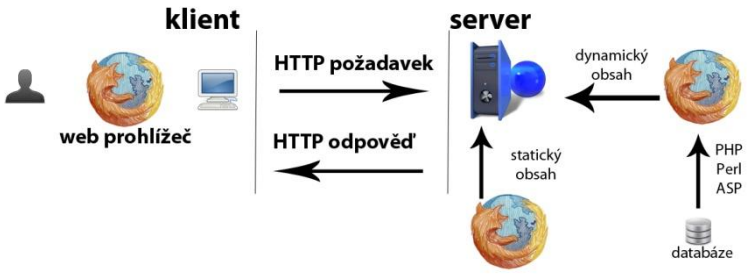
2.2 Webový server

Počítač, který je odpovědný za vyřizování požadavků HTTP od klientů - programů zvaných webový prohlížeč. Vyřízením požadavků se rozumí odeslání webové stránky. Webové stránky jsou obvykle dokumenty HTML.

Jednotlivé webové servery se mohou v různých jednotlivostech značně lišit. Přesto mají několik společných vlastností. Každý webový server je připojen k počítačové síti a přijímá požadavky ve tvaru HTTP. Tyto požadavky vyřizuje a počítači, který požadavek vznesl, vrací odpověď. Odpověď obvykle představuje nějaký HTML dokument. Může to být ale i dokument v jiném formátu.

Webový server má v zásadě dvě možnosti, jak získávat informace, které vrací klientům:

- Předem připravené datové soubory (HTML stránky), které webový server bez změny poskytne klientovi tzv. statický obsah.
- Na základě požadavku klienta jsou data shromážděna, přečtena ze souboru nebo databáze, zformátována a připravena k prezentaci ve formátu HTML a poskytnuta webovému prohlížeči tzv. dynamický obsah [2].



Obrázek 1 - webový server

2.3 Webová aplikace

Je aplikace poskytovaná uživatelům z webového serveru přes počítačovou síť Internet, nebo její vnitropodnikovou obdobu (intranet). Webové aplikace jsou populární především pro všudypřítomnost webového prohlížeče jako klienta. Ten se pak nazývá tenkým klientem, neboť sám o sobě logiku aplikace nezná. Webové aplikace jsou používány pro implementaci mnoha podnikových i jiných informačních systémů, ale i free e-mailů, internetových obchodů, online aukcí, diskusních fór [3].

Web aplikace lze rozdělit do mnoha kategorií typickým rozdělením jsou statické a dynamické.

Statický web podává informace uživateli přesně tak, jak je tvůrce webu napsal. Klient tak může pouze přebírat informace a přecházet mezi jednotlivými stránkami. Statický web je obvykle napsán v jazyce HTML. Zdrojový kód stránek se pak odešle do prohlížeče, který kód zná a podle něj vytvoří vzhled stránky a zobrazí požadované informace. Výhodou statických webů je jejich jednoduchost, kdy si funkční statický web může udělat prakticky kdokoliv. Nevýhodou pak je nemožnost, aby uživatel jakkoli zasahoval do zobrazovaného obsahu [4].

Dynamický web je obvykle rozšířením statických webů o prvky, které se vyhodnotí a sestaví až po určité uživatelské akci. Typickými zástupci jazyků, v nichž se dynamický obsah tvoří, jsou PHP, PERL či ASP, lze však využít jazyky Java, JavaScript a jiné. Principem je vykonání skriptu s danými vstupními daty a následné vyhodnocení a sestavení statické stránky, která se zobrazí. Technologie, které dynamický web představují, se dělí především na serverové a klientské. To podle toho, kde se dynamické skripty vyhodnocují. U serverových technologií je skript vyhodnocen na straně serveru a prohlížeči se tak odešle výsledná statická stránka, sestavená obvykle do HTML (PHP či ASP). Klientské technologie spočívají v odeslání zdrojových kódů stránky prohlížeči, který tak danou technologii musí znát, vyhodnotit skripty a pak teprve sestavit výslednou stránku (JavaScript). Výhody dynamického webu jsou zřejmé, možnost interakce mezi webem a klientem. Nevýhoda spočívá jen ve vyšší složitosti a nutnosti hlubších znalostí internetových technologií a programování [5].

2.4 Značkovací jazyk

Je jazyk, jehož zdrojový text obsahuje současně jak vlastní text, tak instrukce pro jeho zpracování. Ty se zpravidla vyskytují v podobě příkazů či značek [6].

2.4.1 HTML

HyperText Markup Language je značkovací jazyk pro hypertext. Je jedním z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci dokumentů na Internetu. Jazyk je aplikací dříve vyvinutého rozsáhlého univerzálního značkovacího

jazyka SGML[1]. Vývoj HTML byl ovlivněn vývojem webových prohlížečů, které zpětně ovlivňovaly definici jazyka [7].

2.4.2 XHTML

Zkratka XHTML z anglického extensible hypertext markup language. XHTML je značkovací jazyk, který má přesně definovanou syntaxi, která stanovuje, co a jak se má psát. V předchozích verzích HTML často nebyla a také stále není tato syntaxe přesně dodržována. V XHTML, které mnohem více, jak HTML, vychází z technologie XML[2], je nutné tuto syntaxi přesně dodržovat [8].

2.5 CSS

CSS je zkratka pro anglický název Cascading Style Sheets. Jazyk byl navržen standardizační organizací W3C. Byly vydány zatím dvě úrovně specifikace CSS1 a CSS2, dokončuje se revize CSS 2.1 a pracuje se na verzi CSS3. Hlavním smyslem je umožnit návrhářům oddělit vzhled dokumentu od jeho struktury a obsahu. Původně to měl umožnit už jazyk HTML, ale v důsledku nedostatečných standardů a konkurenčního boje výrobců prohlížečů se vyvinul jinak. Starší verze HTML obsahují celou řadu elementů, které nepopisují obsah a strukturu dokumentu, ale i způsob jeho zobrazení. Z hlediska zpracování dokumentů a vyhledávání informací není takový vývoj žádoucí [9].

2.6 Programování webových aplikací

Pro programování webových aplikací lze použít mnoho programovacích jazyků. Jak jsem již uvedl, programovací jazyky můžeme rozdělit do dvou skupin, podle toho zda se skript vykonává na straně serveru nebo na straně klienta.

Ačkoli je mnoho webových aplikací psáno přímo v čistém programovacím jazyce jako je PHP či Perl[3], existuje pro jejich tvorbu řada systémů, tzv. frameworků, které díky automatizaci tohoto procesu nabízejí programátorům možnost popsat program na vyšších úrovních [2].

2.6.1 PHP

PHP je rekurzivní zkratka z Hypertext Preprocessor, což je skriptovací programovací jazyk, určený především pro programování dynamických internetových stránek. Nejčastěji se začleňuje přímo do struktury jazyka HTML, XHTM.

Při použití PHP pro dynamické stránky jsou skripty prováděny na straně serveru. PHP je vedle ASP[4] jedním ze dvou nejrozšířenějších skriptovacích jazyků pro web. Oblíbeným se stal především díky jednoduchosti použití, bohaté zásobě funkcí, a tomu, že kombinuje vlastnosti více programovacích jazyků a nechává tak vývojáři částečnou svobodu v syntaxi. V kombinaci s operačním systémem Linux, databázovým systémem

[1] http://cs.wikipedia.org/wiki/SGML
[2] http://cs.wikipedia.org/wiki/XML
[3] http://cs.wikipedia.org/wiki/Perl
[4] http://cs.wikipedia.org/wiki/Active_Server_Pages

MySQL nebo PostgreSQL a webovým serverem Apache[5] je často využíván k tvorbě webových aplikací [10].

Od PHP 4 před několika lety začaly vznikat první knihovny se zobecněnými funkcemi a třídami. Začaly také vznikat i první frameworky. Slabá podpora objektového programování v PHP 4 vývoji podobných nástrojů příliš nepřála. Právě pětková verze přinesla možnosti pro vznik velkých nástrojů jako jsou Zend Framework, CakePHP, Symfony a další. A to hlavně svým vylepšeným objektovým přístupem, lepší podporou utf8, množstvím nových funkcí a knihoven do PHP přímo zabudovaných, PDO knihovnou pro přístup k databázi a také solidní prací s XML [11].

2.6.2 JavaScript

Používá se jako interpretovaný programovací jazyk pro WWW stránky, často vkládaný přímo do HTML kódu stránky. Jsou jím obvykle ovládány různé interaktivní prvky GUI[6] (tlačítka, textová políčka) nebo tvořeny animace a efekty obrázků. Program v JavaScriptu se obvykle spouští až po stažení WWW stránky z Internetu na straně klienta. Z toho plynou jistá bezpečnostní omezení, JavaScript např. nemůže pracovat se soubory, aby tím neohrozil soukromé uživatele [12].

2.7 Databázové systémy, jazyk SQL

Jazyk SQL je nástroj pro organizování, správu a získávání dat uložených v počítačové databázi. Zkratka SQL je odvozená z anglického slova Structured Query Language. Jak vyplívá z názvu, SQL je počítačový jazyk, který se používá pro komunikaci s databází. Ve skutečnosti SQL pracuje s jedním specifickým typem databáze, který nazýváme relační databáze [13].

2.7.1 MySQL

MySQL je databázový systém, vytvořený švédskou firmou MySQL AB, nyní vlastněný společností Sun Microsystems, dceřinou společností Oracle Corporation. Je považován za úspěšného průkopníka dvojího licencování – je k dispozici jak pod bezplatnou licencí GPL[7], tak pod komerční placenou licencí.

MySQL bylo od počátku optimalizováno především na rychlost, a to i za cenu některých zjednodušení: má jen jednoduché způsoby zálohování, a až donedávna nepodporovalo pohledy, triggery, a uložené procedury. Tyto vlastnosti jsou doplňovány teprve v posledních letech, kdy začaly nejčastějším uživatelům scházet [14].

2.7.2 PostgreSQL

PostgreSQL je relační databáze podporující SQL92 a SQL99 normu a má mnoho dalších moderních rysů. Obsahuje také triggery, pohledy, transakce, vlastní datové typy, agregační funkce, procedury a mnoho dalšího.

