Informační systém studijního oddělení

Lukáš Chytílek

Bakalářská práce
2016

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2015/2016

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Lukáš Chytílek**
Osobní číslo:	**A12024**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**prezenční**
Téma práce:	**Informační systém studijního oddělení**
Téma anglicky:	**An Information System for the Student Affairs Department**

Zásady pro vypracování:

1. **Prostudujte předpisy fakulty a univerzity týkající se studijních záležitostí.**
2. **Proveďte rešerši vhodných komerčních a nekomerčních systémů, které by splňovaly požadavky uveřejňování informací řazených do kategorií.**
3. **Navrhněte přehledný systém pro informování o základních úkonech studijního oddělení, včetně členění do kategorií.**
4. **Vytvořte systém, který bude umožňovat přidávání dalších úkonů a informací pomocí formuláře přímo v systému bez nutnosti úpravy systému.**
5. **Věnujte pozornost zabezpečení systému.**

Rozsah bakalářské práce: –

Rozsah příloh: –

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

Seznam odborné literatury:

1. **RAHMEL, Dan. Joomla: Podrobný průvodce tvorbou a správou webů. Brno: Computer Press, a.s., 2010. ISBN 978-80-251-2714-8.**
2. **PROCHÁZKA, David. CSS a XHTML: tvorba dokonalých www stránek krok za krokem. 2. vydání. Praha: GRADA, 2011. Průvodce. ISBN 978-80-247-3897-0.**
3. **KOSEK, Jiří. PHP-tvorba interaktivních internetových aplikací. Grada Publishing, a.s., 1999. 492 s. ISBN 80-7169-373-1.**
4. **HOGAN, Brian P. HTML5 a CSS3: výukový kurz webového vývojáře. Vyd. 1. Brno: Computer Press, 2011, 272 s. ISBN 978-802-5135-761.**
5. **LUBBERS, Peter, Brian ALBERS a Frank SALIM. HTML5: programujeme moderní webové aplikace. Vyd. 1. Brno: Computer Press, 2011, 304 s. ISBN 978-802-5135-396.**

Vedoucí bakalářské práce: **doc. Ing. Jiří Vojtěšek, Ph.D.**
Ústav řízení procesů

Datum zadání bakalářské práce: **19. února 2016**

Termín odevzdání bakalářské práce: **27. května 2016**

Ve Zlíně dne 19. února 2016

L.S.

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl jsem seznámen s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně 12. 5. 2016

.....
podpis diplomanta

ABSTRAKT

Cílem této bakalářské práce je vytvoření informačního systému jako webové aplikace pro studenty, který bude informovat o situacích a postupech, které musí studenti udělat, když si vyřizují studijní záležitosti. Systém bude otevřený a bude do něj možné přidávat další položky. Při řešení bakalářské práce bylo použito jazyků HTML, PHP, CSS a MySQL, techniky ORM a CRUD operací. Informační systém byl testován nástroji proti útokům Cross Site Scripting, SQL injection a dalších. V práci jsem vytvořil systém, který je složený ze dvou částí, frontend, který je viditelný pro návštěvníka webové aplikace a backend, který slouží pro administraci. Přínosem této práce je zobrazit důležité předpisy a informace pro studenty na webové stránce.

Klíčová slova: Informační systém, HTML, CSS, PHP, MySQL, ORM, CRUD, FAI

ABSTRACT

Purpose of this bachelor thesis is creating of informations system as a web application for students, which will inform about situations and procedures which must students do when they have to deal with academic issues. System will be opened and it will be possible to add new items. For solving this work were used HTML, PHP, CSS and MySQL languages, ORM techniques and CRUD operations. Information system was tested by tools like Cross Site Scripting, SQL injection etc. In this work I created a system, which is created from 2 parts. Frontend, which is visivle for visitor of web application and backend, which is used for administration. Benefit of this work is to view important reagulations and informations for students on web page.

Keywords: Information system, HTML, CSS, PHP, MySQL, ORM, CRUD, FAI

Chtěl bych poděkovat vedoucímu své bakalářské práce doc. Ing. Jiřímu Vojtěškovi, Ph.D. za odborné vedení, podnětné připomínky a rady při vypracování práce.

OBSAH

ÚVOD

Informační systém může mít několik podob, jako například účetnictví, seznamy zaměstnanců ve firmě, pro vedení agendy, školní informační systém, manažerské informační systémy atd. Takové informační systémy se dají stáhnout open source, koupit si licenci nebo naprogramovat vlastní informační systém.

Téma bylo navrhnuté vedoucím práce panem Ing. Jiřím Vojtěšek, Ph.D. Důvodem, proč jsem si zvolil téma na vytvoření informačního systému je ten, že se již pár let zajímám o webové technologie a tvorbu webových stránek.

Informační systém studijního oddělení bude sloužit pro zobrazení důležitých předpisů a informacích týkající se studijních záležitostí. Systém je rozdělen na backend a frontend. Backend slouží jako administrace, kde je možné vkládat a upravovat data v systému. Frontend pak tyto data zobrazí návštěvníkovi.

Informační systém je napsán v jazycích HTML, PHP, CSS a MySQL. Struktura a design stránky je napsán v jazycích HTML a CSS, pro skripty a ORM model je napsán v PHP a MySQL.

ORM model neboli Objektově relační mapování se používá pro databázové operace čtení, zápis, úprava a mazání, také jako CRUD operace.

Práce bude určena pro studijní oddělení na Fakultě aplikované informatiky ve Zlíně, za účelem informovat studenty předpisy na fakultě.

I. TEORETICKÁ ČÁST

1 PŘEDPISY FAKULTY A UNIVERZITY

Každá univerzita má své předpisy a normy. Univerzita Tomáše bati má předpisy a normy, které zahrnují studijní a zkušební řád, stipendijní řád a disciplinární řád. Fakulta aplikované informatiky si pak tyto řády doplňují podle sebe.

1.1 Studijní a zkušební řád

Studijní a zkušební řád Univerzity Tomáše Bati ve Zlíně podle § 17 odst. 1 písm. f) zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů, ve znění pozdějších předpisů, stanovuje pravidla studia pro studium v bakalářských, magisterských a doktorských studijních programech. Řád musí dodržovat studenti, kteří jsou zapsání na jeden ze studijních programů fakulty. [1]

Studijní a zkušební řád popisuje, jak je rozvržený akademický rok, jakým způsobem je stanovený studijní plán, jak je jmenována rada studijního programu, kreditový systém, způsob zakončení předmětu, způsob výuky, dokumentace předmětu. Také stanovuje, jakým způsobem jsou ověřovány studijní výsledky, udělení zápočtů, průběh zkoušek, stanovuje podmínky pro pokračování ve studiu, zápis do dalšího roku studia, přerušení studia, podmínky pro řádné ukončení studia.

1.2 Stipendijní řád

Stipendijní řád Univerzity Tomáše Bati ve Zlíně obsahuje pravidla pro přiznávání stipendií studentům ve studijních programech na fakultách UTB.

Všechny stipendia jsou přiznána po projednání stipendijní komisí FAI, která se skládá z šesti členů. Student na základě stipendijního řádu může zažádat o prospěchové stipendium, to se určuje na základě výsledků akademického roku. Výše prospěchového stipendia se určuje podle studijního průměru.

Další možností je získat mimořádné stipendium. Mimořádné stipendium může být přiznáno za vynikající výsledky v průběhu studia, sportovní výsledky, podpora studia atd.

1.3 Disciplinární řád

Disciplinární řád obsahuje pravidla pro disciplinární řízení vůči studentům studijních programů na UTB.

Disciplinární přestupek je porušení povinností stanovených právními předpisy UTB. Přestupkem může být plagiátorství, podvodné jednání, porušení finančních a majetkových zájmů UTB atd. Studentovi může být za porušení svých povinností uloženo napomenutí, podmíněné vyloučení nebo vyloučení ze studia.

Přestupky projednává disciplinární komise fakulty. Disciplinární komise zahajuje a projednává řízení. Rozhodnutí vydává děkan nebo rektor.

2 KOMERČNÍ, NEKOMERČNÍ A VLASTNÍ SYSTÉMY

V dnešní době může mít webové stránky kdokoliv, stačí k tomu mít přístup k internetu. Pro své webové stránky můžeme využít systémy typu blogger.com, kde se stačí pouze zaregistrovat, a můžeme začít vkládat obsah na stránky, v takovém případě není potřeba doména ani webový hosting. Dalším způsobem pro vytvoření vlastních webových stránek je využití tzv. CMS neboli redakčního systému, k tomu je už potřeba mít vlastní doménu a webový hosting.

2.1 Content management system

Content management systém (CMS) neboli Systém pro správu obsahu, který umožňuje správu internetových stránek bez nutnosti znalostí v oblasti tvorby webových stránek.

Využívání CMS snižuje náklady na financování programátora. Obsah webových stránek si tak můžeme kdykoliv a kdekoliv aktualizovat podle sebe. Pro úpravu obsahu webových stránek tak stačí mít pouze internetový prohlížeč a internet.

CMS umožňuje spravovat články, kategorie, obrázky, fotogalerie a komentáře. K úpravě textu se používají WYSIWYG editory, které umožňují formátovat text bez znalostí jazyka HTML.

CMS obsahují také šablony a pluginy, díky kterým je vývoj webové stránky rychlejší.

2.2 Komerční systémy

Komerční systémy jsou systémy, které jsou licencovány pod vlastní firmy a mají uzavřený zdrojový kód. Z tohoto důvodu není možné měnit zdrojový kód ale pouze obsah webové stránky. Změny zdrojového kódu může provádět pouze firma, který systém vlastní. Příkladem může být CMS vBulletin nebo i portál IS/STAG.

2.2.1 vBulletin

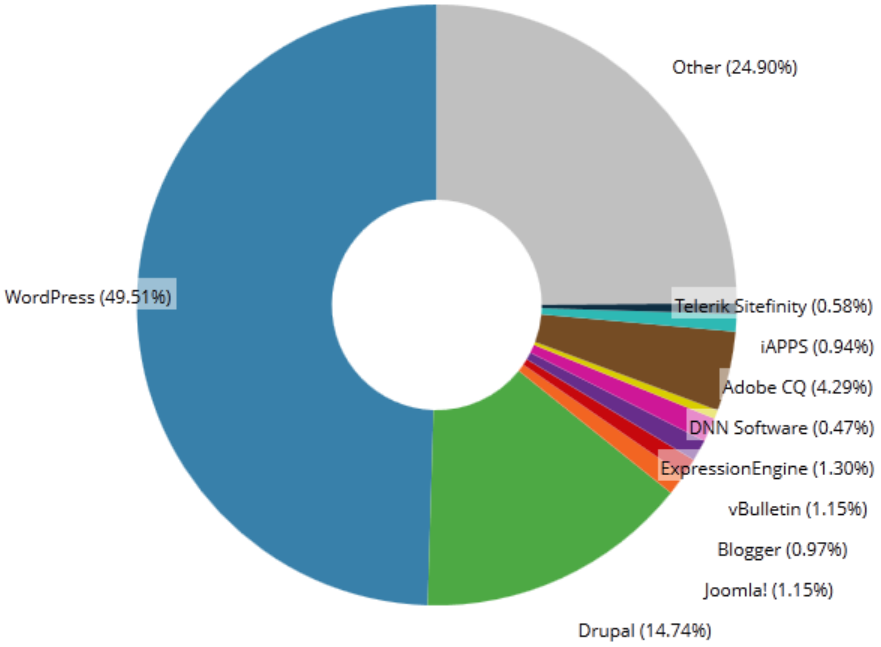
vBulletin je komerční CMS. Tento CMS vznikl v roce 2000 a je založen na platformě PHP a MySQL. [3]

Zpočátku byl systém myšlen pro vytváření diskuzního fóra. Dnes lze použít vBulletin i pro osobní nebo firemní webové stránky.

Ačkoliv je to placený CMS, tak má velkou komunitu (přes 40 000) a nabízí různé doplňky a vzhledy.

2.3 Nekomerční systémy

Nekomerční systémy jsou systémy, které se vyvíjí jako open source. V případě nutnosti je možné upravit kód podle svých představ. Mezi nekomerční systémy řadíme Wordpress, Joomla!, Drupal, Prestashop, TextPattern a další.



Obr. 1. Statistika používání CMS [2]

2.3.1 Wordpress

Wordpress je systém pro správu obsahu neboli CMS. Funguje na PHP a MySQL. Wordpress je open source projekt vyvíjen pod licencí GNU GPL.

Wordpress byl vytvořen v roce 2003, ze začátku byl využíván jako publikační nástroj pro blogy. Od roku 2003 se však Wordpress změnil a v dnešní době běží na Wordpressu i weby, jako jsou katalogové weby, diskuzní fóra, elektronický obchod nebo i sociální sítě. [4]

Wordpress obsahuje vlastní šablony a pluginy, které se díky obrovské komunitě vývojářů rozrůstají.

Výhodou je obrovská komunita vývojářů a návodů k dispozici, bezplatné a placené pluginy.

Nevýhodou může být zabezpečení, které je po standardní instalaci zranitelné vůči útokům.

2.3.2 Joomla!

Joomla! je zdarma systém pro správu obsahu. Joomla! je založená na PHP a využívá databázi MySQL. Joomla! je open source projekt vyvíjen pod licencí GNU GPL.

Joomla! byla vytvořena v roce 2005 a je využívána pro blogy, diskuzní fóra, hlasování, kalendář, rezervační systémy, firemní adresáře, backendové sítě atd. [5]

Joomla! v základu obsahuje základní balíček pluginů a šablon. Pro systém lze najít spoustu pluginů, doplňků, šablon a výukových materiálů.

Systém Jomla! používají společnosti jako: MTV, IHOP nebo také Citibank.

Výhodou je možnost provádět autentizaci pomocí OpenID, Google nebo LDAP, velká uživatelská komunita.

Nevýhodou je uživatelské rozhraní, které oproti CMS Wordpress není tak uživatelsky přívětivé, nehodí se pro velmi jednoduché weby.

2.3.3 Drupal

Drupal je zdarma založený systém pro správu obsahu. Je založený na PHP a využivá databáze MySQL a PostgreSQL. Drupal je open source projekt vyvíjen pod licencí GNU GPL.

Drupal byl vytvořen v 2001, systém je modulární, takže je možné vytvořit blog, e-shop, fórum nebo i korporátní web. [6]

Drupal v základu obsahuje moduly pro vytváření článků, diskusního fóra, blogů, správu komentářů atd. Každý uživatel systému Drupal může vytvořit vlastní modul.

Systém Drupal používají společnosti jako: whitehouse.gov, london.gov.uk, Sony Music, New York Observer a další.

Výhodou je velká podpora komunity, více než 6000 modulů.

Nevýhodou je použitelnost pro jednoduché weby, z toho důvodu se může jevit CMS jako složitý.

2.4 Vlastní systémy

Komerčních a nekomerčních systémů pro správu obsahu na webových stránkách se dá najít mnoho. Může však nastat situace, kdy žádný ze systémů není vhodný pro dané požadavky, například žádny s dostupných systémů neobsahuje moduly nebo pluginy, nebo i šablony,

které by zadaným požadavkům vyhovovaly. V takovém případě se musí naprogramovat redakční systém speciálně na míru podle požadavků zákazníka.

3 ZÁKLADNÍ POJMY

Při tvorbě webových stránek se můžeme setkat s několika pojmy v oblasti webových technologií. Můžeme se setkat s programovacími jazyky, databázovými jazyky nebo prostředky pro zobrazení webové stránky.

3.1 Programovací jazyky

Pro vytvoření webových stránek je potřeba znát alespoň základní strukturu a syntaxe. Programovací jazyk pro webové stránky je HTML, CSS a PHP.

3.1.1 HTML

HMTL neboli „HyperText Markup Language“ je značkovací jazyk pro hypertex. Je to jeden z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci dokumentů na internetu. [8]

HTML dokument je složen z textu, který je umístěn mezi HTML značky, které se na webové stránce nezobrazí, ale pouze určují, jak bude text zobrazen. HTML dokument začíná značkou <*html*> a končí </*html*>. HTML dokument je pak dále rozdělen na sekci <*head*></*head*> a <*body*></*body*>.

Do HTML značky <*head*> se vkládají informace o stránce, tzv. meta tagy, do kterých lze napsat titulek, popis, klíčové slova stránky, nebo také i autor. Tyto informace se na stránce nezobrazují.

Pod HTML značku <*body*> se již vkládá text, který chceme v HTML dokumentu zobrazit. V <body> se můžou objevit značky pro nadpis, který se značí šesti úrovněmi <*h1*> až <*h6*>, odstavec <*p*>, odkaz <*a*>, obrázek <*img*> atd.

3.1.2 CSS

CSS(Cascading Styling Sheets) je kód, který určuje design stránky. Určuje atributy prvků na stránce, jako je zarovnání prvku, formátování textu atd. Lze tak tedy jednoduše „nastylovat“ stránku. [7]

CSS je nejlepší mít v samostatném souboru, ale lze jej i vložit přímo do HTML hlavičky <*head*> pomocí značky <*style*>. Pomocí CSS stylujeme značky nebo vlastní třídy a identifikátory. Pokud chceme nastylovat nějakou HTML značku, např. odstavec <*p*> a chceme změnit barvu písma na červenou, uděláme to takhle: *p {color: red;}*, při stylování se začíná

HTML značkou, třídou nebo identifikátorem. Pokud jde o třídu, musí se napsat před názvem třídy tečka, např.: .mujstyl{color: red;}, pro identifikátor se píše hash, např.: #mujstyl{color: red;}.

3.1.3 PHP

PHP je programovací jazyk, pomocí kterého se dají programovat stránky na straně serveru. Což znamená, že na straně uživatele se zadávají údaje, na straně serveru se provede naprogramovaný algoritmus a uživateli se zobrazí pouze výsledek. To pomáhá vytvářet interaktivní stránky.

PHP kód se vkládá mezi značky *<?php zde je nějaký php kód ?>* a musí být uložený v souboru .php. PHP nelze vkládat do HTML souboru ale HTML lze vložit do PHP souboru. Každá funkce musí být ukončena středníkem „ ; “.

V PHP se deklarují proměnné symbolem $ neboli znakem dolar. Oproti jazyku C, C# nebo Java se nemusí při deklaraci proměnné psát typ, stačí tedy pouze zápis *$novaPromenna = 10;*. Pokud se dosazuje za proměnou text, musí být v uvozovkách.

PHP obsahuje stejně jako jiné programovací jazyky větvení, cykly, pole, práce se soubory, vytváření vlastních funkcí, tříd atd.

3.2 Databázové jazyky

Pro vytváření dynamických webových stránek se používá navíc databáze, která je složena z tabulek a dat, které jsou v tabulkách uloženy.

3.2.1 MySQL

Databázový systém, se kterým se pracuje pomocí jazyka SQL. Může obsahovat například uživateli vložená data, nebo data vložená samotným administrátorem, která pak může zobrazovat na stránkách např. za pomocí PHP.

V MySQL si vytvoříme databázi a do každé databáze vložíme tabulky. Každá tabulka by měla mít svůj primární klíč, nejčastěji se používá u sloupce, který nebude mít nikdy stejné data, příkladem může být id uživatele v tabulce. Pomocí funkce *AUTO INCREMENT* se nemusíme o tzv. id uživatele starat a při vkládání nového uživatele se id číslo zvětší o jedna. Ošetření, aby se nevyskytly v databázi např. stejné emailové adresy, je dobré využít vlastnost *UNIQUE*, která nedovolí vložit do tabulky email, který se v tabulce již nachází.

3.2.2 SQL

Používá se k práci s MySQL. Pomocí příkazů se např. dají přidávat data do databáze, měnit její obsah, zobrazovat databázi apod. [9] SQL obsahuje příkazy pro manipulaci s daty, pro definici dat, řízení dat a ostatní.

Jakmile máme databázi vytvořenou a potřebujeme ji naplnit daty, použijeme příkaz *INSERT*, např.: *INSERT INTO uzivatel (jmeno,prijmeni,email,heslo) VALUES ('Jan','Novák','novak@mail.cz', 'heslo123';*. Chceme-li data z tabulky vybrat, slouží k tomu příkaz *SELECT*, např.: *SELECT * FROM uzivatel.* Pro úpravu tabulky použijeme příkaz *UPDATE*, např.: *UPDATE uzivatel SET jmeno = 'Tomáš', prijmeni = 'Novák', heslo = 'noveheslo123' WHERE email = 'novak@mail.cz'*. Pro smazání dat z tabulky použijeme příkaz *DELETE*, např.: *DELETE FROM uzivatel WHERE email = 'novak@mail.cz'*.

3.3 Prostředky pro webové stránky

3.3.1 Server

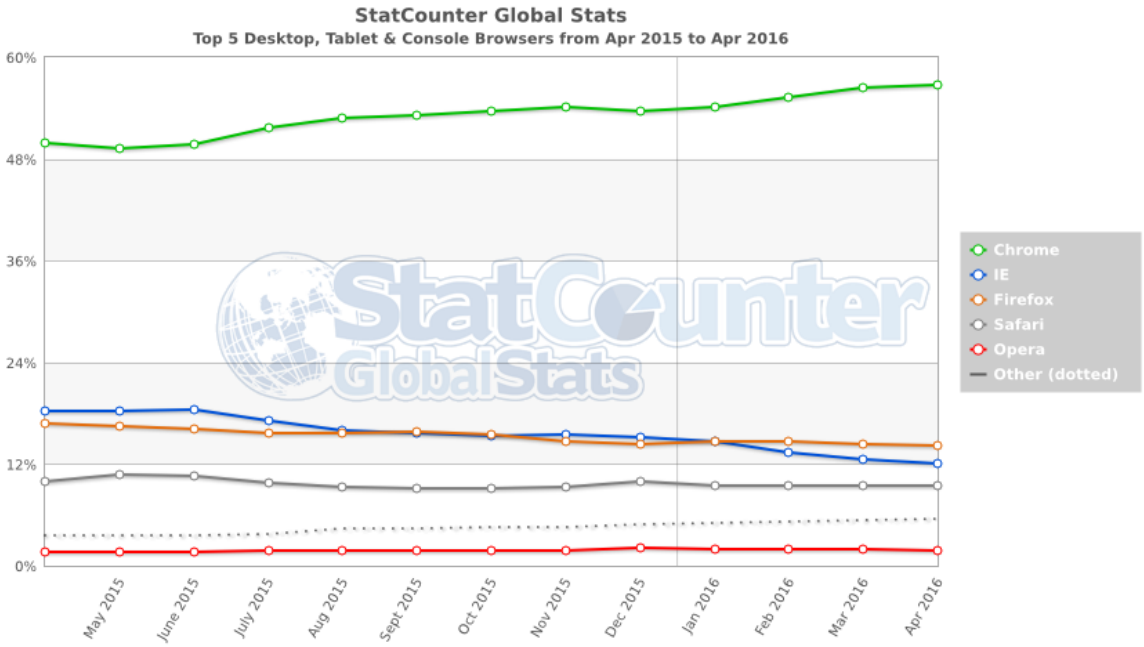
Server je počítač, který poskytuje nějaké služby stanicím připojeným k síti, ve které se server nachází. Může poskytovat služby jako je tiskový server, datový server, DNS server atd. Stanice, které chtějí těchto služeb využívat, musí být k serveru připojeny.

3.3.2 WWW

Světová síť, na které se nacházejí dokumenty (např. webové stránky). Základem WWW je tzv. hypertext, což je systém vzájemně propojených dokumentů, které jsou pomocí WWW přístupné. S pomocí browseru je možné si tyto dokumenty prohlížet, vyhledávat, sledovat videa atd.

3.3.3 Browser

Program, pomocí kterého lze prohlížet internetové stránky. Browser má základní funkci číst HTML, CSS, PHP dokumenty a další, které patří ke stránce, pomocí HTML zobrazit obsah stránek a pomocí CSS stránky nastylovat, takže ve výsledku uživateli zobrazí kompletní stránku. Browsery také dokážou např. za pomocí pluginu přehrát multimédia jako videa nebo muziku. Dokáže také využívat také využívat vybavení stanice jako je kamera nebo mikrofon a používat je k interakci se zobrazovanou stránkou. Mezi nejpoužívanější browsery se řadí Google Chrome, Mozilla Firefox, Opera, Internet Explorer a Safari.



Obr. 2. Statistika používání prohlížečů [10]

4 ZABEZPEČENÍ

Při vytváření webových stránek se musí brát ohled i na jejich zabezpečení. Špatně zabezpečený web může být lehce napaden útokem hackera. Při úspěšném útoku může útočník odcizit citlivé informace, nahrát na web škodlivý vir nebo i smazat celou databázi.

Při zabezpečení se musí kontrolovat vstupy od uživatele, ochrana session, ochrana proti SQL injection nebo XSS.

4.1 Kontrola vstupů

Mezi kontrolované vstupy patří např. formuláře, URL adresy požadavku, cookies. Všechny vstupy od uživatele na webové stránce by měly být před dalším zpracování kontrolovány, ať už z důvodu omylem vyplněného formuláře nebo adresy, nebo záměrného útoku na stránku.

Vstupy můžeme kontrolovat testováním, zda vstupní data obsahují pouze povolené znaky, nebo můžeme použít v jazyku PHP funkci *htmlentities();* na převod HTML značek na speciální znaky.

4.2 Ochrana session

Dalším bodem zabezpečení je ochrana session. Session jsou data, která se uchovávají na straně serveru, přístup k session lze pomocí *$_SESSION*. Session nejsou nijak zabezpečené a z toho důvodu se musí session hlídat. Hlavním cílem zabezpečení session je, aby útočník nezískal identifikátor session a nemohl tak ukrást identitu jiného uživatele.

Identifikátor session lze zjistit tak, že útočník vloží odkaz do např. do diskuze, uživatel si otevře odkaz, který útočník vložil do diskuze. Jakmile se otevře uživateli stránka, tak útočník může přes proměnou *$_SERVER['HTTP_REFERRER']* získat kompletní URL včetně identifikátoru session, pokud byl předáván pomocí metody GET.

Na ochranu session je důležité mít správně ošetřené vstupy (kontrolovat vstupy, které může útočník využít) a v případě odkazovaní na jiný web jej otevřít v novém okně prohlížeče.