[5] http://cs.wikipedia.org/wiki/Apache_HTTP_Server
[6] GUI - Grafické uživatelské rozhraní
[7] http://cs.wikipedia.org/wiki/GPL

Z hlediska programátora je výborná možnost přístupu z mnoha jazyků C/C++ pomocí knihovny libpq a libpq++, scriptovacích jazyků Perl pomocí knihovny pgsql_perl5, Python pomocí knihovny PyGreSQL, ale i třeba JAVA pomocí knihovny jdbc a PHP [15].

2.7.3 Oracle

Oracle je systém řízení báze dat, moderní multiplatformní databázový systém s velice pokročilými možnostmi zpracování dat, vysokým výkonem a snadnou škálovatelností.

Aktuální verzí je Oracle Database 11g. Tento systém podporuje nejen standardní relační dotazovací jazyk SQL podle normy SQL92, ale také proprietární firemní rozšíření Oracle, imperativní programovací jazyk PL/SQL rozšiřující možnosti vlastního SQL, dále podporuje objektové databáze a databáze uložené v hierarchickém modelu dat . Také obsahuje širokou paletu nástrojů pro podporu snadného nasazení na gridových sítích [16].

3 Zabezpečení dat

3.1 PHP injection

PHP Injection je vcelku často vyskytující se chyba. PHP Injection se také někdy nazývá nechráněná inkluze. Tento název trochu lépe popisuje to, o co se vlastně jedná. "Inkluze" vychází se slova include, což je funkce jazyka PHP pro vkládání externího souboru (html, php, txt) do skriptu. A co znamená "nechráněná" je asi každému jasné. Tato chyba tedy umožní útočníkovi uploadovat na server svůj skript. Pokud na serveru, na němž je napadený web uložen, neběží safe_mode, může útočník získat nad webem úplnou kontrolu. Může mazat, editovat, či přidávat další soubory, což může být pro postižený web zničující [22].

Příklady a obrana

Typická adresa napadnutelná přes PHP Injection vypadá následovně:

```
http://nejakyweb.cz/index.php?promenna=stranka.php
```

Potom soubor index.php musí obsahovat podobný kód:

```
include $_GET['promenna'];
```

Funkce při tomto použití vloží do stránky jakýkoliv soubor, který zadáme v parametru URL adresy. Nemusí se však jednat pouze o soubor, ale i o celý jiný web v případě, že místo jména souboru vložíme absolutní URL adresu:

```
http://nejakyweb.cz/index.php?promenna=http://hecker.tld.cz
```

Do napadeného webu bude vložen zdrojový kód ze stránky http://hecker.tld.cz a ve výsledku uvidíte chaos způsobený zkřížením původního a includovaného webu.

Proti PHP Injection je však jednoduchá obrana, kontrolovat proměnnou $_GET['promenna'] nebo od verze PHP 4 vypnout v souboru php.ini funkci allow_url_fopen(), která dovoluje includovat soubory i z jiných serverů [22].

3.2 SQL Injection

Pod pojmem SQL injection se skrývá podvržení vstupních dat (hodnot proměnných odesílaných serveru) tak, aby byl nějakým způsobem pozměněn výsledek SQL dotazu. Pokud útočník zná strukturu tabulky (nejlépe i SQL dotazů), které svými proměnnými ovlivní, má vše daleko jednodušší, než když musí odhadovat, jaké sloupce jsou použity, jaké mají asi datové typy a jak se jmenují. Proto je důležité, aby skripty podávaly co nejméně informací o struktuře vašich databází a tabulek v případě, že se vyskytne nějaká chyba [23].

Co hrozí?

Při nejlepším se útočník dostane tam, kam nemá, ale neprovede nic s vaší aplikací; v horším případě se dostane třeba k uživatelským heslům a v nejhorším případě vám smaže tabulky nebo upraví jejich obsah [24].

Příklady a obrana

Pro zobrazení článku pomocí ID nám může posloužit tento dotaz:

```
//nezabezpeceny nachylny dotaz
$dotaz = "select * from clanky where id = '$_GET["id"]'";
```

Tento kód načítá vše z tabulky clanky kde id = proměnné id z adresy. Jak vidíte, tento vstup není nijak chráněný, a tak útočník může zadat tento dotaz do adresy:

```
clanek.php?id=1 and 1=1/*
```

Pokud se nám zobrazí článek, útočník pozná, že je zde chyba, které může zneužít:

```
clanek.php?id=1 and truncate table clanky/*
```

Tímto dotazem vám vyprázdní tabulku clanky, ale může vám ji také smazat, vložit do ní nové údaje [24].

Jak se bránit?

Pokud má být id číslo, tak použijeme PHP funkci is_numeric().

```
$id = $_GET["id"];
//kontrola, zda je id cislo
if (!is_numeric($id)):
echo "Toto ne!";
else:
$dotaz = "select * from clanky where id='$id'";
endif;
```

Tímto jsme zabránili vložení jakéhokoliv SQL dotazu do adresy, protože se kontroluje, jestli je proměnná id číslo nebo ne. Pokud by id mělo být typu string, použijeme PHP funkci is_string.

Toto byl jednoduchý příklad SQL Injekce a také jednoduchý příklad ochrany. Zásadou při psaní každé webové aplikace je kontrola a zabezpečení všech vstupních polí od uživatele, tzn. escapování všech znaků, na některých serverech to za nás dělá direktiva magic_quotes_gpc, ale ne vždy je dobré se na toto spoléhat. Kontrolujte, zda proměnné jsou stejného typu, jaký vy požadujete (ukázáno v předchozím příkladu). Všechny výstupy, které nechcete formátovat pomocí HTML, například komentáře, nechte ošetřit pomocí funkce html_entities() nebo pomocí html_specialchars(), abyste zabránili také XSS (Cross-site-scripting)[8] [24].

3.3 Ukládání hesel - hash

Pokud se uživatelé ve webové aplikaci mohou registrovat, obvykle pro jejich přihlášení požadujeme heslo, které se typicky ukládá do databáze. Jednou ze závažných chyb je ukládat toto heslo bez jakéhokoliv kódování tak, jak ho uživatel zadal. Umožňuje to sice jednoduchou implementaci funkce poslání zapomenutého hesla, ale skrývá to riziko hromadného získání všech uživatelských údajů – a to přinejmenším autorem aplikace a správcem serveru. Proto je vhodné místo celého hesla ukládat pouze jeho hash a při ověřování kontrolovat hash zadaného řetězce s hashem uloženým.

Hashovací funkce převedou jakýkoliv řetězec na hash pevné délky, ze kterého se původní řetězec nedá v rozumném čase zpátky zjistit. Navíc by nemělo být možné v rozumném čase najít jiný řetězec, pro který funkce vrátí stejný hash.

Byť se tento způsob ukládání hesla dá v podstatě považovat za bezpečný, skrývá ještě jedno riziko. Tím je kolize hesel – když dva uživatelé zadají stejné heslo, tak má i stejný hash a když to jeden z uživatelů jakkoliv zjistí, může se přihlásit i na druhého uživatele. Řešit se to dá např. tak, že se pro každého uživatele vygeneruje náhodný řetězec, který se s heslem vhodně smíchá. Metoda smíchání není kritická a nebylo prokázáno, že např. pouhé zřetězení náhodného řetězce s heslem sníží bezpečnost celého řešení [21].

4 Návrh vhodné databáze

4.1 Výběr databáze

Před začátkem tvorby aplikace bylo nutné vybrat vhodnou databázi. V úvahu připadaly databáze Oracle Database 10g, MySQL a PostgreSQL.

[8] http://cs.wikipedia.org/wiki/Cross-site_scripting

Vzhledem k tomu, že tato aplikace je webová a na mnoha webhostinzích není podpora databáze Oracle, byla tato databáze vyřazena z výběru i přesto, že je velice využívaná ve velkých firmách a má dobrou podporu PL/SQL.

Další databáze, která přišla v úvahu, byla databáze PostgreSQL. Hlavní výhodou této databáze je, že je zdarma. Tím pádem se vyskytuje více na webhostinzích oproti databázi Oracle. Do nástupu poslední verze MySQL měla výhodu v tom, že MySQL neobsahovala pohledy, triggery a uložené procedury. Proto nic nebránilo vybrat databázi MySQL, která je jednoznačně populárnější a vyskytuje se ve standardní nabídce i těch nejlevnějších webhostingů. V poslední řadě měla být webová aplikace programována v jazyce PHP. Tento programovací jazyk má mnoho tutoriálů, kde většina seznamuje uživatele s databází MySQL.

4.2 Tabulky MySQL

Specialitou MySQL je to, že se při vytváření nové tabulky musí zadat její typ. MySQL podporuje různé typy tabulek, které se navzájem liší řadou vlastností. Mezi nejdůležitější typy tabulek patří tabulky MyISAM, InnoDB a HEAP.

4.2.1 HEAP

Tabulky HEAP jsou zajímavé tím, že se vytváří pouze v operační paměti RAM a že používají tzv. index typu hash, který umožňuje rychlý přístup k záznamům. Tabulky typu HEAP by se měly používat pouze tehdy, pokud chceme maximální rychlostí spravovat malá množství dat. Vzhledem k tomu, že se tento typ tabulek ukládá pouze do paměti RAM, tabulky se po ukončení MySQL odstraní [17].

4.2.2 MyISAM

MyISAM představuje v MySQL standardní typ tabulek. Jedná se o stabilní, vyspělý a jednoduše spravovatelný typ tabulek. Pokud nemáme žádný zvláštní důvod, proč použít jiný typ, většinou použijeme tento [17].

4.2.3 InnoDB

Typ InnoDB je v porovnání s předchozím typem nový. Podporuje všechny vlastnosti typu MyISAM, navíc se ještě pyšní dvěma odlišnostmi. Dají se zde spouštět transakce a podporují používaní pravidel integrity. Existují i důvody, které hovoří proti používání tabulek typu InnoDB. InnoDB ještě nedosáhl tak vysoké stability typu MyISAM, nemůžeme zde vytvářet fulltextový index, správa tabulek InnoDB je složitější, komerční licence v MySQL s podporou InnoDB je dražší než verze bez ní.

Použití transakcí a pravidel integrity bylo pro mě velmi důležité, proto jsem tabulky typu InnoDB využíval v celé webové aplikaci [17].

4.3 Návrh databáze

Na začátku vývoje každé databáze stojí především její návrh. Návrh databáze, má vliv na to, jak bude výkonná výsledná aplikace, jak jednoduše, nebo naopak složitě se bude

programovat a spravovat a konečně, jak to bude s její flexibilitou v případě, kdy v ní bude nutno provést nějaké změny. Chyby, kterých se při návrhu dopustíme, se později budou napravovat jen velmi špatně [17].

4.3.1 Integritní omezení

Integrita databáze, jinak řečeno konzistence. Relační model dat specifikuje strukturu dat v databázi, ale k tomu, abychom mohli používat databázi jako zdroj dat, je nutné zajistit, aby se do ní dostali jen data, která tam patří a neztratila se data, která nemají. Také je potřeba k tomu mít určité mechanizmy. Těmito mechanizmy jsou integritní omezení.

A databáze, respektive data, jsou konzistentní, pokud jsou ve stavu a vyhovující integritním omezením. To znamená, že se žádnou úpravou nebo smazáním dat neztratila nebo nepoškodila data nebo že v DB nejsou data, která tam nepatří například seznam kontaktů smazaného uživatele atd. Proto existují integritní omezení, která mají podobným nehodám zabránit. Vzhledem k tomu, že k porušení integrity databáze může dojít několika způsoby, rozeznáváme několik druhů integritních omezení [19].

- **Entitní integritní omezení:** Povinné integritní omezení, které zajišťuje úplnost primárního klíče tabulky; zamezí uložení dat, která neobsahují všechna pole sdružená do klíče, nebo data, jež by v těchto polích byla stejná jako v nějakém jiném, již zapsaném, řádku tabulky.
- **Doménová integritní omezení:** Zajišťují dodržování datových typů/domén definovaných u sloupců databázové tabulky.
- **Referenční integritní omezení:** Zabývají se vztahy dvou tabulek, kde je jejich relace určena vazbou primárního a cizího klíče.
- **Aktivní referenční integrita:** Definuje činnosti, které databázový systém provede, pokud jsou porušena některá pravidla [20].

4.3.2 Vazby mezi tabulkami

Pokud existují vztahy[9] (relace) mezi tabulkami (entitami[10]), pak u těchto vztahů definujeme dvě základní vlastnosti [18]:

- Kardinalitu
- Parcialitu

Kardinalita

Vyjadřuje skutečnost, kolik (jeden či mnoho) řádků jedné tabulky může vstoupit do vztahu s kolika řádky druhé tabulky. Existují tři typy vztahů:

- **1:1** Jednomu řádku v jedné tabulce odpovídá právě jeden řádek v tabulce druhé.
  Př.: Manžel má jednu manželku a manželka má jednoho manžela.

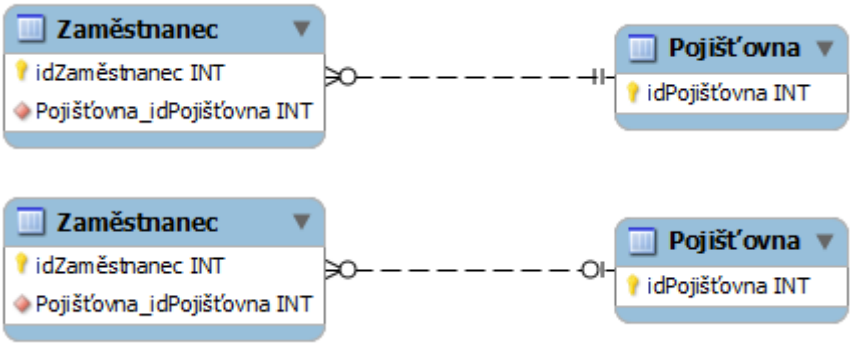
[9] Vztah - je vazba mezi dvěma nebo více entitami.
[10] Entita - představuje nezávisle existující objekt reálného světa.

- **1:N** Jednomu řádku v jedné tabulce může odpovídat více řádků v tabulce druhé.
  Př.: Člověk náleží k jedné zdravotní pojišťovně a zdravotní pojišťovna má v evidenci více lidí.
- **M:N** Mnoha řádkům v jedné tabulce může odpovídat mnoho řádků v tabulce druhé. V relačních databázích není možné modelovat takový vztah přímo, modelují se pomocí mezilehlé tabulky.
  Př.: Jeden zákazník si může koupit několik výrobků a zároveň každý výrobek si může koupit několik zákazníků [18].

Parcialita

Parcialita vyjadřuje povinnost či nepovinnost existence ve vztahu.

- **jednostranně parciální:** Znamená, že například zaměstnanec musí náležet k jedné pojišťovně, pojišťovna však nemusí mít v evidenci ani jednoho zaměstnance (ale může jich mít i více).
- **oboustranně parciální:** Vyjadřuje, že zaměstnanec nemusí náležet k žádné (může náležet k jediné) zdravotní pojišťovně a zdravotní pojišťovna nemusí mít v evidenci ani jednoho zaměstnance [18].



Obrázek 2 - jednostranná, oboustranná parcialita

4.3.3 Normální formy

Normální formy tabulek se používají pro lepší (systematické) návrhy databázových systémů pro efektivní ukládání dat a minimalizaci redundancí při zachování integrity a konzistence dat.

Obecně platí, že čím je tabulka ve vyšší normální formě, tím kvalitněji je tabulka navržena [18].

- **Nultá normální forma (0NF)** - tabulka v nulté normální formě obsahuje alespoň jeden sloupec (atribut), který může obsahovat více druhů hodnot.
- **První normální forma (1NF)** - tabulka je v první normální formě, pokud všechny sloupce (atributy) nelze dále dělit na části nesoucí nějakou informaci neboli prvky musí být atomické. Jeden sloupec neobsahuje složené hodnoty.

- **Druhá normální forma (2NF)** - tabulka je v druhé normální formě, pokud obsahuje pouze atributy (sloupce), které jsou závislé na celém klíči.
- **Třetí normální forma (3NF)** - tabulka je ve třetí normální formě, pokud neexistují žádné závislosti mezi neklíčovými atributy (sloupci) [20].
- **Boyce-Coddova normální forma (BCNF):** Tabulka je v Boyce-Coddově normální formě, jestliže pro každou netriviální závislost X-->Y platí, že X obsahuje klíč schématu R [18].
- **Čtvrtá normální forma (4NF)** - tabulka je ve čtvrté normální formě, pokud sloupce (atributy) v ní obsažené popisují pouze jeden fakt nebo jednu souvislost.
- **Pátá normální forma (5NF)** - tabulka je v páté normální formě, pokud by se přidáním libovolného nového sloupce (atributu) rozpadla na více tabulek [20].

Praktická část

Praktická část bakalářské práce se bude zabývat popisem důležitých části bakalářské práce s popisem jejich funkcionality.

5 Inzerční systém

5.1 Analýza informačního systému

Cílem bakalářské práce bylo vytvořit webovou aplikaci pro inzerci motocyklů. Systém bude umožňovat vyhledávání inzerátů a prohlížet jednotlivé inzeráty. Návštěvníkům bude umožňovat vytvoření svého osobního účtu pro vkládání inzerátů.

Na internetu se vyskytuje mnoho informačních systémů zaměřených na inzerci motocyklů. Proto v následujících kapitolách budou popsány nejznámější z nich.

5.1.1 Motorkari.cz

Webová aplikace motorkari.cz je projektem firmy MOTOportal s.r.o., sídlící v Praze. Tento portál je na trhu již deset let. Projekt motorkari.cz vnikl v roce 2001 jako databáze motocyklů, rok od roku se rozrůstal o psaní článků, bazar a mnoho dalšího. Není to jen webový časopis či archiv redakčních testů nových modelů motocyklů. Je to také zájmová a sociální komunita téměř 35 tisíc motorkářů, kteří se setkávají v diskusním fóru, vkládají fotky a zkušenosti se svými motocykly do sekce motorkáři, vyměňují si videa nebo zveřejňují své cestopisy a zážitky v sekci cestování. Ve spojení s katalogem firem s nabídkou nových produktů a bazarem pro ojeté i nové stroje, příslušenství a náhradní díly jde o jedinečný a ucelený projekt na českém internetu [25].

Projekt motorkari.cz disponuje rozsáhlou komunitou čtenářů. Před vložením inzerátu se musí každý nový uživatel registrovat. Svůj účet může použít pro vstup do diskusního fóra či psaní cestopisu. Formulář pro vložení údajů předchází výběr značky motocyklu a jeho modelů, což je značně nepraktické, tuto slabinu by vyřešilo použití technologie AJAX[11]. Stejným způsobem je vyřešeno i rychlé vyhledávání inzerátu. Značnou výhodou je zobrazení detailu inzerátu, kde se zobrazí graf vývoje ceny, můžete zjistit i bližší informace o motocyklu z hodnocení uživatele.

5.1.2 Motoinzerce.cz

Webová aplikace motoinzerce.cz je projektem firmy Motostar s.r.o., sídlící v Jilemnici. Tento projekt vznikl v roce 2002 jako součást internetového obchodu motostar.cz.

Oproti projektu motorkari.cz je tento projekt zaměřen pouze na inzerci, hlavní výhodou tohoto projektu je, že uživatel, který chce vložit inzerát, se nemusí registrovat. Po vložení inzerátu mu na e-mail přijde id inzerátu a heslo pro editaci a smazání. Formulář pro

[11] http://cs.wikipedia.org/wiki/AJAX

vkládání inzerátů je rozsáhlý a nepřehledný a na druhou stranu obsahuje technologii AJAX, která usnadňuje uživateli vkládání inzerátů. Disponuje také rychlým vyhledáváním v levém panelu, což usnadňuje prácí při vyhledávání inzerátů. Jako soukromá osoba můžete vložit naráz maximálně deset inzerátů což je dostačující.

5.1.3 Shrnutí

Popsané informační systémy v předchozí kapitole jsou velmi propracované a složité aplikace, které jsou vyvíjeny týmem odborníků po dobu několika let. Z tohoto důvodu informační systém pro inzerci motocyklů nedosahuje takové kvality, ale i přesto se některými prvky vyrovná velkým informačním systémům.

5.2 Role uživatelů

Informační systém pro inzerci motocyklů rozeznává tři typy uživatelů. Neregistrovaný uživatel má možnost registrace, vyhledávání inzerátů, zobrazení detailu konkrétního inzerátu.