4.3 SQL injection

SQL injection se značí technika pro napadení SQL databáze webové stránky. Útočník může napadnout stránku tak, že vloží přes neošetřený vstup (formulář nebo URL) vlastní kód, který pozmění SQL dotaz na napadené webové stránce. Útočník tak může pozměnit SQL

příkaz, pomocí kterého může získat data z databáze, přihlásit se do administrace webu nebo dokonce smazat celou databázi. [11]

Jako příklad můžeme využít databázi uživatelů, která se využívá pro přihlášení na webové stránky. Na webu mám formulář na zadání jména a hesla. Útočník zadá do vstupu pro zadání jména např. *admin or 1=1*, SQL příkaz máme *SELECT * FROM uzivatele WHERE jmeno = '$jmeno';* proměnná *$jmeno* obsahuje řetězec *„admin or 1=1"*, který bude bez ošetření vložen do SQL dotazu, tím vznikne dotaz *SELECT * FROM uzivatele WHERE jmeno = admin or 1=1*. Protože 1=1 vrací *true*, SQL příkaz bude úspěšně proveden. Tak se může útočník dostat na nezabezpečenou stránku. Podobným způsobem lze smazat i tabulky v databázi, příkladem může být zadáno do vstupu *admin; DROP TABLE uzivatele*, v případě, že útočník zná nebo uhodne název tabulky, může ji celou smazat. [12]

Pro ochranu před útokem pomocí SQL injection si lze vytvořit slovník znaků, které se zadají přes vstup, pomocí takového slovníku můžeme znaky převést na jiné nebo je i zcela smazat. Další způsobem kontroly vstupu může být využití *magic_quotes(),* který uzavírá string do „/" neboli escapování pomocí *addslashes()*. Před odeslání vstupních dat do SQL příkazu lze také escapovat znaky funkcí *mysql_real_escape_string()*. Příkladem použití je *$jmeno = mysql_real_escape_string($_POST['jmeno']);*.

4.4 Cross Site Scripting

Cross site Scripting neboli XSS je metoda pro napadení webových stránek s využitím bezpečnostních chyb ve zdrojovém kódu webové stránky. XSS útok využívá nezabezpečené vstupy na webové stránce k podstrčení vlastního obsahu, změny vzhledu stránky, nefunkčnosti webové stránky nebo i odcizení údajů. Nezabezpečeným vstupem můžeme brát data, která jsou získávaná přes formuláře nebo URL adresy. [13]

Útočník může pomoci XSS podstrčit na nezabezpečené webové stránky svůj kód, který ohrozí funkčnost webových stránek. Jako potenciální kód útoku může být využit javascript.

Příkladem útoku může být zadání do URL adresy, která má formát *http://www.web.cz/kategorie.php?id=5*, útočník může upravit URL tak, aby vložil svůj kód na webovou stránku. Parametr *id* budeme brát jako proměnou *$_GET*. Pokud útočník podstrčí za parametr *id*, například *<script>alert('XSS útok');</script>*. URL tak bude mít formát *http://www.web.cz/kategorie.php?id=<script>alert('XSS útok');</script>* a

javascriptový kód se úspěšně provede. Stejným způsobem lze vložit do formuláře na webu stejný kód a pokud není vstup kontrolován, provede se následující kód.

Obranou proti XSS útoku je vstupní data kontrolovat a podezřelé znaky filtrovat a převádět na HTML entity, příkladem může být záměna značky < za *&lt* nebo > za *&gt*. Převod na HTML entity lze za použití funkce *htmlspecialchars()*. Příkladem použití je *$hledej=htmlspecialchars($_POST['hledany_vyraz']);*.

II. PRAKTICKÁ ČÁST

5 INFORMAČNÍ SYSTÉM

Před samotnou tvorbou informačního systému jsem provedl průzkum open source redakčních systémů, které bych mohl použít. Na základě požadavků vedoucího bakalářské práce pana doc. Ing. Jířího Vojtěška, Ph.D., jsem dospěl k závěru, že bych si i za použití open source redakčního systému musel některé pluginy sám doprogramovat a problém by dělal rozdělení článku do více kategorií, rozhodl jsem se, že raději naprogramuji vlastní informační systém.

Vytvoření informačního systému je rozděleno do několika kategorií. Prvním a důležitým bodem pro tvorbu webových aplikací je vytvoření přehledného a jednoduchého vzhledu s dobrou strukturou, která udrží návštěvníka na naší webové stránce. Dalším bodem je kódování navrhnuté šablony. Třetím bodem je navrhnutí a vytvoření tabulek v databázi. Do posledního bodu bych zařadil vytváření webových scriptů pro práci s databází, zobrazování obsahu na webu a zabezpečení stránky.

Informační systém je nahrán na adrese http://www.studijni.fai.utb.cz.

5.1 Požadavky na informační systém

Hlavním požadavkem na informační systém je správa článků a kategorií. Mezi další požadavky lze zařadit správa uživatelů a správu souborů.

Jednotlivé články se budou ukládat pod jednotlivé kategorie, která se vybere z již vytvořených kategorií. Při vytváření článku by měla být možnost zadat, od kdy do kdy by měl být článek zobrazen nebo být zobrazen nastálo. V seznamu článků v administraci by mělo být zobrazené, v jaké je článek kategorii, zda je zveřejněn a tlačítka na smazání a editaci. Také aby bylo možné přesouvat články v jednotlivých kategorií. Článek by měl jít také přidat do více kategoriích.

Při vytváření kategorie, by mělo jít přidat kategorii již pod nějakou dříve vytvořenou kategorii. U kategorií je požadavkem hlavně editace, mazání a řazení kategorií.

Informační systém by měl obsahovat také správu uživatelů a správu souborů. U uživatelů je to pouze přidání, editace a mazání uživatele. U správce souborů to je vytváření a mazání složky, přidání a mazání souborů a také zobrazení souborů.

Na základě těchto požadavků, jsem se rozhodl, že pro mě bude lepší vytvořit vlastní systém než využít již nějaký stávající redakční systém, např. wordpress, ať už z hlediska psaní

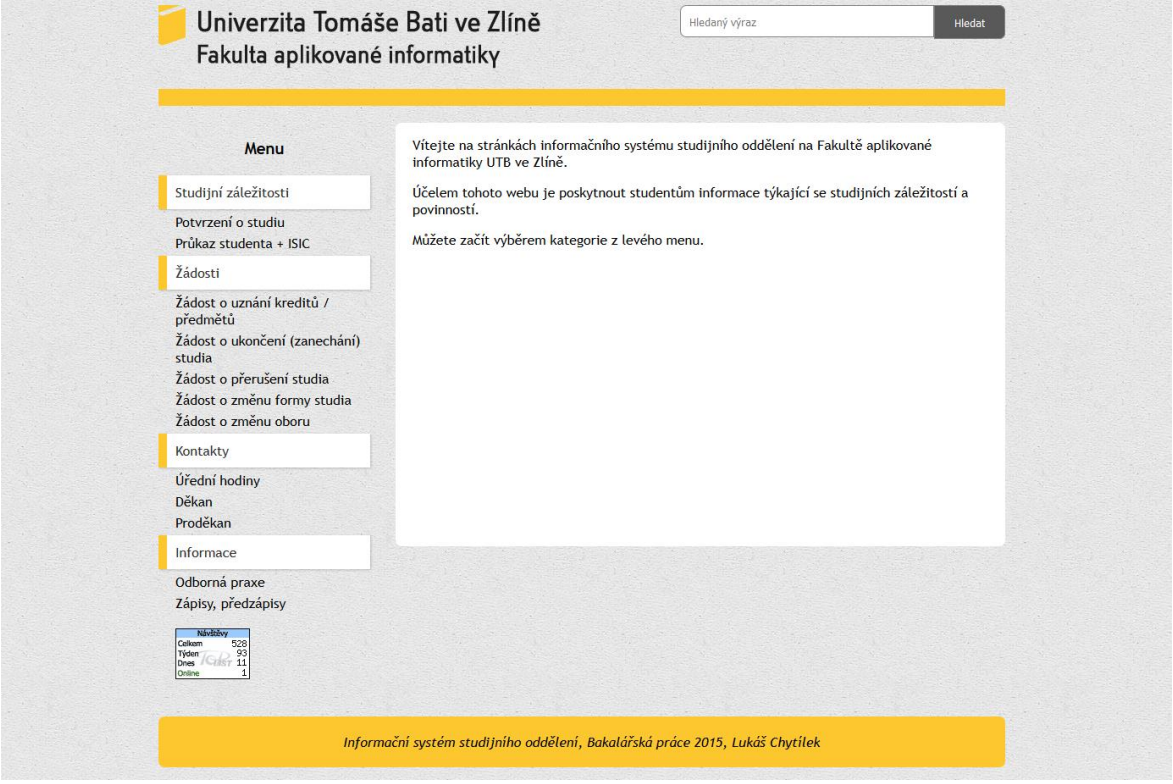
vlastní šablony a funkcí, či zkoumání, jak celý systém funguje, by mi zabrala práce stejný čas a navíc má dle mého názoru wordpress rozsáhlejší administraci, ve které by se bez manuálu mohl uživatel lehce překliknout nebo změnit, co nechce.

5.2 Vzhled informačního systému

Při vytváření vzhledu pro informační systém jsem se zaměřil hlavně na to, aby měl web stejné barevné kombinace a pozadí, jako mají webové stránky fakulty. Použil jsem design manual Univerzity Tomáše Bati.

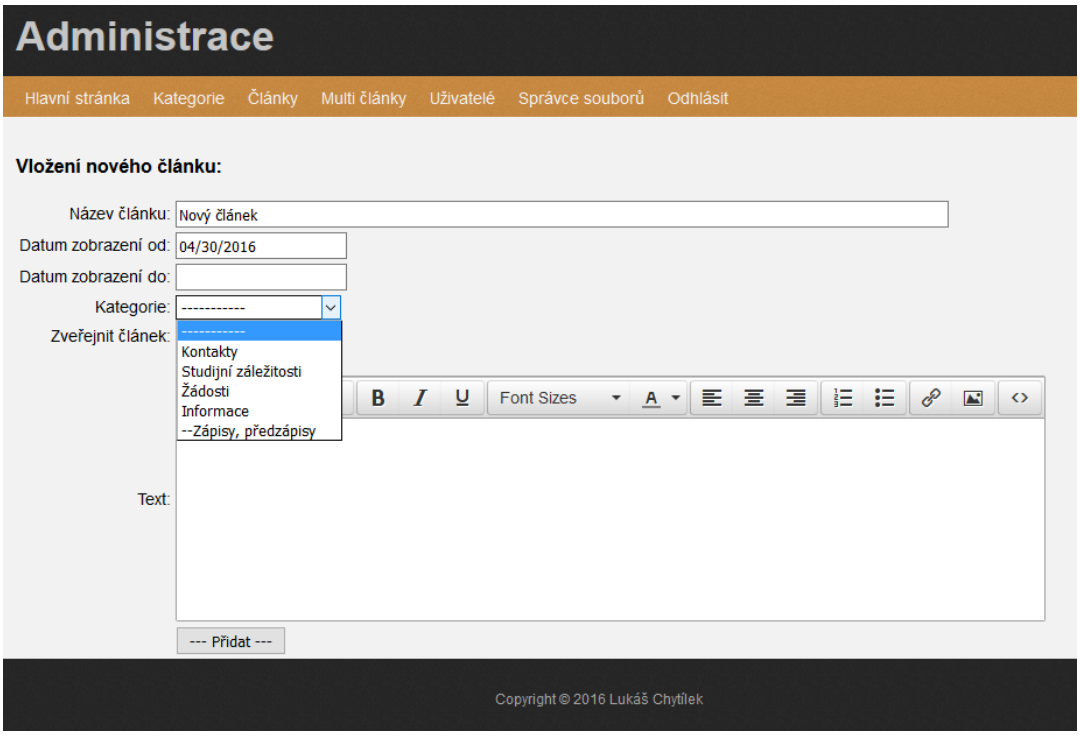
Obr. 3. Webové stránky fakulty

Při porovnání webové stránky fakulty a informačního systému lze na první pohled vidět několik rozdílů. Při vytváření informačního systému jsem již nepoužil horizontální menu a obrázek pod tímto menu. Rozhodl jsem se tak pro menu, které je zobrazené na levé straně informačního systému. Toto menu je strukturované do stromu pod sebe. V hlavičce informačního systému je logo fakulty a formulář pro vyhledávání.



Obr. 4. Vzhled informačního systému

Vzhled pro administrační část jsem také vytvořil pomocí HTML a CSS, skripty kombinují PHP a MySQL. Administrace má menu tvořeno horizontálně.



Obr. 5. Vzhled administrační stránky

5.3 Kódování šablony

Šablonu informačního systému jsem přepsal do jazyka HTML a CSS. HTML a CSS by měl být každý v samostatném souboru. Pomocí HTML jsem si rozvrhnul strukturu informačního systému a pomocí CSS jazyka jsem systém nastyloval do výsledné podoby.