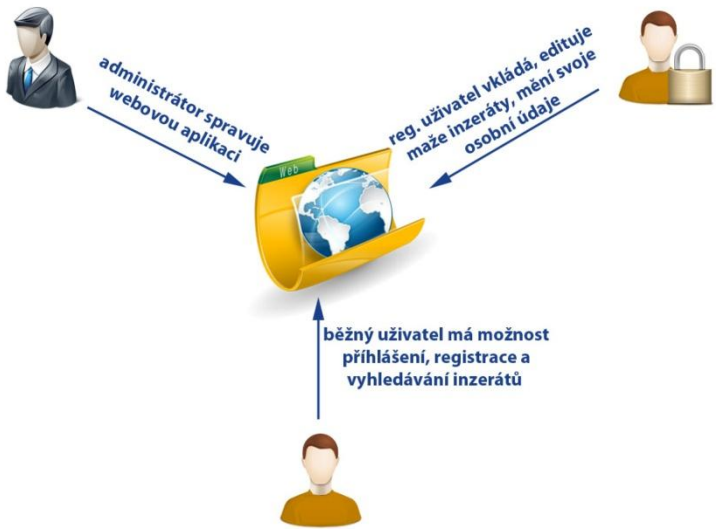
Dalším typem uživatele je registrovaný uživatel. Po zadání správného hesla mu je zpřístupněna položka v horizontálním menu „vlož inzerát“, kde může vkládat inzeráty. Dále mu je zpřístupněna položka „můj účet“, kde jsou zobrazeny údaje o jeho účtu. Má zde možnost změnit registrační údaje a heslo. Jsou zde také zobrazeny vložené inzeráty, které může editovat, mazat a zobrazit detail inzerátu.

Posledním typem uživatele informačního systému je administrátor, který po zadání hesla má možnost smazat jakýkoliv inzerát, smazat účet registrovaného uživatele a jeho inzeráty. Další funkce, kterou má k dispozici je promazání inzerátů dle časového intervalu.

Obrázek 3 - use case diagram

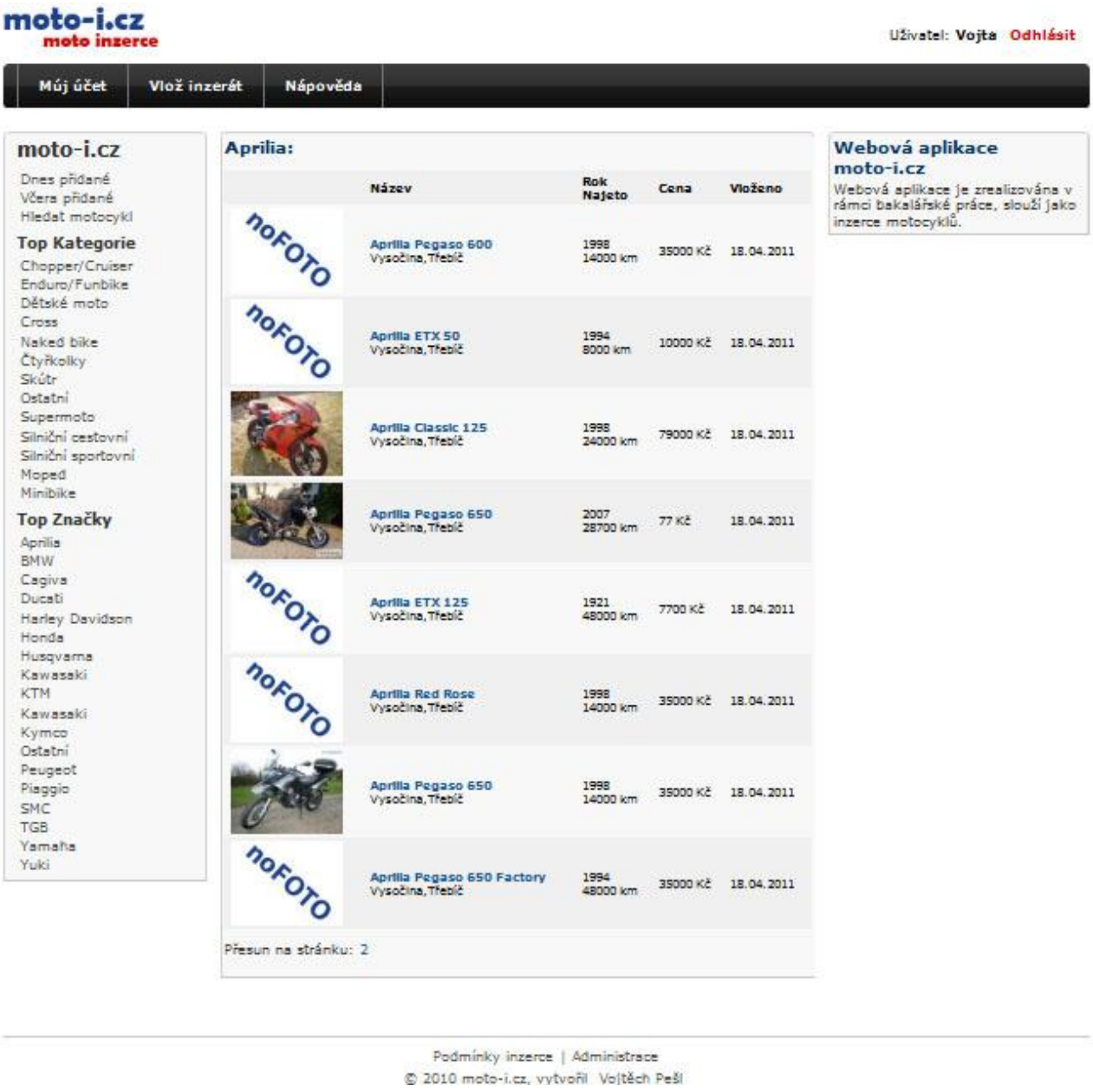
5.3 Rich picture diagram

Tento diagram vyjadřuje základní problematiku a situace, které mohou nastat v informačním systému. Pomocí jednoduchých diagramů zachycuje uživatelské role a činnosti inzerčního systému.



Obrázek 4 - rich picture diagram

5.4 Vzhled aplikace



Obrázek 5 - vzhled aplikace

Layout aplikace je navržen jako třísloupcový. Obsahuje klasické prvky webu, v pravém horním rohu obsahuje navigaci, která slouží k logování uživatele případně k registraci, horizontální menu, levý sloupec, pro rychlé vyhledávání inzerátů. Prostřední sloupec tvoří obsah zvolené stránky. Pravý panel se skládá z informačních bannerů. V spodní části layoutu je patička webu, obsahující hlavní informace o webu.

5.5 Použité technologie

Při realizaci inzerčního systému byly použity technologie, které jsou popsány v úvodních kapitolách. Jako značkovací jazyk bylo vybráno HTML, k tvorbě vzhledu a

rozložení webové stránky byly použity kaskádové styly CSS. Na tvorbu aplikační vrstvy byl použit programovací jazyk PHP a jako uložiště dat byla použita databáze MySQL.

Pro zobrazování galerie obrázků byl použit widget LightBox, který je postaven nad knihovnou JAK[12].

5.6 Použité programy a operační systém

Při vývoji webové aplikace byly použity programy, které jsou zcela zdarma nebo pod licencí MSDN AA[13].

K tvorbě a editaci PHP, HTML, CSS a javascriptových souborů byl použit program NetBeans IDE ve verzi 6.9.1. NetBeans je open source projekt s velmi rozsáhlou uživatelskou základnou, rostoucí komunitou vývojářů. Mezi jeho hlavní výhody patří zvýrazňování syntaxe a možnost využití debuggru.

K zprovoznění webového a databázového serveru jsem používal balíček XAMP[14] ve verzi 1.7.4, který v sobě obsahuje XDebug[15]. Aby fungoval debugger v NetBeans, bylo za potřebí, mít správnou verzi XDebug a správně nastavit soubor php.ini.

K návrhu databáze jsem použil program MySQL Wokrbench[16] ve verzi 5.2. Je to grafický modelovací nástroj pro databázi MySQL, umožňuje vytvořit databázi, importovat modely SQL a mnoho dalšího. Pro přístup k databázi jsem také používal webové rozhraní phpMyAdmin, které obsahuje balíček XAMP.

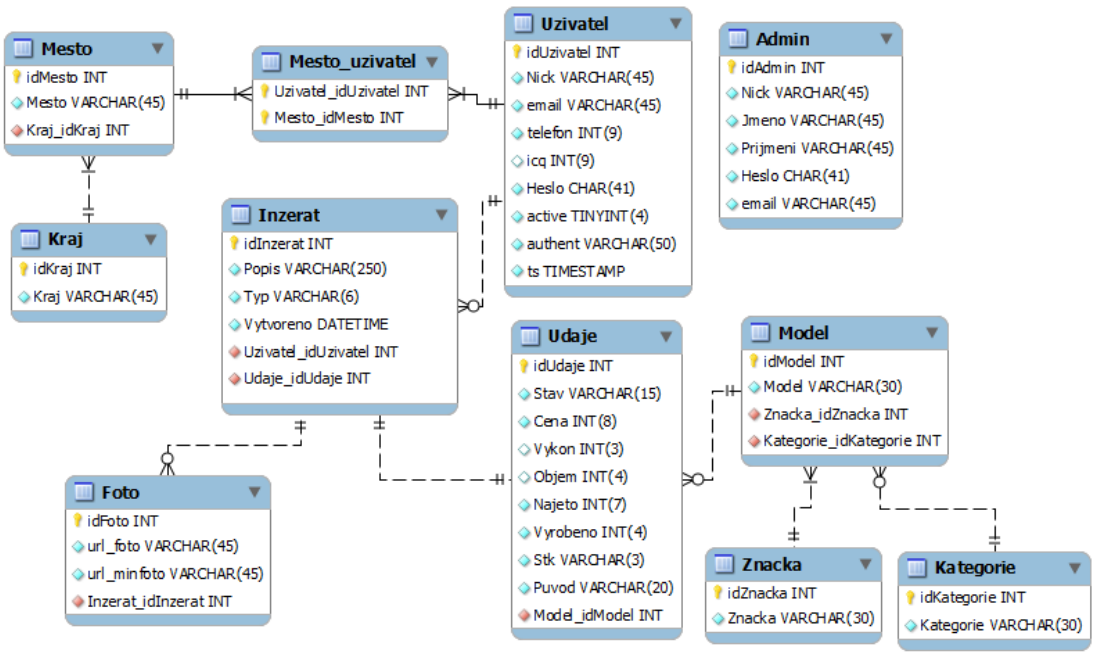
Celá webová aplikace byla vyvíjena pod operačním systémem Windows 7.

[12] JAK je kompaktní a jednoduchý objektově orientovaný framework, usnadňující práci v prostředí jazyka JavaScript
[13] http://www.microsoft.com/cze/education/licence/msdn_academic_alliance/default.mspx
[14] http://www.apachefriends.org/en/xampp.html
[15] http://www.xdebug.org/
[16] http://wb.mysql.com/

6 Návrh databáze

Databáze pro inzerční systém je navržena tak, aby byla zachována integrita a bezpečnost dat.

6.1.1 E-R diagram



Obrázek 6 E-R diagram

6.1.2 Popis tabulek databáze

Tabulka Uživatelů

V tabulce *Uzivatel*, jsou uloženy všechny informace o uživatelích a jejich přihlašovací údaje. Jednomu řádku tabulky uživatel může odpovídat nula až N řádků tabulky inzerát. Pro atribut *Nick* je nastaven index *UNIXUE*, který zabraňuje tomu, aby dva uživatelé měli stejné přihlašovací jméno. Do atributu *ts* se ukládá čas posledního přihlášení. Atribut *authent* obsahuje náhodný řetězec, který se používá při ověřování pomocí e-mailu [17]. V atributu *Heslo* se uchovává hash hesla.

Tabulka 1 - popis tabulky uživatel

atribut	datový typ	popis
IdUzivatel	INT	Primární klíč tabulky
Nick	VARCHAR(45)	Přezdívka uživatele
Email	VARCHAR(45)	Email uživatele
Telefon	INT(9)	Telefon uživatele
Icq	INT(9)	Icq číslo uživatele
Heslo	CHAR(41)	Heslo uživatele
Active	TYNYINT(4)	Aktivita uživatele
Authent	VARCHAR(50)	Náhodný řetězec
Ts	TIMESTAMP	Čas posledního přihlášení

Tabulka Administrátorů

V tabulce *Admin* jsou uloženy všechny informace o Administrátorech a jejich přihlašovací údaje. Pro atribut *Nick* je nastaven index *UNIXUE*, který zabraňuje tomu, aby dva uživatelé měli stejné přihlašovací jméno. V atributu *Heslo* se uchovává hash hesla.

Tabulka 2 - popis tabulky admin

atribut	datový typ	popis
idAdmin	INT	Primární klíč tabulky
Nick	VARCHAR(45)	Přezdívka administrátora
Jméno	VARCHAR(45)	Jméno administrátora
Příjmení	VARCHAR(45)	Příjmení administrátora
Heslo	CHAR(41)	Heslo administrátora
Email	VARCHAR(45)	Email administrátora

Tabulka Měst

V tabulce *Města* jsou uloženy názvy všech větších měst, každého kraje. Jednomu řádku tabulky město musí odpovídat vždy jeden řádek tabulky kraj.

Tabulka 3 - popis tabulky město

atribut	datový typ	popis
IdMesta	INT	Primární klíč tabulky
Mesto	VARCHAR(45)	Název města
Kraj_idKraj	INT	Cizí klíč tabulky kraj

Tabulka Krajů

V tabulce *Kraj* jsou uloženy názvy všech krajů ČR. Jednomu řádku tabulky kraj musí odpovídat vždy alespoň jeden řádek z tabulky město.

Tabulka 4 - popis tabulky kraj

atribut	datový typ	popis
idKraj	INT	Primární klíč tabulky
Kraj	VARCHAR(45)	Název kraje

Tabulka Inzerátů

V tabulce *Inzerat* jsou uloženy hlavní údaje inzerátu. Atribut *Vytvoreno* uchovává datum a čas vzniku inzerátu. Jednomu řádku tabulky inzerát může odpovídat nula až N řádků tabulky foto, zároveň mu odpovídá vždy jeden řádek tabulky údaje.

Tabulka 5 - popis tabulky inzerát

atribut	datový typ	popis
idInzerat	INT	Primární klíč tabulky
Popis	VARCHAR(250)	Popis inzerátu
Typ	VARCHAR(6)	Typ inzerátu
Vytvoreno	DATETIME	Datum a čas vytvoření inzerátu
Uzivatel_idUzivatel	INT	Cizí klíč tabulky uživatel
Udaje_idUdaje	INT	Cizí klíč tabulky údaje

Tabulka údajů

V tabulce *Udaje* jsou uloženy podrobné údaje o motocyklu. Jednomu řádku tabulky údaje odpovídá vždy jeden řádek tabulky inzerát, zároveň mu odpovídá vždy jeden řádek tabulky model.

Tabulka 6 - popis tabulky údaje

atribut	datový typ	popis
idUdaje	INT	Primární klíč tabulky
Stav	VARCHAR(15)	Stav motocyklu
Cena	INT(8)	Cena motocyklu
Vykon	INT(3)	Výkon motocyklu
Obejm	INT(4)	Objem motocyklu
Najeto	INT(7)	Počet najetých km
Vyrobeno	INT(4)	Rok výroby
Stk	VARCHAR(3)	STK motocyklu
Puvod	VARCHAR(20)	Země původu motocyklu
Model_idModel	INT	Cizí klíč tabulky model

Tabulka Modelů

V tabulce *Model*, jsou uloženy názvy modelů jednotlivých značek. Jednomu řádku tabulky model může odpovídat nula až N řádků tabulky údaje, zároveň mu odpovídá vždy jeden řádek tabulky značka a kategorie.

Tabulka 7 - popis tabulky model

atribut	datový typ	popis
idModel	INT	Primární klíč tabulky
Model	VARCHAR(30)	Název modelu
Znacka_idZnacka	INT	Cizí klíč tabulky značka
Kategorie_idKategorie	INT	Cizí klíč tabulky kategorie

Tabulka Značek

V tabulce *Znacka* jsou uloženy značky motocyklů. Jednomu řádku tabulky značka odpovídá vždy jeden až N řádků tabulky model.

Tabulka 8 - popis tabulky značka

atribut	datový typ	popis
idZnacka	INT	Primární klíč tabulky
Znacka	VARCHAR(30)	Název značky

Tabulka Kategorií

V tabulce *Kategorie* jsou uloženy kategorie motocyklů. Jednomu řádku tabulky kategorie může odpovídat nula až N řádků tabulky model.

atribut	datový typ	popis
IdKategorie	INT	Primární klíč tabulky
Kategorie	VARCHAR(30)	Název kategorie

Tabulka Fotografií

V tabulce *Foto* jsou uloženy odkazy na fotografie a jejich miniatury. Jednomu řádku tabulky foto odpovídá vždy jeden řádek tabulky inzerát.

Tabulka 9 - popis tabulky foto

atribut	Datový typ	Popis
idFoto	INT	Primární klíč tabulky
url_foto	VARCHAR(45)	Odkaz na fotografii
url_minfoto	VARCHAR(45)	Odkaz na miniaturu fotografie
Inzerat_idInzerat	INT	Cizí klíč tabulky inzerát

7 Návrh webové aplikace

7.1.1 Přístup k databázi

Všechny přístupy k databázi se provádí přes objekt třídy *mydb*. Tato třída zabraňuje výskytu nadbytečného zdrojového kódu a je velmi praktická i při vyhledávání chyb. Prakticky všechny soubory PHP projektu začínají tím, že se nejprve načítají soubor *mydb.php*, pak vytvoří objekt *mydb* a uloží do proměnné *db* [17].

```
$db = new mydb();
```

7.1.2 Zpracování dat z formulářů

Data z formulářů se zpracovávají, vyhodnocují či ukládají prakticky u všech databázových aplikací.

Velký význam při vyhodnocování těchto dat má sice nepatrná, ale o to nepostradatelná funkce *array_item*, která je nadefinována v souboru *mainfunction.php*. Příkaz *array_item($x, „abc“)* vrací výraz *$x[„abc“]* pouze v případě, pokud tento prvek existuje. V opačném případě vrací hodnotu *FALSE*. Vzhledem k tomu, že není vždy předem jasné, které prvky všechna políčka obsahují, ušetří příkaz array_item čtení obtěžujících varovných hlášení, že prvek pole neexistuje, popřípadě varovné hlášení o tom, že dané políčko vůbec neexistuje [17].

```
// otestování, zda položka pole existuje. Pokud ano, tak se vrátí
function array_item($ar, $key) {
  if(is_array($ar) && array_key_exists($key, $ar))
    return($ar[$key]);
  else
    return FALSE;
}
```

Dále pokud se budou zpracovávat data z formuláře, nachází se v poli *formdata*, u jejichž prvků byly odstraněny případně se vyskytující *Magic Quotes* [17].

```
$formdata = array_item($_POST, "form");
if(is_array($formdata)) {
  // vyhnutí se magic quotes
  if(get_magic_quotes_gpc())
    while($i = each($formdata))
      $formdata[$i[0]] = stripslashes($i[1]);
}
```

V závislosti na tom, zda bylo zmáčknuté tlačítko formuláře, se zobrazí prázdný formulář nebo proběhne validace zadaných dat do formuláře.

```
if(array_item($formdata, "btnSave")){
      $errormsg = "";
```

```
      if(validate_data($formdata, $errormsg, $db)) {


      }
      $new_account = TRUE;
      redirect(baseurl(), 15);
}
else if($new_account==FALSE) show_form1($formdata, $db);
```