HTML kód, viz Příloha PI, začíná tagem <*html*>, v něm je pak rozdělený na <*head*> a <*body*>. V části <head> se udává hlavička webové stránky, která se nezobrazuje. V hlavičce je nastavený <*title*> neboli nadpis webové stránky v prohlížeči. Hlavička také obsahuje nepovinné tagy, např. klíčové slova stránky, popis stránky, odkaz na CSS nebo javascriptový soubor. Mezi tagem <*body*> a <*/body*> se vytváří webová struktura. V tagu <*header*> je zobrazeno logo a formulář pro vyhledávání. Hlavní část informačního systému je obalen do bloku pomocí tagu <*div*> a v něm je pak rozdělená struktura na menu a pro zobrazení textu. V tagu <*footer*> je definovaná patička systému. Zdrojový kód není vypsaný kompletně.

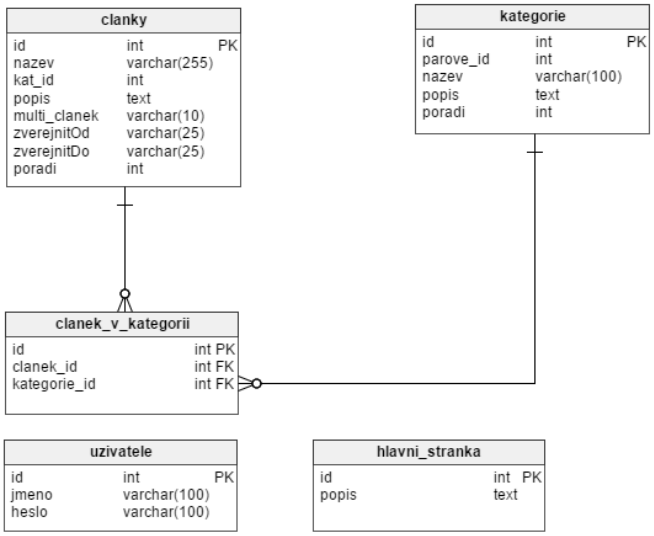
CSS se využívá pro stylování webové stránky, pro vysvětlení vkládám níže úryvek z kódu:

```
body{
  width: 100%;
  height: 100%;
  margin 0;
  padding 0;
  font-family: Trebuchet MS;
  background-image: url(../img/bg.png);
}
.main_site{
  width: 1000px;
  margin: 0 auto;
}
.banner{
  height: 20px;
  text-align: center;
  font-style: italic;
  background-color: #FDC82F;
  color: #FDC82F;
}
.main_content{
  min-height: 500px;
  overflow:auto;
}
.left_side{
  width: 25%;
  float: left;
  height: 100%;
  display: block;
}
```

Pro stylování webové stránky se musí v .css souboru napsat název tagu, který chceme stylovat a dát za něj složené závorky, do kterých píšeme kód. K různým tagům lze přiřadit i třídy nebo identifikátory, identifikátor se v CSS značí #nazev_indentifikátoru nebo třídu .nazev_třídy. V případě informačního systému jsem pro celou stránku nastavil jednotný font a obrázek pozadí. Stránka, na které se bude zobrazovat obsah jsem omezil šířku na 1000px a vycentroval do středu stránky. Třída *.banner* slouží k rozdělení hlavičky a zbytku stránky pomocí dlouhého boxu.

5.4 Tvorba databáze pro informační systém

Při vytváření databáze jsem musel brát v potaz požadavky na funkčnost informačního systému a zaměřit se na to, k čemu bude využíván.



Obr. 6. Model databáze

Prvním požadavkem bylo přihlášení do administrace pro správu obsahu. Aby bylo možné využívat přihlašování, vytvořil jsem tabulku *uzivatele*. Tabulka obsahuje sloupec *id*, *jmeno* a *heslo*. Heslo je zašifrováno pomocí SHA1.

Další tabulkou v databázi je tabulka *hlavni_stranka.* Tabulka je velmi jednoduchá, má pouze sloupec *id* a *popis*. Ukládá se zde obsah, který budu zobrazen na hlavní stránce informačního systému.

Pro členění do kategorií slouží tabulka *kategorie*. Tabulka se skládá ze sloupců *id, parove_id, nazev, popis a poradi*. Při vytváření kategorií pomocí administrace, lze k vytvářené kategorii přiřadit jinou kategorii, pod kterou pak bude zobrazena.

Dalším bodem je vytváření a správa článků na webu. K tomu je vytvořena tabulka *clanky*, kde každý článek mý svoji *id, nazev, kat_id, popis, multi_clanek, zverejnitOd, zverejnitDo a poradi.* Sloupec *multi_clanek* slouží k ukládání hodnot *true/false*. Sloupec *poradi* definuje řazení článků podle nastavení v administraci.

Zajímavým bodem v administraci je vytváření tzv. „multi článků“, které umožňují přidat jeden článek k více kategoriím i podkategoriím. Článek, který bude vytvořen jako „multi článek“, mu bude uložena do sloupce *multi_clanek* hodnota *true*. Zároveň se bude také ukládat do tabulky *clanek_v_kategorii* do každého řádku id článku a id kategorie, ve které se bude zobrazovat. Relace těchto databázi je 1:n (jeden článek k několika kategoriím).

5.5 Vytváření webových scriptů

Při vytváření informačního systému jsem si vytvořil vlastní scripty, abych si usnadnil práci při programování a zpřehlednil kód, protože kdybych vše vkládal např. do souboru **index.php**, měl by soubor velké množství řádků kódu a z mého pohledu by to nebylo moc ideální. Všechny funkce, které jsem si vytvořil, jsem uložil do souboru **functions.php**, třídy a funkce pro práci s databázi mám uložené ve složce includes.

5.5.1 PHP scripty pro práci s databází

Při tvorbě scriptů pro práci s SQL databází jsem využíval ORM model (Objektově relační mapování) a MySQL knihovnu PDO. Pro každou tabulku v databázi jsem vytvořil vlastní třídu. Každá třída obsahuje základní operace CRUD (vytvoření, čtení, úprava a mazání z tabulky v databázi).

Příkladem může být třída **KategorieClass.php**, u deklaraci proměnných je vytvořen objekt na připojení k databázi. Připojení k databázi probíhá v kontruktoru pomocí příkazu *$Conn = new PDO();* Ve třídě jsem vytvořil funkce *Load($intId), LoadByParoveId($intId), LoadAll(), LoadAllByParoveId(), Save()* a *Delete().* Funkce *Load($intId)* je naprogramovaná takto:

```
public static function Load($intId){
        $Contact = new Kategorie();
        $stmt = self::$Conn->prepare('SELECT * FROM kategorie WHERE id = :id');
        $stmt->setFetchMode(PDO::FETCH_CLASS, 'Kategorie');
        $stmt->execute(array('id'=>$intId));
        return $Contact = $stmt->fetch();
}
```

Na začátku funkce se vytvoří instance na třídu *Kategorie*, poté si připravíme dotaz na vyhledání kategorie podle jména. Příkazem *execute* se provede dotaz a vrátí výsledek hledání.

Pro výpis dat podle *id* se musí funkce zavolat *$load = Kategorie::Load(5);* Pokud kategorie pod *id* = 5 existuje, lze vypsat název pomocí *echo $load->Nazev;*

Funkce *LoadByParoveId()* je napsaná na stejném principu, pouze je změněná podmínka u SQL dotazu, kde je *id* vyměněné za *parove_id*. Funkce se volá v programu stejným způsobem jako *Load()*. Funkce *LoadAll* uloží všechny položky z databáze do pole. Pro zobrazení výsledků je nutné pak projít pole funkcí *foreach* pro vypsání hodnot. Nejdříve si nechám načíst do pole hodnoty *$poleDat = Kategorie::LoadAll();* poté pomocí funkce *foreach($poleDat as $data){ echo $data->Nazev;}. Tímto způsobem lze vypsat všechny kategorie*. Příklad funkce *LoadAll()*:

```
public static function LoadAll(){
        $Contact = new Kategorie();
        $arrObjects = array();
        $stmt = self::$Conn->query('SELECT * FROM kategorie');
        $stmt->setFetchMode(PDO::FETCH_CLASS, 'Kategorie');
        while($Contact = $stmt->fetch()){
                array_push($arrObjects, $Contact);
        }
        return $arrObjects;
}
```

Funkce *LoadAllByParoveId()* načte také všechny záznamy do pole ale pouze v případě, že je splněna podmínka.

Funkce *Save()* slouží pro vytváření a úpravu kategorie. Na začátku funkce je testováno, pokud již s takovým *id* nějaká kategorie již neexistuje. Pokud tomu tak není a *id* žádná kategorie nevlastní, je vytvořená nová kategorie. V opačném případě se data jen upraví na řádku s daným *id*. Před voláním funkce *Save() je potřeba vytvořit instanci třídy, např. $instance = new Kategorie();* poté lze jednoduchým způsobem pomocí setterů nastavit proměnné, které se mají uložit, *$instance->Nazev = 'Nova kategorie';* novou kategorii lze nakonec uložit vložením funkce save *$instance->Save();*.

Příklad funkce *Save()*:

```
public function Save(){
try {
        if($this->id){
        $stmt = self::$Conn->prepare('UPDATE kategorie SET parove_id=:paroveId na-
        zev=:nazev, popis=:popis WHERE id=:id');
        $stmt->execute(array(
                ':id' => $this->id,
                ':paroveId' => $this->parove_id,
                ':nazev' => $this->nazev,
                ':popis' => $this->popis
         ));
        echo 'UPDATED!!!';
        } else{
```

```
            $stmt = self::$Conn->prepare('INSERT INTO clanky (parove_id, nazev, popis) VALUES
            (:paroveId,:nazev,:popis)');
            $stmt->execute(array(
                    ':paroveId' => $this->parove_id,
                    ':nazev' => $this->nazev,
                    ':popis' => $this->popis
                    ));
            return self::$Conn->lastInsertId();
            }
    } catch (PDOException $e){
            echo 'Error: '.$e->getMessage();
    }
    }
```

Pro každou tabulku jsem vytvořil vlastní třídu, která se velmi podobá třídě **KategorieClass.php.**

5.5.2 PHP scripty na webové stránce

Pro informační systém jsem vytvořil několik funkcí, pro usnadnění práci se systémem. Scripty ukládám do samostatného souboru s názvem **functions.php**.

Scripty obsahují výpis kategorií do stromu, výpis článků, hlavní stránky, vyhledávání na stránce a přihlášení do administrace.

Funkce pro výpis hlavní stránky:

```
function Main()
 {
    $hlavni = HlavniStranka::Load(0);
    echo $hlavni->Popis;
 }
```

Funkce není nijak složitá, pouze načte dat z databáze.

Pro výpis článků na webu jsem vytvořil funkci, která využívá třídu **ClankyClass.php** pro načtení dat z databáze. Výpis zdrojového kódu:

```
function ArticleList()
 {
   $id = $_GET['id'];
   $id=htmlspecialchars($id);
   $id=htmlentities($id, ENT_QUOTES);
   $clanek = Clanky::Load($id);
   if($clanek){
     echo "<h2>".$clanek->Nazev."</h2>";
     echo $clanek->Popis;
   }
 }
```

Kategorie má v databázi uložený i popis, který je třeba zobrazit na stránce. Funkce nejdříve načte data z databáze pomocí funkce *Load()*, jakmile se vypíše popis kategorie, pokračuje script dál procházet kód, který dále vypisuje všechny články a kategorie pod danou kategorií. Zdrojový kód:

```
function CategoryNameList()
```

```
{
  $id = $_GET['id'];
  $id=htmlspecialchars($id);
  $id=htmlentities($id, ENT_QUOTES);
  $kategorie = Kategorie::Load($id);
  if($kategorie){
   echo "<h2>".$kategorie->Nazev."</h2>";
   echo $kategorie->Popis;
   echo "<h4>Odkazy:</h4>";
   $clanek_list = ClanekVKategorii::LoadAllByKatId($id);
   if($clanek_list){
    foreach($clanek_list as $mlt_clanek){
     $clanek = Clanky::Load($mlt_clanek->Clanek_id);
     if($clanek){
      echo "<a href='clanek.php?id=".$clanek->Id."'>".$clanek->Nazev."</a><br>";
     }
    }
   }
   $kategorie_list = Kategorie::LoadAllByParoveId($id);
   if($kategorie_list){
    foreach($kategorie_list as $kat){
     echo "<a href='kat.php?id=".$kat->Id."'>".$kat->Nazev."</a><br>";
    }
   }
   $clanky = Clanky::LoadAllByKatId($id);
   if($clanky){
    foreach($clanky as $clanek){
     echo "<a href='clanek.php?id=".$clanek->Id."'>".$clanek->Nazev."</a><br>";
    }
   }
  }
}
```