Funkce *validate* je použita například u registrace uživatele. Funkce provádí kontrolu správnosti dat zadaných v poli *formdata*, nekontrolují se zde jenom zadaná data z formuláře, ale i zda neexistuje uživatelské jméno v databázi či e-mail. V závislosti na výsledcích pak funkce vrací *TRUE* nebo *FALSE* [17].

```
function validate_data($formdata, &$errormsg, $db) {
  $errormsg = "";
  $username = array_item($formdata, "username");
  $pw1      = array_item($formdata, "password1");
  $pw2      = array_item($formdata, "password2");
  $email    = array_item($formdata, "email");
  $telefon  = array_item($formdata, "telefon");
  $region   = array_item($formdata, "region");
  $city     = array_item($formdata, "city");
  // uživatelské jméno
  if(!$username)
    $errormsg .= "Zadejte, prosím, své uživatelské jméno! <br>";
  else {
    $sql = "SELECT COUNT(*) FROM Uzivatel " .
           "WHERE Nick=" . $db->sql_string($username) .
           "LIMIT 1";
    if($db->querySingleItem($sql)>0)
      $errormsg .= "Vámi vybrané uživatelské jméno je již obsazeno.
                    Vyberte, prosím, jiné! <br>";
  }

  // heslo
  if(!$pw1 || !$pw2 || $pw1!=$pw2)
    $errormsg .= "Zadejte, prosím, do následujících dvou políček
                  dvakrát stejné heslo! <br>";
  // e-mailová adresa
  $regex = '/[A-Z0-9.%_-]+@[A-Z0-9._%-]+/i';
  if(strlen(trim($email))<7 || !preg_match($regex, $email)) {
```

```
    $errormsg .= "Zadejte, prosím, platnou e-mailovou adresu! <br>";
  }
  else{
    $sql="SELECT COUNT(*) FROM UZIVATEL WHERE email=". $db-
>sql_string($email);
    if($db->querySingleItem($sql)>0)
    $errormsg .= "Vámi vybraný e-mail je již obsazen. <br />";
  }
      //Zde se vyskytují další podmínky validace
      //Zde se vyskytují další podmínky validace
 // pokud se nevyskytla žádná chyba
  if($errormsg != "")
    return FALSE;
  else
    return TRUE;
}
```

7.1.3 Transakce

Z pohledu databáze je zřejmě nejzajímavější funkcí této aplikace funkce *save_data*. Ukládá data obsažená v poli *formdata* do tabulek databáze. Pro ukládaní dat se používá buď příkaz *INSERT* nebo příkaz *UPDATE*. Pokud se během ukládání nevyskytne žádná chyba, vrátí funkce *TRUE*. Vrácená hodnota se pak používá k tomu, aby se mohla potvrdit, popřípadě stornovat transakce spuštěná před zavoláním této funkce [17].

```
$db->execute("START TRANSACTION");
if(save_data($formdata, $db))
      $db->execute("COMMIT");
else{
      $db->execute("ROLLBACK");
      echo "<p>Chyba databáze!</p>\n";
}
```

7.1.4 Registrace uživatele

Uložení dat o uživateli

Pokud se během formálních testů ve funkci *valide_data* nevyskytla žádná chyba, pak se data o uživateli uloží ve funkci *sava_data*. U příkazu *INSERT* se ve sloupci *authent* tabulky uživatel uloží náhodné číslo vytvořené pomocí *mt_rand()*. Heslo se zašifruje pomocí funkce SQL s názvem *PASSWORD()*. Výsledný řetězec obsahuje 41 znaků a je šifrován tak, že je sice možné správnost hesla zkontrolovat, ale nelze je dešifrovat. Sloupec *active* se v příkazu *INSERT* nevyskytuje, takže nový záznam dostane přidělenou standardní hodnotu 0 [17].

```
function save_data($formdata, $db) {
  // uložení uživatele
  $authent  = mt_rand();
  $username = $db->sql_string(array_item($formdata, "username"));
  $password = $db->sql_string(array_item($formdata, "password1"));
  $email    = $db->sql_string(array_item($formdata, "email"));
  $telefon  = $db->sql_string(array_item($formdata, "telefon"));
  $region   = $db->sql_string(array_item($formdata, "region"));
  $city     = $db->sql_string(array_item($formdata, "city"));
  $icq      = $db->sql_string(array_item($formdata, "icq"));
  $sql = "INSERT INTO Uzivatel (`Nick`, `email`, `telefon`, `icq`,
`Heslo`, `authent`) " .
         "VALUES ($username, $email, $telefon, $icq, PASSWORD($password),
'$authent')";
  if(!$db->execute($sql))
    return FALSE;
  $userID = $db->insertId();
  $sql = "SELECT idMesto FROM MESTO WHERE Mesto=$city";
  if (!$cityID = $db->querySingleItem($sql))
    return FALSE;
  $sql = "INSERT INTO Mesto_uzivatel (`Uzivatel_idUzivatel`,
`Mesto_idMesto`)" .
         "VALUES ($userID, $cityID)";
  if(!$db->execute($sql))
    return FALSE;
  return TRUE;
}
```

Pomocí PHP funkce *mail* odešleme novému uživateli e-mail, který bude obsahovat odkaz v této podobě [17]:

```
http://localhost/moto_inzerce/authent.php?userID=12&authent=2141719562
```

Vzhledem k tomu, že webová aplikace je tvořena na localhostu, je odkaz pouze zobrazen.

```
  $subject = "Vítejte ve motoinzerci moto-i.cz!";
  $msg = "Aktivujte si, prosím, svůj nový účet\r\n" .
         "ve motoinzerci tím, že do 24 hodin\r\n" .
         "klepněte na níže uvedený internetový odkaz!\r\n\r\n";
  mail(array_item($formdata, "email"),
       $subject,
       $msg . baseurl() . "/authent.php?userID=$userID&authent=$authent",
```

```
      "From: admin@moto-i.cz");
  echo "<p><b>Následující text slouží pouze k testovacím účelům,
       aby se dalo ověřit, že systém pro přihlašování funguje správně,
       když nelze z lokálního systému odesílat e-maily.</b></p>\n";
  echo "<p>$msg<br />",
       build_href(baseurl() . "/authent.php",
                  "userID=$userID&authent=$authent",
                  "Aktivovat účet v motoinzerci"),
       "</p>\n";
```

Uživatelé, kteří do 24 hodin svůj účet neaktivují, se příkazem *DELETE* z tabulky uživatel odstraní. Zabrání se tím tomu, aby se v tabulce uživatel nehromadily neplatné přihlašovací údaje a zbytečně tak neblokovala uživatelská jména [17].

```
  $sql = "DELETE FROM Uzivatel
          WHERE active=0
          AND   ts<DATE_SUB(NOW(), INTERVAL 1 day)";
  $db->execute($sql);
```

Ověření a aktivace pomocí e-mailu

Jakmile uživatel zadá vygenerovanou adresu při registraci do internetového prohlížeče, předají se parametry *userID* a *authent* na stránku *authent.php*. Skript nejprve aktivuje relace a automatický převod adresy URL, pokud počítač klienta nepodporuje soubory *cookiessession.use_trans_sid*. Pomocí jednoduchého dotazu *SELECT* pak zjistíme, zda údaje *userID* a *authent* souhlasí [17].

```
ini_set('session.use_trans_sid', 1);
session_start();
$db = new mydb();
$username = FALSE;
$userID   = array_item($_REQUEST, 'userID');
$authent  = array_item($_REQUEST, 'authent');
$sql = "SELECT Nick FROM Uzivatel " .
       "WHERE idUzivatel=$userID ".
       "AND authent='$authent' " .
       "AND active=0 LIMIT 2";
$rows = $db->queryObjectArray($sql);
```

Pokud souhlasí, pak se pomocí příkazu UPDATE povolí přihlášení uživatele. Údaje pro přihlášení se uloží do proměnných relacea a do souboru cookies, přičemž doba platnosti souboru cookie se nastaví na půl roku. Také se vytvoří složka pro nahrávání fotografií [17].

```
if($rows && is_array($rows) && count($rows)==1) {
```

```
  $username = $rows[0]->Nick;
  $sql = "UPDATE Uzivatel SET active=1 WHERE idUzivatel=$userID";
  $db->execute($sql);
  $_SESSION['userID'] = $userID;
  $_SESSION['username'] = $username;
  // platnost je půl roku
  setcookie('motoUserID',  $userID, time()+180*24*60*60);
  setcookie('motoAuthent', $authent, time()+180*24*60*60);
  $dir = basedir('/users/user_'.$userID);
  if (!is_dir($dir)) {
      if (!mkdir_p($dir)) {
           return False;
      }
  }
}
```

7.1.5 Přihlášení do systému

Pokud se chce uživatel přihlásit do systému, musí do formuláře souboru *login.php* zadat svoje uživatelské jméno a heslo. Tento formulář se zobrazí pomocí funkce *show_form1*. Pro zpracování parametrů formuláře se volá funkce *try_to_login*. Tato funkce pomocí příkazu *SELECT* otestuje, zda uživatelské jméno a heslo souhlasí. Pokud ano, pak se příslušným způsobem nastaví proměnné relace cookies a poté se zavolá stránka *index.php* [17].

```
function try_to_login($formdata, $db) {
  $username = array_item($formdata, "username");
  $password = array_item($formdata, "password");
  $sql = "SELECT idUzivatel, authent FROM Uzivatel " .
         "WHERE Nick=" . $db->sql_string($username) . " " .
         "  AND Heslo=PASSWORD(" . $db->sql_string($password) . ") " .
         "  AND active=1 LIMIT 2";
  $rows = $db->queryObjectArray($sql);
  if($rows && is_array($rows) && count($rows)==1) {
    $_SESSION['userID'] = $rows[0]->idUzivatel;
    $_SESSION['username'] = $username;
    // platnost je půl roku
    setcookie('motoUserID',  $rows[0]->idUzivatel, time()+180*24*60*60);
    setcookie('motoAuthent', $rows[0]->authent, time()+180*24*60*60);
    $sid = SID ? '?' . SID : '';
    header("Location: " . baseurl() . "/index.php");
    exit; }
```

```
  else
    return FALSE;
}
```

Od chvíle přihlášení se musí prakticky na každé internetové stránce sledovat, zda jsou nastaveny proměnné relace s názvem *userID* a *username*, popřípadě zda se dá ze souboru cookie načíst číslo *userID*. Za provádění této kontroly zodpovídá funkce *getUserID*, kterou nalezneme v souboru *mainfunction.php*. Funkce se volá z většiny stránek aplikace poté, co se vytvoří připojení k databázi. V nejjednodušším případě se mohou *userID* a *username* načíst přímo z pole *$_SESSION* [17].

```
  ini_set('session.use_trans_sid', 1);
  session_start();
  $username = array_item($_SESSION, 'username');
  $userID   = array_item($_SESSION, 'userID');
  if($username && $userID)
    return;
```