Protože je v systému i možnost vyhledávání, musí být data zpracovány. K tomu slouží funkce *Search*, která si převezme parametry ze zadaného vstupu. Pokud jsou zadány maximálně tři písmena, vyhledávací script hledá pomocí regulárních výrazů. V případě, že je vetší než tři písmena, hledá se pomocí fulltextu. Pokud script úspěšně najde v databázi nějaké data, vypíšou se odkazy pod sebe. Zdrojový kód:

```
function Search(){
  $hledat=$_POST['search'];
  if($hledat){
   $hledat=htmlspecialchars($hledat);
   $hledat=htmlentities($hledat, ENT_QUOTES);
   if(strlen($hledat) < 4){
    $hledatsql="SELECT * FROM clanky WHERE popis REGEXP '$hledat'";
   }
   else if(strlen($hledat) >= 4){
    $hledatsql="SELECT * FROM clanky WHERE MATCH(popis) AGAINST('$hledat') ";
   }
   $vypishledat=mysql_query($hledatsql);
   $hledatcisla=mysql_num_rows($vypishledat);
   $hledat=html_entity_decode($hledat);
   if ($hledatcisla) {
    echo "<h3>Zadaný výraz: ".$hledat."</h3>";
    echo "<strong>Nalezeno ".$hledatcisla." výsledků</strong> <br />";
    echo "<br />";
```

```
    while ($vypsathled=mysql_fetch_array($vypishledat)) {
     echo '<a href="clanek.php?id='.$vypsathled['id'].'">'.$vypsathled['nazev'].'</a>';
     echo "<br />";
    }
   }
   if ($hledatcisla == 0) {
    echo "<h3>Zadaný výraz: ".$hledat."</h3>";
    echo "<strong>Nalezeno 0 výsledků</strong> <br />";
    echo "<br />";
   }
  } else{
   echo "<h3>Zadaný výraz: ".$hledat."</h3>";
   echo "<strong>Nalezeno 0 výsledků</strong> <br />";
   echo "<br />";
  }
 }
```

Protože je v informačním systému i administrace, vytvořil jsem script na kontrolu přihlašování do systému. Script kontroluje vstupní data z přihlašovacího formuláře a pracuje s nimi. Nejdříve se pokusí získat data z databáze podle jména, pokud najde jména v databázi, porovnají zadané heslo a heslo uložené v databázi. Jestliže vše budou souhlasit, podmínka vrátí *true* a přihlášení proběhlo úspěšně. Zdrojový kód:

```
$jmeno=htmlspecialchars($_POST['jmeno']);
$jmeno=htmlentities($jmeno, ENT_QUOTES);
$heslo=htmlspecialchars($_POST['heslo']);
    $heslo=htmlentities($heslo, ENT_QUOTES);
    $heslo_md5=md5($heslo);
    $uzivatel = Uzivatele::LoadByJmeno($jmeno);
    if($uzivatel){
     if($uzivatel->Heslo != $heslo_md5){
      echo "<span class='login_kontrola'>Špatné jméno nebo heslo!</span>";
     } else {
      echo "<span class='login_kontrola'>Úspěšně přihlášen.</span>";
      $_SESSION['jmeno']=$jmeno;
      $_SESSION['heslo']=$heslo;
     }
    } else {
     echo "<span class='login_kontrola'>Špatné jméno nebo heslo!</span>";
    }
```

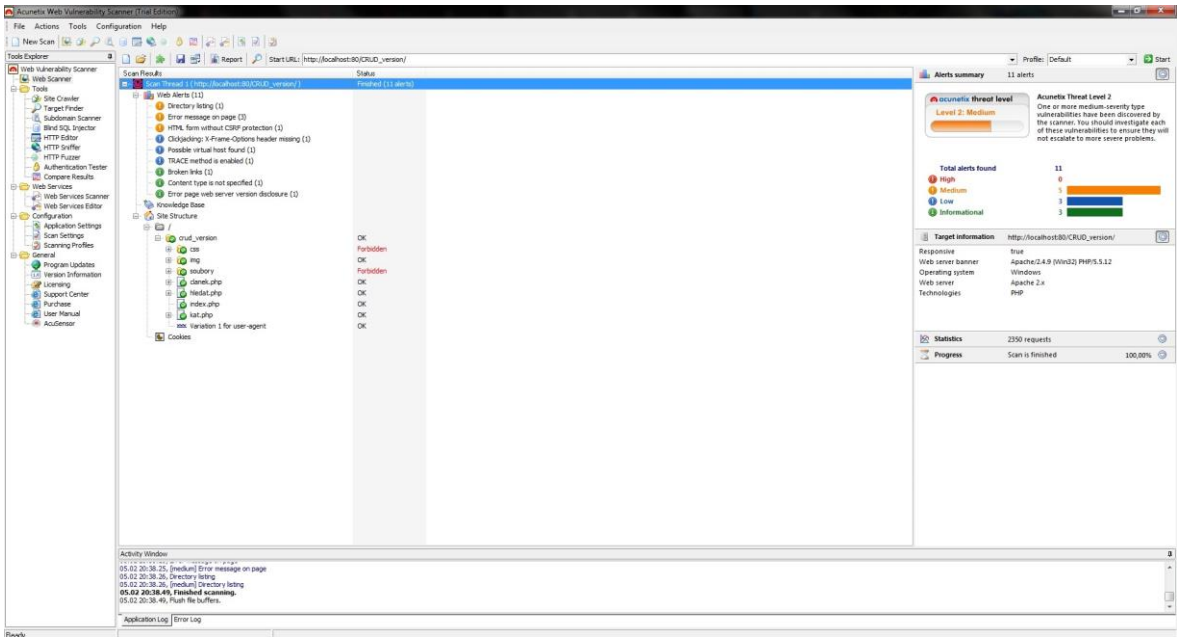
5.6 Zabezpečení informačního systému

Při vytváření informačního systému jsem musel myslet i na zabezpečení systému, protože nikdy není možné říci, čeho je uživatel schopen.

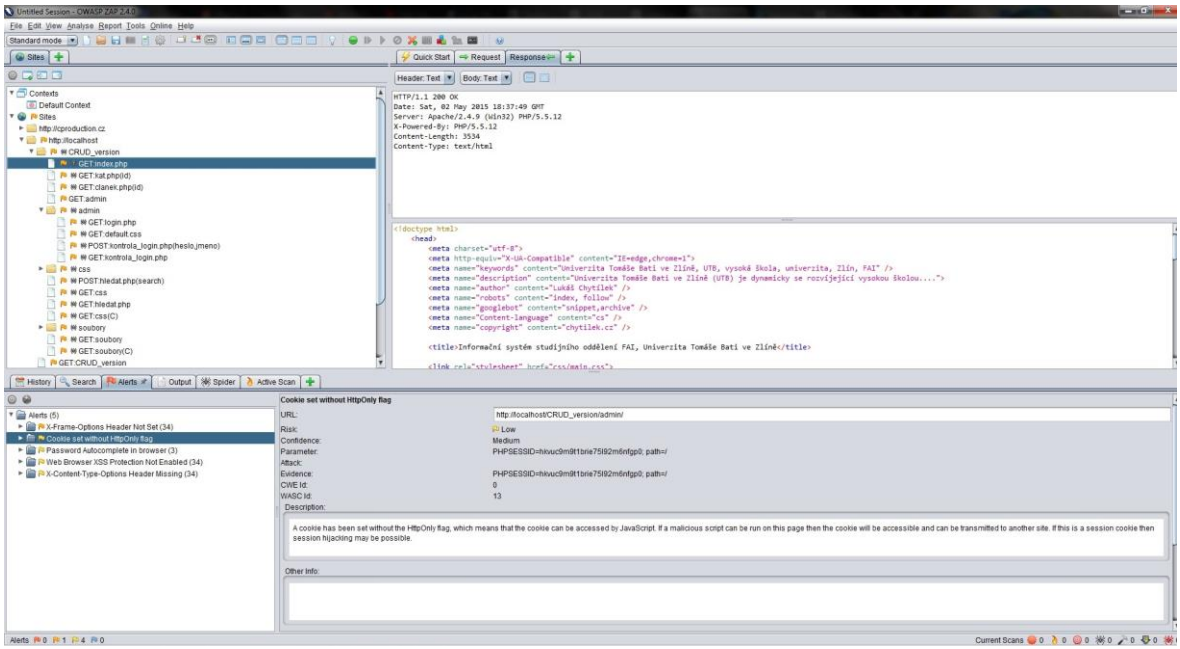
Jako první krok jsem si ošetřil všechny možné vstupy pomocí funkcí *htmlspecialchars()* a *html_entities()*. Dalším krokem jsem nastavil šifrování hesla u přístupu do administrace metodou *SHA1*.

Pro analýzu bezpečnosti informačního systému jsem využil 14-ti denní Trial verzi programu Acunetix Web Vulnerability Scanner a OWASP ZAP 2.4.0.

Když jsem spustil obě analýzy bez ošetření vstupních dat, oba ve zkratce popsali, že informační systém je velmi zranitelný, hlavně proti SQL injection a XSS útoku. Další testování jsem provedl již se zabezpečenými vstupy. Po provedení testu nebylo na již nikde psané, že web je náchylný k SQL injection a XSS útoky.



Obr. 7. Výsledek testu programu Acunetix Web Vulnerability Scanner



Obr. 8. Výsledek testu programu OWASP ZAP 2.4.0.

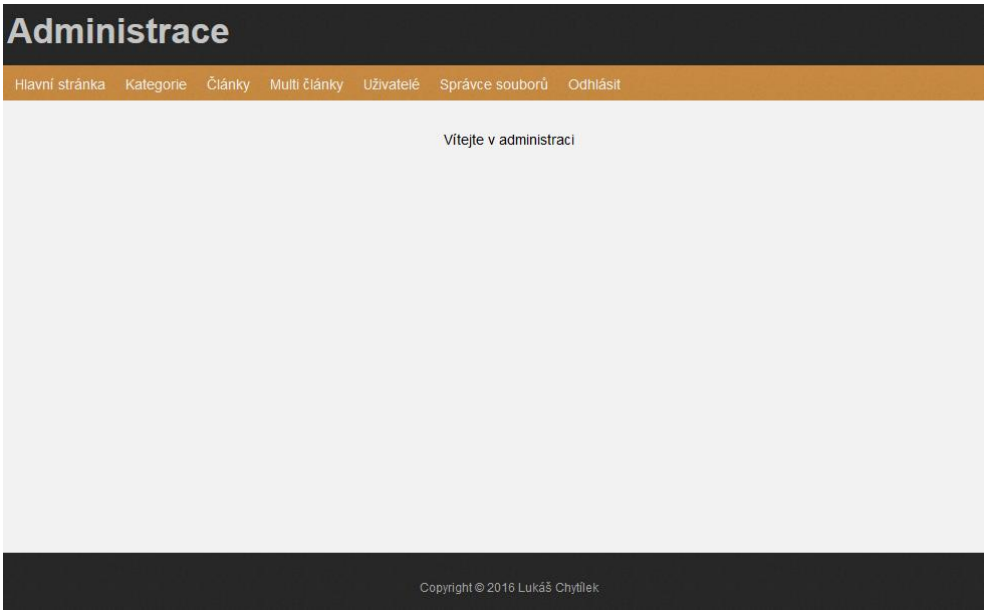
Podle výsledků u jednotlivých programů jsem zjistil, že informační systém je chráněn proti SQL injection a XSS útokům. Samozřejmě je potřeba brát takové výsledky s rezervou.

6 VKLÁDÁNÍ OBSAHU DO INFORMAČNÍHO SYSTÉMU

Hlavní předností informačního systému je skrze administraci vkládat obsah na své webové stránky. Jelikož naprogramovaný informační systém obsahuje i administraci pro správu obsahu, tak není vůbec težké takový článek vložit na webové stránky.

6.1 Postup pro vložení obsahu do IS

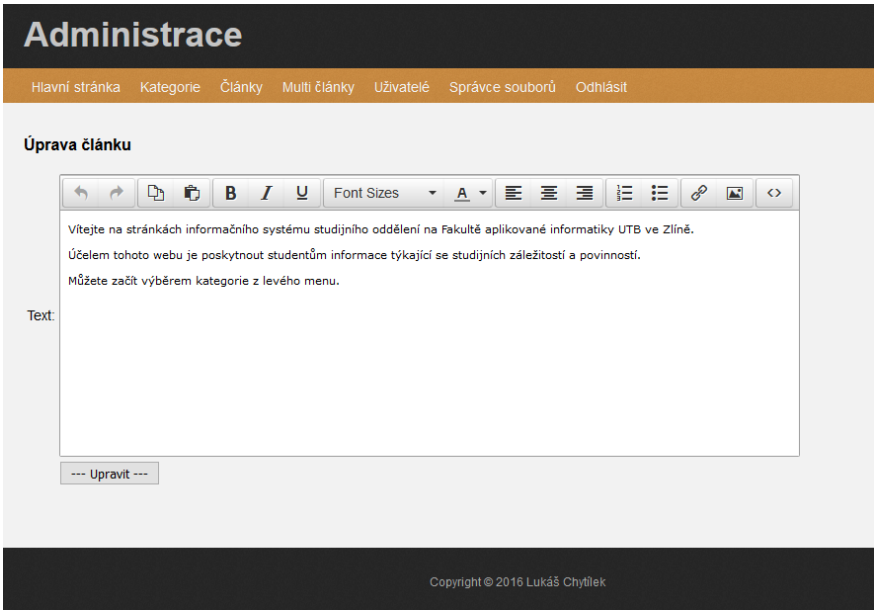
Abychom mohli vložit do našeho IS nějaký obsah, musíme se přihlásit do administrace. Po správném přihlášení se nám zobrazí hlavní stránka s menu.



Obr. 9. Hlavní stránka administrace

6.1.1 Úprava textu hlavní stránky

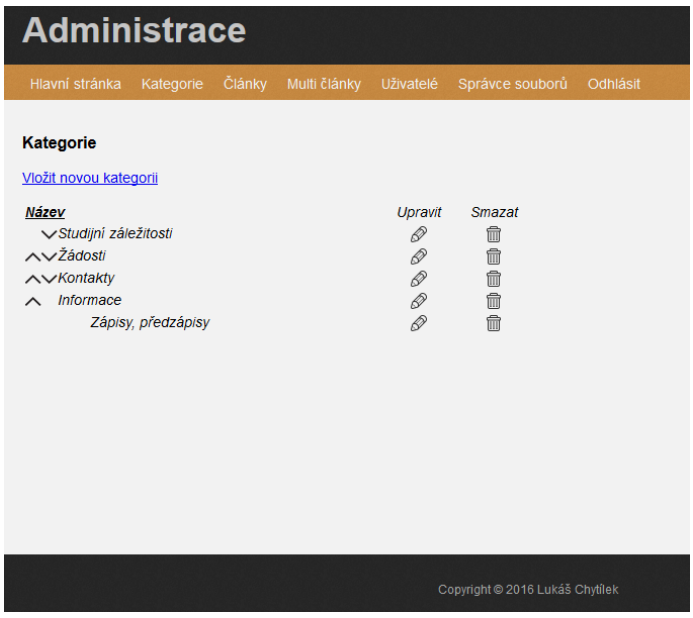
Pro úpravu textu na hlavní stránce si otevřeme v menu položku „hlavní stránka“, kde se zobrazí pouze textové pole s WYSIWYG editorem TinyMCE [18]. Po úpravě textu uložíme kliknutím na tlačítko „upravit“. Po kliknutí na tlačítko se stránka aktualizuje.



Obr. 10. Úprava hlavní stránky

6.1.2 Vložení a úprava kategorie

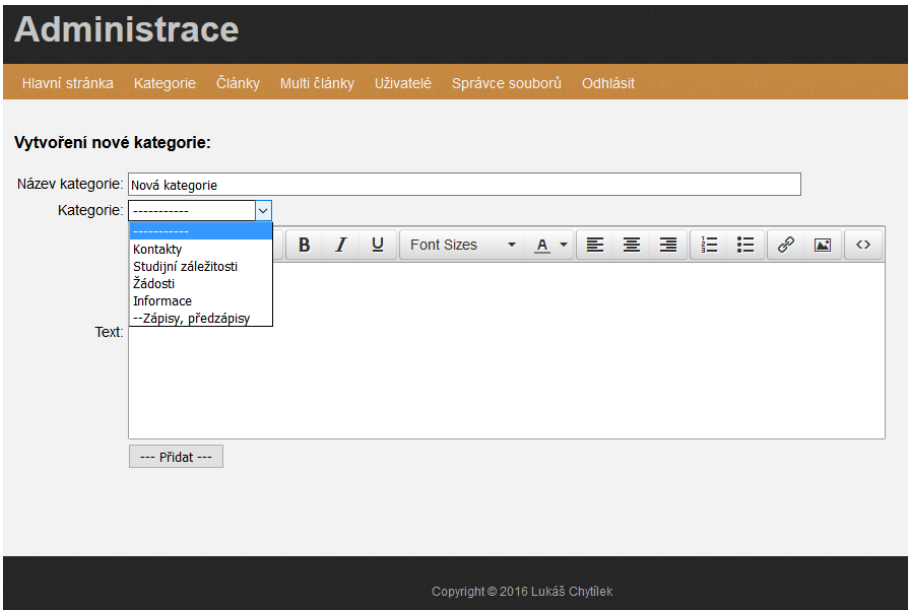
Dalším možností administrace je vložení a úprava kategorií. Po kliknutí v menu na kategorie se zobrazí stránka s odkazem na vložení nové kategorie a pod tím je seznam již vytvořených kategorií zobrazených ve stromové struktuře. Dá se také měnit řazení jednotlivých kategorií.



Obr. 11. Seznam kategorií

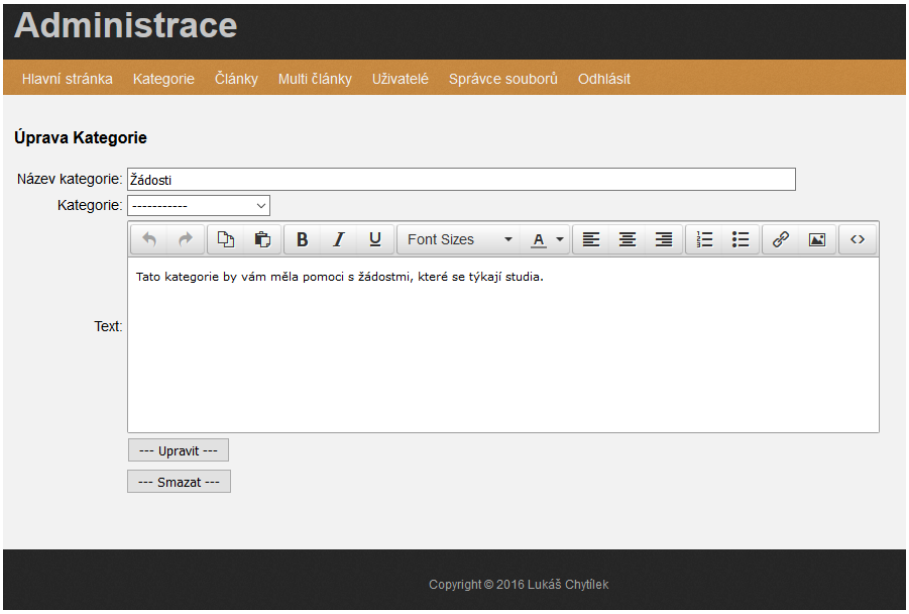
Kliknutím na odkaz „vložit novou kategorii“ se otevře nová stránka, která bude vyžadovat vyplnění následujících údajů: název kategorie, kategorie a text. Položka kategorie, která se

vyplňuje pod názvem kategorie, slouží k tomu, abychom mohli novou kategorie vložit již pod vytvořenou kategorii.



Obr. 12. Vložení nové kategorie

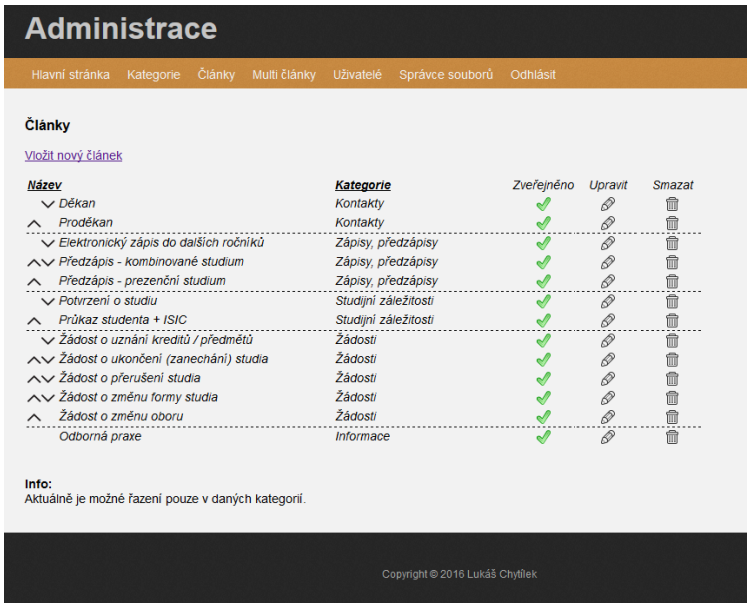
Abychom mohli kategorii upravit, v seznamu kategorií stačí na danou kategorii kliknout a otevře se nové okno, kde jsou pole již vyplněné a my je můžeme upravit. Pokud chceme kategorii smazat, slouží k tomu tlačítko „smazat“.



Obr. 13. Úprava kategorie

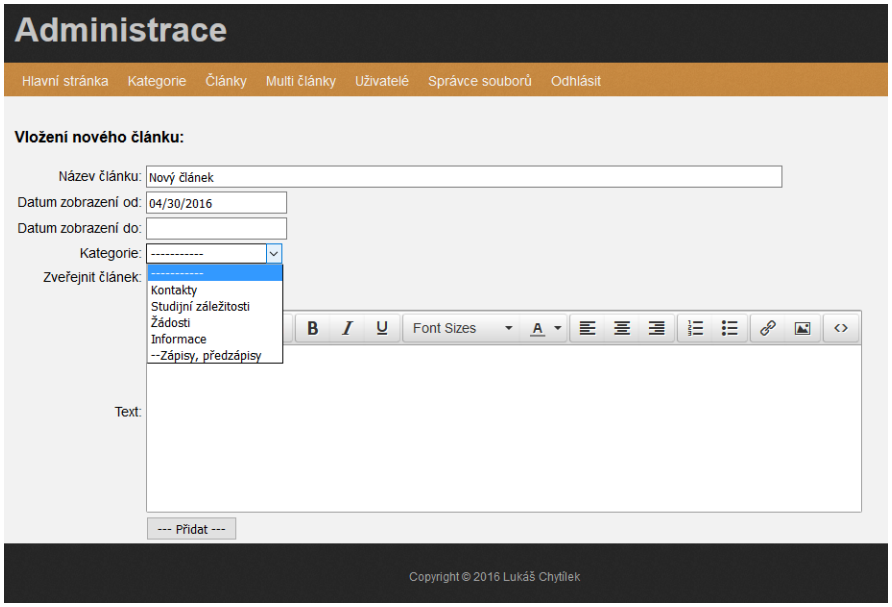
6.1.3 Vložení a úprava článku

Chceme-li si vložit nebo upravit nějaký článek, klikneme v menu na „články“. Po kliknutí se zobrazí odkaz „vložit nový článek“ a pod ním seznam již vložených článků. V seznamu článků lze řadit články v dané kategorii a je zde i zobrazeno, zda je článek zveřejněn. Články se dají posouvat pouze v každé kategorii zvlášť a posunutí se provede klikem na šipku dolů nebo nahoru. Posouvání článků se projevuje na stránce v menu, kde jsou články umístěny v menu tak, jak se nastaví v administraci. Příkladem může být seznam článků níže na Obr. 14, kde lze vidět jednotlivě seřazené články podle kategorií od sebe oddělené, kde jde vidět, že lze posouvat články pouze vrámci kategorie, ve které se nachází. Např. chceme-li, aby se článek *Proděkan* zobrazil na stránce, pod kategorií *Kontakty*, první, kliknutím na šipku nahoru jej posuneme na první pozici a na stránce se nám poté zobrazí jako první v kategorie *Kontakty*.



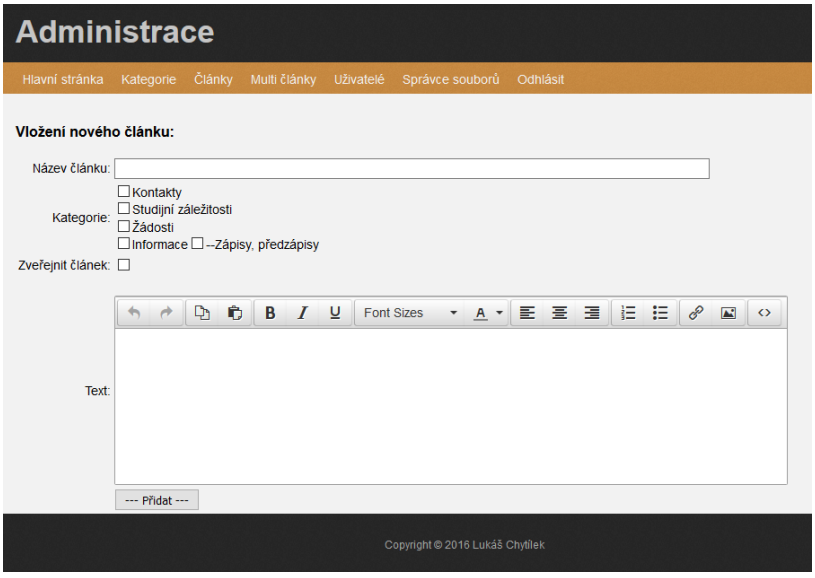
Obr. 14. Seznam článků

Vkládání a úprava článků je velmi podobná jako vložení a úprava kategorie. Slouží k tomu pole pro vložení názvu článku, výběr kategorie, pod kterou bude článek zobrazen a text.



Obr. 15. Vložení nového článku

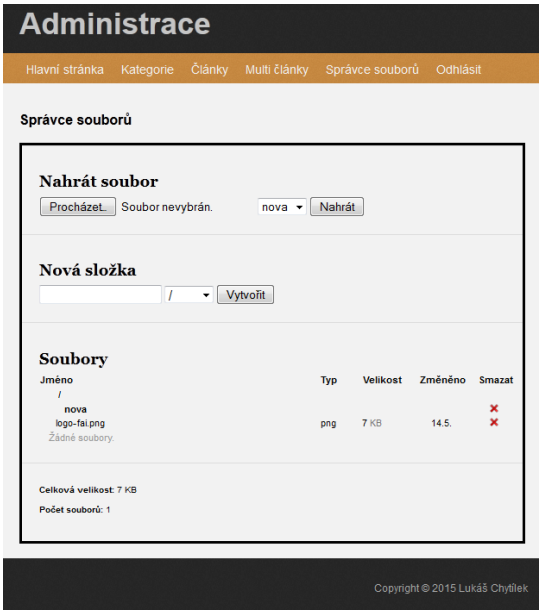
Jestliže chceme vložit jeden článek pod více kategorií, otevřeme v menu položku „multi články“. To funguje na stejný způsob jako přidávání a úprava článku s tím rozdílem, že při vytváření článku se již nevybírá jedna kategorie, ale může se jich pomocí check boxu vybrat více. Každý řádek zobrazuje hlavní kategorii a jeho podkategorie.