Pokud nejsou k dispozici žádné proměnné relace, načítá skript proměnné typu cookies názvem *motoUserID* a *motoAuthen*. Příkaz *SELECT* pak zjišťuje k nim vhodné uživatelské jméno. Pokud dotaz vrací přesně jeden výsledek, pak se uloží do *username* i do proměnných relace [17].

```
  $userID =  array_item($_COOKIE, 'motoUserID');
  $authent = array_item($_COOKIE, 'motoAuthent');
  $sql = "SELECT Nick FROM Uzivatel " .
         "WHERE idUzivatel  = " . $db->sql_string($userID) . " " .
         "AND   authent = " . $db->sql_string($authent) . " " .
         "LIMIT 2";
  $rows = $db->queryObjectArray($sql);
  if($rows && is_array($rows) && count($rows)==1) {
    $username = $rows[0]->Nick;
    $_SESSION['userID']   = $userID;
    $_SESSION['username'] = $username;  }
  else {
    $userID = FALSE;
    $username = FALSE;  }
```

Odkaz pro odhlášení vás přesměruje na stránku *logout.php*, kde se proměnné typu *SESSION* odstraní a proměnné typu cookies nastaví na hodnotu *FALSE*. Zavoláním funkce *header* vás poté přesměruje na úvodní stránku [17].

```
session_start();
unset($_SESSION['userID']);
```

```
unset($_SESSION['username']);
setcookie('motoUserID',  FALSE, 0);
setcookie('motoAuthent', FALSE, 0);
header("Location: " . baseurl() . "/index.php");
```

7.1.6 Ukládání fotografií

Pro ukládání fotografií je vytvořena speciální třída *ImgImporter*. Tato třída obsahuje proměnné *supported_types* pro uchování formátu fotografií, které mohou být importovány. Další proměnné, které obsahuje, jsou *img* a *thumbnail*. Tyto proměnné obsahují velikost, na kterou se má fotografie a její miniatura normovat. Při vytváření objektu třídy pomocí constructoru tyto proměnné nastavíme.

```
  function __construct() {
    $this->supported_types=array('jpg', 'jpeg', 'png', 'gif');
    $this->img["max_x"]=640;
    $this->img["max_y"]=0;
    $this->thumbnail["max_x"]=100;
    $this->thumbnail["max_y"]=0;
  }
```

Nejdůležitější funkce této třídy je *copyImage*. Pomocí této funkce kopírujeme fotografie do zadané složky. Funkce má tři vstupní parametry, cestu k fotografii a cestu, kam se má nahrát fotografie a její miniatura.

```
function copyImage($old_name, $new_name, $new_name_tmb)
```

Funkce *copyImage* využívá funkce *image_shrink_size* a *image_resiz*. První vrátí rozměry zmenšené fotografie a druhá funkce převzorkuje fotografii na vypočtené rozměry.

7.1.7 Vyhledávání

Webová aplikace umožňuje dva druhy vyhledávání, jednoduché vyhledávání *find.php* pomocí odkazu v levém panelu a složitější vyhledávání *find_form.php* pomocí formuláře. Pokud se při vyhledávání nalezne více než *n* záznamů, zobrazí se prvních *n* záznamů. Pomocí odkazů je pak možné se přesunout na další stránku s výsledky vyhledávání. Výsledky vyhledávání se seřadí podle času vložení inzerátu [17].

Nejprve se vyhodnotí všechny údaje zadané do skriptu. Některé parametry se musí uvnitř URL adresy kódovat, aby se v nich ošetřily případně se vyskytující speciální znaky, které by mohly způsobovat problémy. Kódování se pak opět ruší pomocí funkce *urldecode* [17].

```
$kategorie  = urldecode(array_item($_REQUEST, 'kategorie'));
$znacka     = urldecode(array_item($_REQUEST, 'znacka'));
$day        = urldecode(array_item($_REQUEST, 'day'));
$page       = array_item($_REQUEST, 'page');
```

```
if(!$page || $page<1 || !is_numeric($page)) $page=1;
elseif($page>100) $page=100;
```

Pokud proměnná *$day* obsahuje nějaké hodnoty, zavolají se postupně tři funkce: *build_day_query*, která vytvoří příkaz SQL, funkce *show_titles*, která zobrazí výsledky vyhledávání a nakonec funkce *show_page_links*, která v případě potřeby zobrazí odkazy.

Ve funkci *build_day_query, build_znacka_query, build_kategorie_query* se vytvoří příkaz SQL pro jednotlivá vyhledávání. Nejdůležitější je však část s příkazem *LIMIT* , která nastaví způsob zobrazení výsledků *LIMIT 20,11* znamená, že má příkaz *SELECT* vrátit výsledek vyhledávání od 20 do 30. Tyto funkce jsou použity v závislosti na vstupních údajích [17].

```
function build_day_query($day, $page, $pagesize){
     switch ($day){
        case 'today':
          $sql ="SELECT `idInzerat`, `Vytvoreno` FROM Inzerat WHERE ".
          "DATE(Inzerat.Vytvoreno)=DATE(NOW()) ORDER BY Vytvoreno DESC ";
        break;
        case 'yesterday':
           $sql ="SELECT `idInzerat`, `Vytvoreno` FROM Inzerat WHERE ".
           "DATE(Inzerat.Vytvoreno)=DATE(DATE_SUB(NOW(), INTERVAL 1 DAY))
ORDER BY Vytvoreno DESC ";
           break;
     }
     $sql .= "LIMIT " . (($page-1) * $pagesize) . "," . ($pagesize + 1);
     return $sql;
}
```

Odkazy na další stránky

Způsob zobrazení odkazů na další stránky spočívá v tom, že u každé stránky se otestuje, zda jsou ještě k dispozici nějaké další výsledky (*pagesize+1*). Tento údaj stačí k tomu, aby se vytvořil odkaz na další stránku s výsledky vyhledávání.

Odkazy na další stránky vytváří funkce *show_page_links*. Funkce vyžaduje ke své činnosti čtyři parametry: číslo stránky, na které se nachází, množství výsledků na stránku, počet naposledy nalezených výsledků vyhledávání a seznam parametrů v *query*, které se mají v odkazu na následující (předchozí) stránku předat.

Funkce *show_page_links* nejprve vytváří odkazy na předchozí stránky a poté odkazy na následující stránky. Pro vytvoření odkazů se použije pomocná funkce *build_href* [17].

```
function show_page_links($page, $pagesize, $results, $query) {
```

```
  if(($page==1 && $results<=$pagesize) || $results==0)
    return;
  echo "<p>Přesun na stránku: ";
  if($page>1) {
    for($i=1; $i<$page; $i++)
      echo build_href("find.php", $query . "&page=$i", $i), " ";
    echo "$page "; }
  if($results>$pagesize) {
    $nextpage = $page + 1;
    echo build_href("find.php", $query . "&page=$nextpage", $nextpage);
  }
  echo "</p>\n";
}
```

8 Popis funkcionality

V této části bakalářské práce bude popsána funkcionalita informačního systému pro inzerci motocyklů z pohledu uživatele.

8.1 Nepřihlášený uživatel

Nepřihlášený uživatel má možnost přihlášení či registrace pomocí navigace v pravém horním rohu. Dále může využívat vyhledávání a zobrazení inzerátu. K vyhledávání se dostane pomocí vertikálního menu v levém panelu, zde je i odkaz pro vyhledávání s více kriterii.

Obrázek 7 - nepřihlášený uživatel navigace

Obrázek 8 - vertikální menu

Přihlásit

V této sekci se zobrazí přihlašovací formulář a uživatel má možnost se přihlásit.

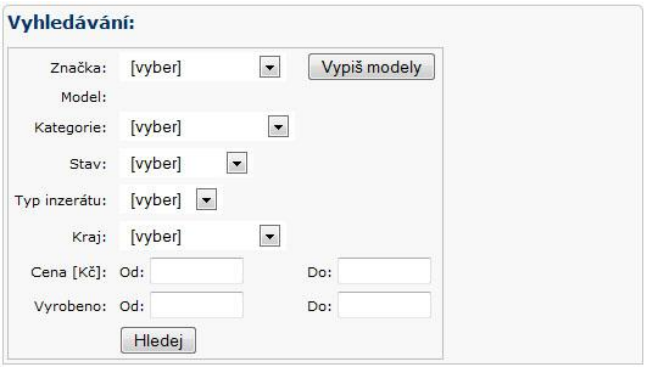
Registrace

V této sekci má uživatel možnost registrace pomocí přehledného registračního formuláře. Po vyplnění povinných údajů, které jsou vyznačeny, musí uživatel provést autorizační proces pomocí odkazu v e-mailové zprávě.

Obrázek 9 - registrační formulář

Vyhledávání s více kriterii

V této sekci se zobrazí uživateli formulář, který slouží k vyplnění kriterií, podle kterých mají být vyhledány inzeráty.



Obrázek 10 - formulář pro vyhledávání

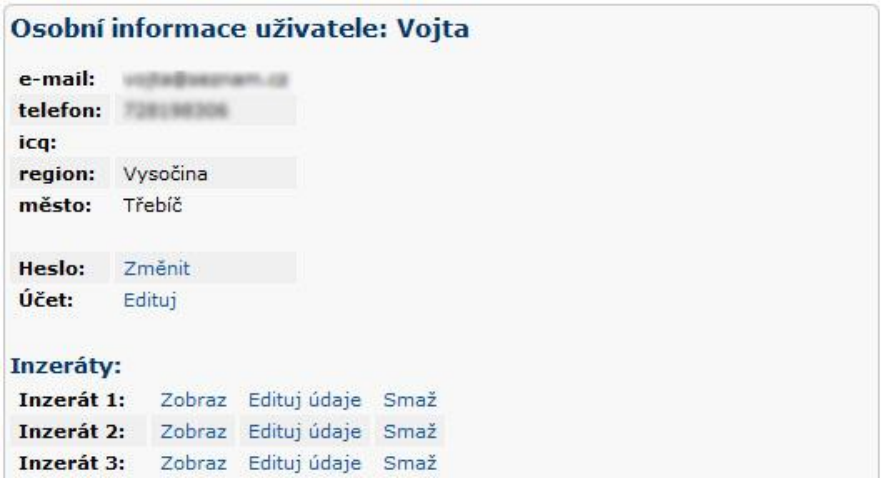
8.2 Přihlášený uživatel

Přihlášený uživatel má přístupné funkce jako nepřihlášený uživatel. Přihlášený uživatel si v informačním systému může zobrazit svůj profil nebo vložit inzerát, do těchto sekcí se dostane pomoci horizontálního menu.