Obr. 16. Vložení nového multi článku

6.1.4 Vložení obrázků a souborů

Abychom mohli na web vložit obrázek nebo soubor, je potřeba ho nahrát přes administraci na FTP. To se udělá tak, že v menu otevřeme položku „Správce souborů“.



Obr. 17. Správce souborů

Ve správci souborů můžeme vytvářet nové složky a nahrávat soubory. Pro nahrátí nového souboru/obrázku klikneme na „Procházet“, vybereme v počítači soubor, který chceme nahrát, poté vybereme, do jaké složky chceme soubor nahrát a kliknem na „Nahrát“. Takhle se nahraje na FTP jakýkoliv soubor.

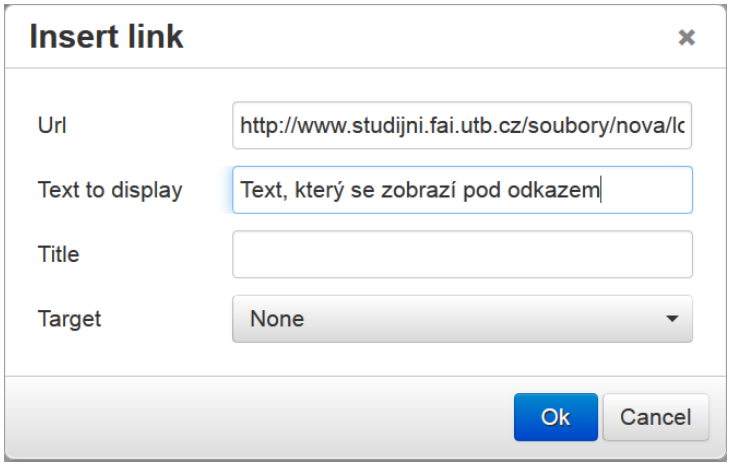
Soubory se dají ve správci souborů otevírat i mazat. Soubory se na disk ukládají do adresáře „soubory“.

6.1.1 Zobrazení souborů a obrázků

Soubory, které byly úspěšně nahrány na FTP, nyní můžeme zobrazit na webu. Příklad si můžeme ukázat na hlavní stránce, otevřeme v menu „Hlavní stránka“. V panelu textového editoru je na konci ikona odkazu a obrázku.

Obr. 18. Panel textového editoru

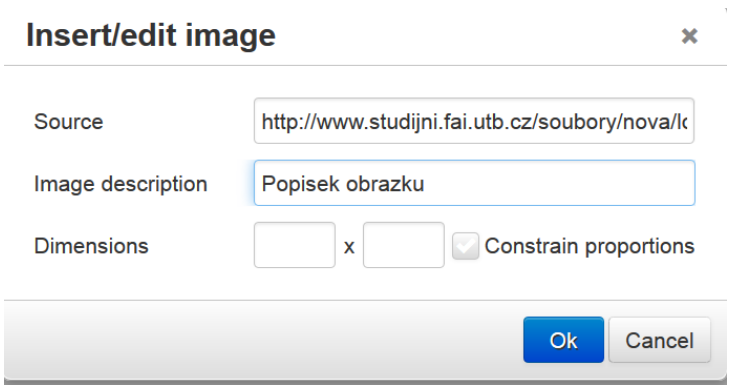
Chceme-li soubor zobrazit jako odkaz, klikneme na ikonu odkazu. Do Url napíšeme cestu k souboru. Na Obr. 17. je vidět, že máme uložený soubor, který je ve složce „nova“, do Url musíme napsat **http://www.studijni.fai.utb.cz/soubory/nova/logo-fai.png**, vyplníme text, který se zobrazí pod odkazem a dáme „Ok“.



Obr. 19. Vložení odkazu

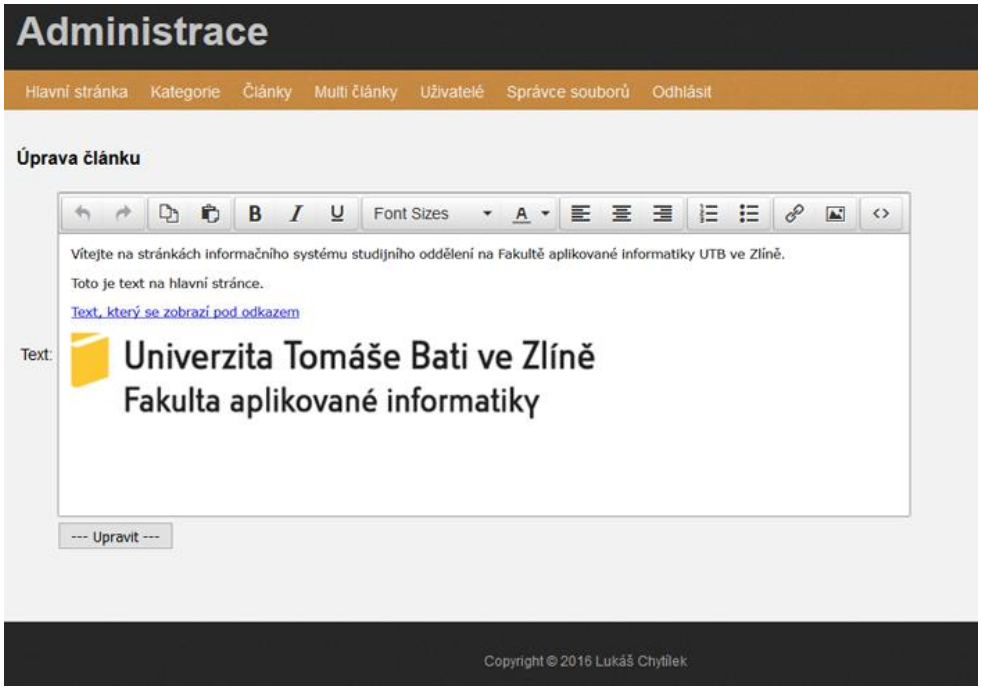
Pokud ovšem chceme vložit klasický odkaz na jinou stránku, zadáme do Url celou adresu, např. **http://google.com**.

Pro vložení obrázku na web vybereme v panelu textového editoru ikonu obrázku, která je jako poslední. Do Source napíšeme cestu k souboru, cesta bude stejná, jako byla u Url, tedy absolutní cesta bude **http://www.studijni.fai.utb.cz/soubory/nova/logo-fai.png**, může se použít také relativní cesta **soubory/nova/logo-fai.png**, ale obrázek již bude zobrazen bez náhledu v editoru (lze vidět jen orámování). Do „Image descrtiption“ můžeme napsat popisek obrázku a „Dimensions“ velikost obrázku, pokud jej chceme změnit.



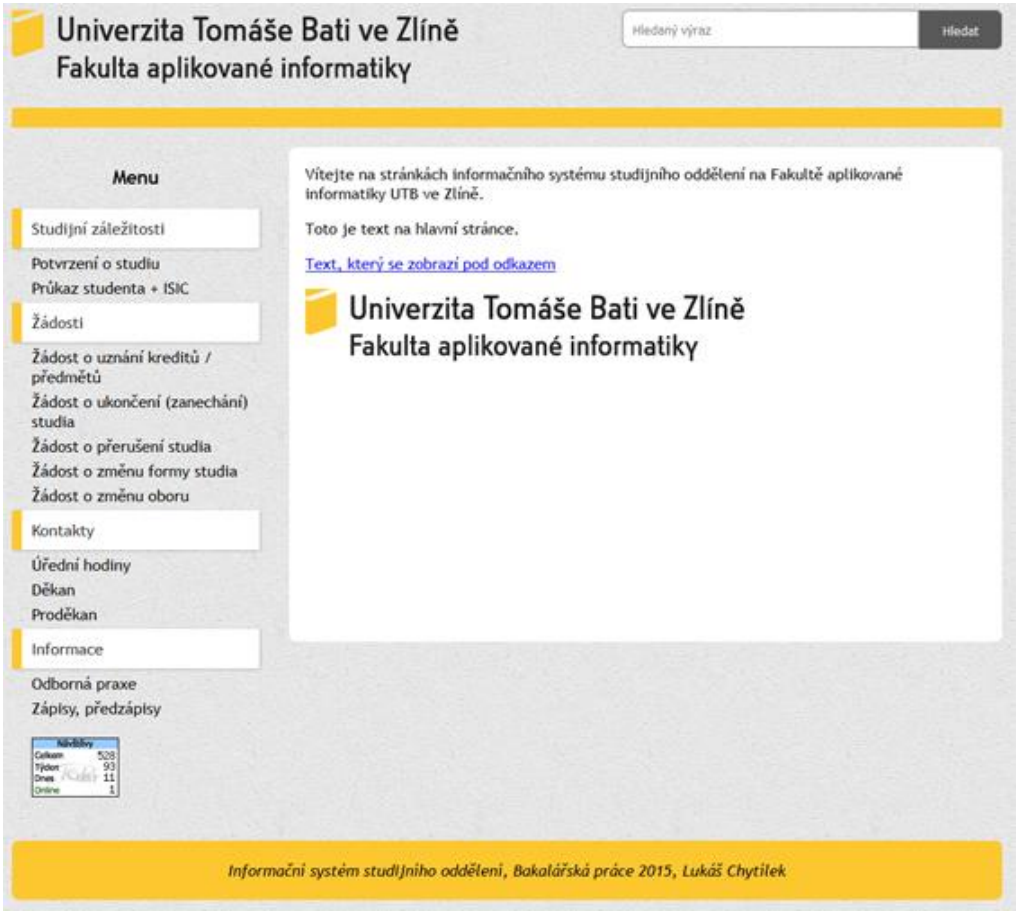
Obr. 20. Vložení obrázku

V editoru se pak zobrazí jak odkaz, tak i obrázek, viz Obr. 21.



Obr. 21. Výsledek vložení odkazu se souborem a obrázku

Na hlavní stránce se nyní zobrazí odkaz na soubor i náš vložený obrázek.



Obr. 22. Hlavní stránka s odkazem a obrázkem

ZÁVĚR

Cílem této bakalářské práce bylo vytvořit informační systém pro studijní oddělení, který bude sloužit pro zobrazení důležitých předpisů a informací týkajicích se studijních záležitostí. Pro informační systém byla vytvořena i administrační část, která slouží k úpravě a přidávání obsahu pomocí databáze.

V teoretické části jsem popsal, čím se zabývají předpisy a normy univerzity a fakulty. Je zde popsáno také základní informace v oblasti tvorby webových stránek a co vše je potřeba k tomu znát. V poslední části teorie je popsáno, jak se ochránit přes napadení webové stránky.

V praktické části jsou popsány způsoby, jakými jsem přistupoval ke tvorbě informačního systému pro studijní oddělení. Při vytváření vzhledu pro informační systém jsem využil design manul univerzity, kde je popsáno z jakých barev, fontů, atd. je vytvořen web fakulty. Informační systém jsem vytvořil pomocí jazyků HTML, CSS, PHP a MySQL.

Při práci jsem si musel vytvořit několik pomocných scriptů hlavně pro práci s databázemi. Pro každou tabulku v databázi jsem naprogramoval vlastní třídy, které slouží pro výpis, vkládání, úpravu a mazání dat z tabulky. K tomu jsem využil metody ORM, kterou jsem využil k základním operacím CRUD.

Součástí práce jsem se také musel zaměřit na bezpečnost informačního systému, aby neznámý návštěvník nemohl využít chyb webových stránek. Všechny vstupní data, která by mohl uživatel vyplnit, jsem ošetřil takovým způsobem, aby v případě zadání nějakého scriptu vyhlásila stránka chybu.

SEZNAM POUŽITÉ LITERATURY

[1] Studijní a zkušební řád. [online]. [cit. 2015-05-01]. Dostupné z: http://www.utb.cz/fai/o-fakulte-fai/predpisy-a-norny-fai

[2] Statistika používání CMS. [online]. [cit. 2015-05-01]. Dostupné z: http://gs.statcounter.com/

[3] vBulletin. [online]. [cit. 2015-05-01]. Dostupné z: http://www.vbulletin.com/en/about-us/

[4] Wordpress. [online]. [cit. 2015-05-01]. Dostupné z: http://www.cwordpress.cz/

[5] Joomla!. [online]. [cit. 2015-05-01]. Dostupné z: http://www.joomla.org/about-joomla.html

[6] Drupal. [online]. [cit. 2015-05-01]. Dostupné z: https://www.drupal.cz/o-systemu-drupal

[7] CSS. [online]. [cit. 2015-05-01]. Dostupné z: http://www.w3schools.com/css/css_intro.asp

[8] HTML. [online]. [cit. 2015-05-01]. Dostupné z: http://www.w3schools.com/html/html_intro.asp

[9] SQL. [online]. [cit. 2015-05-01]. Dostupné z: http://www.w3schools.com/sql/sql_intro.asp

[10] Statistika používání webových prohlížečů. [online]. [cit. 2015-05-01.]. Dostupné z: http://gs.statcounter.com/

[11] XSS. [online]. [cit. 2015-05-01]. Dostupné z: https://www.owasp.org/index.php/SQL_Injection

[12] SQL injection. [online]. [cit. 2015-05-01]. Dostupné z: http://www.w3schools.com/sql/sql_injection.asp

[13] SQL injection. [online]. [cit. 2015-05-01]. Dostupné z: https://www.owasp.org/index.php/XSS

[14] HOGAN, Brian P. *HTML5 a CSS3: výukový kurz webového vývojáře*. Vyd. 1. Brno: Computer Press, 2011, 272 s. ISBN 9788025135761.