Obrázek 11 - horizontální menu

Profil uživatele

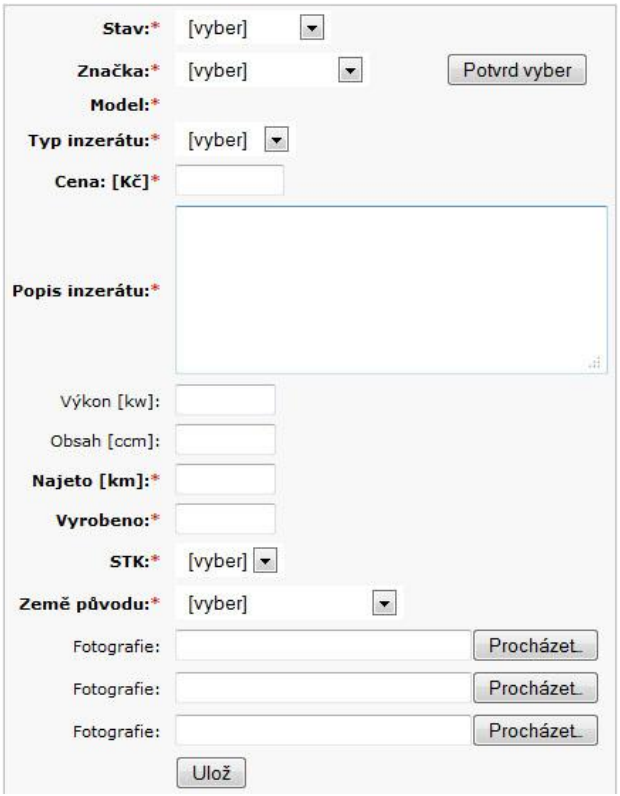
V této sekci si má možnost přihlášený uživatel, zobrazit svůj profil, kde může měnit své údaje, heslo. Je zde také zobrazen výpis všech inzerátů, které muže zobrazit, editovat či smazat.



Obrázek 12 - osobní účet uživatele

Vkládání inzerátu

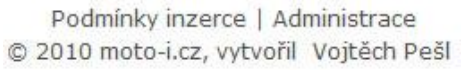
V této sekci může přihlášený uživatel vkládat inzeráty pomocí formuláře. Formulář obsahuje povinná pole, bez kterých nejde inzerát vložit. Uživatel může vložit maximálně tři fotografie.



Obrázek 13 - formulář pro vložení inzerátu

8.3 Administrátor

Administrátor má možnost přihlášení pomocí přihlašovacího formuláře, který může vyvolat pomocí odkazu v patičce stránky, po zadání správného uživatelského jména a hesla se dostane do administrační části. Administrátor má také přístupné funkce jako nepřihlášený uživatel.



Obrázek 14 – patička

Administrace

V této sekci má možnost uživatel smazat jakýkoliv inzerát, smazat účet registrovaného uživatele a jeho inzeráty. Další funkce, kterou má k dispozici je promazání inzerátů dle časového intervalu.

Obrázek 15 - administrace

9 Závěr

Cílem bakalářské práce bylo popsat, analyzovat stávající informační systémy pro inzerci vozidel a vytvořit informační systém s využitím databáze MySQL. Na praktické řešení byly kladeny podmínky, aby umožňoval vkládání inzerátů od registrovaných uživatelů, normoval velikosti fotografií, umožňoval vyhledávání inzerátů podle více kriterií a přístup dle práv.

Byl vytvořen informační systém pro inzerci motocyklů, který splňuje všechny požadavky. Informační systémy pro inzerci vozidel jsou propracované a složité systémy, které vyvíjí tým programátorů a jsou tvořeny i řadu let. Proto vytvořený informační systém nedosahuje takových kvalit jako popisované informační systémy, avšak je funkční a plně využitelný.

Před praktickým nasazením bude do informačního systému přidána metoda ochrany CAPTCHA. V budoucnu bude aplikace rozšířena o modul reklam a možnost zvýraznění inzerátu. Co se týče webových technologií, bylo by vhodné použít více ajaxových technologií a použití některého ze zmiňovaných frameworků.

Tvorba bakalářské práce byla pro mě velkým přínosem, rozšířil jsem si znalost technologií používaných pro tvorbu webových aplikací. Při tvorbě praktické části jsem se obohatil o mnoho programátorských znalostí.

Literatura

[1] World Wide Web. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 13.8.2004, last modified on 18.3.2011 [cit. 2011-04-04]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/World_Wide_Web>.

[2] Webový server. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 27.10.2005, last modified on 29.1.2011 [cit. 2011-04-04]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Webový_server>.

[3] Webová aplikace. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 26.10.2005, last modified on 29.1.2011 [cit. 2011-04-04]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Webová_aplikace>.

[4] *Svět hardware* [online]. c1998, c2011 [cit. 2011-04-04]. Statický web. Dostupné z WWW: <http://www.svethardware.cz/glos.jsp?doc=70EBB0F50FC04C4EC125739F003FC46D>.

[5] *Svět hardware* [online]. c1998, c2011 [cit. 2011-04-04]. Dynamický web. Dostupné z WWW: <http://www.svethardware.cz/glos.jsp?doc=37697B519580355AC125739F003FEE64>.

[6] Značkovací jazyk. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 20.4.2006, last modified on 28.3.2011 [cit. 2011-04-05]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Značkovací_jazyk>.

[7] HTML. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 16. 7. 2004, last modified on 4. 4. 2011 [cit. 2011-04-05]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/HTML>.

[8] KUČERA, Miroslav, et al. *Programování na webu*. Druhé přepracované a rozšířené vydání. Praha : [s.n.], 2003. 600 s. ISBN 80-86593-36-3.

[9] Kaskádové styly. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 28. 10. 2004, last modified on 24. 3. 2011 [cit. 2011-04-05]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Kaskádové_styly>.

[10] PHP. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2. 6. 2004, last modified on 27. 3. 2011 [cit. 2011-04-06]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/PHP>.

[11] STOUPA, Václav. *Root.cz : informace nejen ze světa Linuxu* [online]. 28. 3. 2008 [cit. 2011-04-06]. Přehled a vývoj PHP frameworků. Dostupné z WWW: <http://www.root.cz/clanky/prehled-a-vyvoj-php-frameworku/>.

[12] Javascript. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 29. 3. 2011, last modified on 12. 8. 2004 [cit. 2011-04-06]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Javascript>.

[13] GROFF, James R.; WEINBERG, Paul N. *SQL : kompletní průvodce*. 1 vydání. Brno : Computer Press, 2005. 936 s. ISBN 80-251-0369-2.

[14] Mysql. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 21. 11. 2004, last modified on 23. 3. 2011 [cit. 2011-04-06]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Mysql>.

[15] OLŠAVSKÝ, Marek. *Linuxsoft* [online]. 1.9.2004 [cit. 2011-04-07]. Proč PgSQL, data a relace. Dostupné z WWW: <http://www.linuxsoft.cz/article.php?id_article=354>.

[16] Oracle. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 7. 1. 2005, last modified on 11. 2. 2011 [cit. 2011-04-07]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Oracle>.

[17] KOFLER, Michael; ÖGGL, Bernd. *PHP5 a MySQL5 : Průvodce webového programátora*. Vydání první. Brno : Computer Press, 2007. 608 s. ISBN 978-80-251-1813-9.

[18] ŽÁK, David. *Databázové systémy I – Návrh databáze* [Přednáška]. Pardubice: Univerzita Pardubice. 2008. [cit. 2011-04-07].

[19] *Manualy.net* [online]. 10.9. 2006 [cit. 2011-04-07]. Teorie relačních databází: Integritní omezení. Dostupné z WWW: <http://www.manualy.net/article.php?articleID=15>.

[20] Relační databáze. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 26. 11. 2004, last modified on 26. 3. 2011 [cit. 2011-04-08]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Relační_databáze>.

[21] VRÁNA, Jakub. *PHP triky* [online]. 13.4.2005 [cit. 2011-04-08]. Ukládání hesel. Dostupné z WWW: <http://php.vrana.cz/ukladani-hesel.php>.

[22] *Stoyan* [online]. c2006-2011 [cit. 2011-04-08]. Web Hacking - PHP Injection. Dostupné z WWW: <http://stoyan.cz/hacking-php-injection/>.

[23] *Security-Portal* [online]. 24 Leden, 2005 [cit. 2011-04-08]. SQL Injection. Dostupné z WWW: <http://security-portal.cz/clanky/sql-injection>.

[24] VOJÁČEK, Petr. *Programujte* [online]. 23. 04. 2007 [cit. 2011-04-08]. SQL Injection a zabezpečení. Dostupné z WWW: <http://programujte.com/?akce=clanek&cl=2007041802-sql-injection-a-zabezpeceni>.

[25] *Motorkari.cz* [online]. c2001-2011 [cit. 2011-04-28]. Prezentace serveru Motorkáři.cz. Dostupné z WWW: <http://www.motorkari.cz/statika/?stid=1>.

Příloha A – Ukázka widgetu LightBox

Příloha B – Ukázka zobrazení inzerátu

Aprilia Pegaso 650

Typ:	Prodej	**Kraj:**	Vysočina
Stav:	Nové	**Mesto:**	Třebíč
Cena:	77 Kč	**email:**	[redacted]
Najeto:	28700 Km	**telefon:**	[redacted]
Vyrobeno:	2007		
STK:	Ano		
Puvod:	Česká republika (CZ)		
Výkon:	25 kw		
Objem:	650 ccm		
Popis:	Motor ok. Po výměně filtrů, oleje v tlumičích, motorového oleje (cca. 500 km najeto.) jen jezdit! Nová STK. Servisní knížka, dva klíče, CZ manuál. Vše mohu doložit fakturama. Pravidelný servis, nehavarovaná, bez škrábanců.