[15] LUBBERS, Peter, Brian ALBERS a Frank SALIM. *HTML5: programujeme moderní webové aplikace*. Vyd. 1. Brno: Computer Press, 2011, 304 s. ISBN 978-80-251-3539-6.

[16] KOSEK, Jiří. *PHP: tvorba interaktivních internetových aplikací : podrobný průvodce*. Vyd. 1. Praha: Grada, 1999, 490 s. ISBN 8071693731.

[17] KOFLER, Michael a Bernd ÖGGL. *PHP 5 a MySQL 5: průvodce webového programátora*. Vyd. 1. Překlad David Čepička. Brno: Computer Press, c2007, 607 s. ISBN 9788025118139.

[18] WYSIWYG editor tinymce. [online]. [cit. 2015-05-015]. Dostupné z: http://www.tinymce.com/

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

CMS	Redakční systém
IS	Informační systém
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
PHP	Hypertext Preprocessor
MySQL	Databázový systém
ORM	Objektově relační mapování
CRUD	Create Read Update Delete
XSS	Cross-site scripting
FTP	File Transfer Protocol

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

Nenalezena položka seznamu obrázků.

SEZNAM PŘÍLOH

        - up.png
    - include
        - functions.php
        - hlavicka.php
        - menu.php
        - paticka.php
    - tinymce
    - clanek.php
    - clanek_pridat.php
    - clanek_smazat.php
    - clanek_upravit.php
    - hlavni_stranka.php
    - kategorie.php
    - kategorie_pridat.php
    - kategorie_smazat.php
    - kategorie_upravit.php
    - multi_clanek_pridat.php
    - multi_clanek_smazat.php
    - multi_clanek_upravit.php
    - nastaveni.php
    - uzivatele.php
    - uzivatele_pridat.php
    - uzivatele_smazat.php
    - uzivatele_upravit.php
    - wb.php
    - login.php
    - kontrola_login.php
    - odhlasit.php
    - spravce_souboru.php
    - upl_config.php
    - upl_func.php
    - index.php
    - default.css

- config
  - db_config.php
- css
  - main.css
  - normalize.min
  - .htaccess
- img
  - bg.png
  - logo-fai.png
- include
  - ClanekVKategoriiClass.php
  - ClankyClass.php
  - HlavniStrankaClass.php
  - KategorieClass.php
  - UzivateleClass.php
- js
- soubory
- clanek.php
- functions.php
- hledat.php
- kat.php
- index.php
- favicon.ico

PŘÍLOHA P I: INDEX.PHP

```
<?php
include('functions.php');
?>
<!doctype html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
        <meta name="keywords" content="Univerzita Tomáše Bati ve Zlíně, UTB, vy-
soká škola, univerzita, Zlín, FAI" />
        <meta name="description" content="Univerzita Tomáše Bati ve Zlíně (UTB)
je dynamicky se rozvíjející vysokou školou....">
        <meta name="author" content="Lukáš Chytílek" />
        <meta name="robots" content="index, follow" />
        <meta name="googlebot" content="snippet,archive" />
        <meta name="Content-language" content="cs" />
        <meta name="copyright" content="chytilek.cz" />

        <title>Informační systém studijního oddělení FAI, Univerzita Tomáše Bati
ve Zlíně</title>

        <link rel="stylesheet" href="css/main.css">
        <link rel="shortcut icon" href="favicon.ico">

    </head>
    <body>

      <div class="main_site">

        <header class="header">
          <a href="index.php" title="Fakulta aplikované informatiky, Univerzita
Tomáše Bati ve Zlíně" ><img src="img/logo-fai.png" alt="Fakulta aplikované infor-
matiky, Univerzita Tomáše Bati ve Zlíně" /></a>
          <form id="searchForm" method="post" action="hledat.php">
            <input type="search" name="search" placeholder="Hledaný výraz">
             <button>Hledat</button>
          </form>
        </header>

        <div class="cleaner"> </div>

        <div class="banner">
        </div>

        <div class="cleaner"> </div>

        <div class="main_content">
          <div class="left_side">
            <h3>Menu</h3>

            <?php echo CategoryList(); ?>
          </div>
          <div class="right_side">
            <article>
              <?php echo Main();?>
            </article>
          </div>
        </div>
        <div class="cleaner"> </div>
        <footer>
          <p>Informační systém studijního oddělení, Bakalářská práce 2015, Lukáš
Chytílek</p>
        </footer>
      </div>
    </body>
</html>
```

PŘÍLOHA P II: KAT.PHP

```
<?php include('functions.php'); ?>
<!doctype html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
        <meta name="keywords" content="Univerzita Tomáše Bati ve Zlíně, UTB, vy-
soká škola, univerzita, Zlín, FAI" />
        <meta name="description" content="Univerzita Tomáše Bati ve Zlíně (UTB)
je dynamicky se rozvíjející vysokou školou....">
        <meta name="author" content="Lukáš Chytílek" />
        <meta name="robots" content="index, follow" />
        <meta name="googlebot" content="snippet,archive" />
        <meta name="Content-language" content="cs" />
        <meta name="copyright" content="chytilek.cz" />

        <title>Informační systém studijního oddělení FAI, Univerzita Tomáše Bati
ve Zlíně</title>

        <link rel="stylesheet" href="css/main.css">
        <link rel="shortcut icon" href="favicon.ico">

    </head>
    <body>

      <div class="main_site">

        <header class="header">
          <a href="index.php" title="Fakulta aplikované informatiky, Univerzita
Tomáše Bati ve Zlíně" ><img src="img/logo-fai.png" alt="Fakulta aplikované infor-
matiky, Univerzita Tomáše Bati ve Zlíně" /></a>
          <form id="searchForm" method="post" action="hledat.php">
            <input type="search" name="search" placeholder="Hledaný výraz">

                    <button>Hledat</button>
          </form>
        </header>

        <div class="cleaner"> </div>

        <div class="banner">
        </div>

        <div class="cleaner"> </div>

        <div class="main_content">
          <div class="left_side">
            <h3>Menu</h3>

            <?php echo CategoryList(); ?>
          </div>

          <div class="right_side">
            <article>
              <?php echo CategoryNameList(); ?>
            </article>
          </div>
        </div>
        <div class="cleaner"> </div>
        <footer>
          <p>Informační systém studijního oddělení, Bakalářská práce 2015, Lukáš
Chytílek</p>
        </footer>
      </div>
    </body>
</html>
```

PŘÍLOHA P III: CLANEK.PHP

```
<?php include('functions.php'); ?>
<!doctype html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
        <meta name="keywords" content="Univerzita Tomáše Bati ve Zlíně, UTB, vy-
soká škola, univerzita, Zlín, FAI" />
        <meta name="description" content="Univerzita Tomáše Bati ve Zlíně (UTB)
je dynamicky se rozvíjející vysokou školou....">
        <meta name="author" content="Lukáš Chytílek" />
        <meta name="robots" content="index, follow" />
        <meta name="googlebot" content="snippet,archive" />
        <meta name="Content-language" content="cs" />
        <meta name="copyright" content="chytilek.cz" />

        <title>Informační systém studijního oddělení FAI, Univerzita Tomáše Bati
ve Zlíně</title>

        <link rel="stylesheet" href="css/main.css">
        <link rel="shortcut icon" href="favicon.ico">

    </head>
    <body>

      <div class="main_site">

        <header class="header">
          <a href="index.php" title="Fakulta aplikované informatiky, Univerzita
Tomáše Bati ve Zlíně" ><img src="img/logo-fai.png" alt="Fakulta aplikované infor-
matiky, Univerzita Tomáše Bati ve Zlíně" /></a>
          <form id="searchForm" method="post" action="hledat.php">
            <input type="search" name="search" placeholder="Hledaný výraz">

                    <button>Hledat</button>
          </form>
        </header>

        <div class="cleaner"> </div>

        <div class="banner">
        </div>

        <div class="cleaner"> </div>

        <div class="main_content">
          <div class="left_side">
            <h3>Menu</h3>

            <?php echo CategoryList(); ?>
          </div>

          <div class="right_side">
            <article>
              <?php echo ArticleList(); ?>
            </article>
          </div>
        </div>
        <div class="cleaner"> </div>
        <footer>
          <p>Informační systém studijního oddělení, Bakalářská práce 2015, Lukáš
Chytílek</p>
        </footer>
      </div>
    </body>
</html>
```

PŘÍLOHA P IV: HLEDAT.PHP

```
<?php include('functions.php'); ?>
<!doctype html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
        <meta name="keywords" content="Univerzita Tomáše Bati ve Zlíně, UTB, vy-
soká škola, univerzita, Zlín, FAI" />
        <meta name="description" content="Univerzita Tomáše Bati ve Zlíně (UTB)
je dynamicky se rozvíjející vysokou školou....">
        <meta name="author" content="Lukáš Chytílek" />
        <meta name="robots" content="index, follow" />
        <meta name="googlebot" content="snippet,archive" />
        <meta name="Content-language" content="cs" />
        <meta name="copyright" content="chytilek.cz" />

        <title>Informační systém studijního oddělení FAI, Univerzita Tomáše Bati
ve Zlíně</title>

        <link rel="stylesheet" href="css/main.css">
        <link rel="shortcut icon" href="favicon.ico">

    </head>
    <body>

      <div class="main_site">

        <header class="header">
          <a href="index.php" title="Fakulta aplikované informatiky, Univerzita
Tomáše Bati ve Zlíně" ><img src="img/logo-fai.png" alt="Fakulta aplikované infor-
matiky, Univerzita Tomáše Bati ve Zlíně" /></a>
          <form id="searchForm" method="post" action="hledat.php">
            <input type="search" name="search" placeholder="Hledaný výraz">

                    <button>Hledat</button>
          </form>
        </header>

        <div class="cleaner"> </div>

        <div class="banner">
        </div>

        <div class="cleaner"> </div>

        <div class="main_content">
          <div class="left_side">
            <h3>Menu</h3>
            <?php echo CategoryList(); ?>
          </div>
          <div class="right_side">
            <article>
              <p>
                <?php echo Search(); ?>
              </p>
            </article>
          </div>
        </div>
        <div class="cleaner"> </div>
        <footer>
          <p>Informační systém studijního oddělení, Bakalářská práce 2015, Lukáš
Chytílek</p>
        </footer>
      </div>
    </body>
</html>
```

PŘÍLOHA P V: FUNCTIONS.PHP

```
<?php
include('include/ClankyClass.php');
include('include/KategorieClass.php');
include('include/ClanekVKategoriiClass.php');
include('include/HlavniStrankaClass.php');

  function CategoryTree($list,$kat,$pocet)
  {
    $list   =   '<li   class="parents"><a   href="kat.php?id='.$kat->Id.'">'.$kat-
>Nazev.'</a></li>';

    $clanek = ClanekVKategorii::LoadAllByKatId($kat->Id);
    if($clanek){
      foreach($clanek as $clanky){
        $clanek = Clanky::Load($clanky->Clanek_id);
        if($clanek){
          $list   .=   '<li   class="childs"><a   href="clanek.php?cl_id='.$clanek-
>Id.'">'.$clanek->Nazev.'</a></li>';
        }
        $list .= $clanky->ClanekId;
      }
    }

    $clanky = Clanky::LoadAllByMultipleAndKatId($kat->Id);
    if($clanky){
      foreach($clanky as $clanek){
        $list   .=   '<li   class="childs"><a   href="clanek.php?cl_id='.$clanek-
>Id.'">'.$clanek->Nazev.'</a></li>';
      }
    }

    $kategorie = Kategorie::LoadAllByParoveId($kat->Id);
    if($kategorie){
      $pocet++;
      foreach($kategorie as $kat){
        $list .= CategoryTreeChild($list,$kat,$pocet);
      }
    }
    return $list;
  }

  function CategoryTreeChild($list,$kat,$pocet)
  {

    $list   =   '<li   class="childs"><a   href="kat.php?id='.$kat->Id.'">'.$kat-
>Nazev.'</a></li>';

    $clanekVkategorii = ClanekVKategorii::LoadAllByKatId($kat->Id);
    if($clanekVkategorii){
      $list .= '<ul class="child child">';
      foreach($clanekVkategorii as $clanky){
        $clanek = Clanky::Load($clanky->Clanek_id);
        if($clanek){
          $list   .=   '<li   class="childs"><a   href="clanek.php?cl_id='.$clanek-
Id.'">'.$clanek->Nazev.'</a></li>';
        }
      }
      $list .= '</ul>';
    }

    if(isset($_GET['cl_id'])){
      $newId = $_GET['cl_id'];
      $clanky = Clanky::LoadAllByMultipleAndKatId($kat->Id);
      if($clanky){
        foreach($clanky as $clanek){
```

```
          if($newId == $clanek->Id){
            $newClanky = Clanky::LoadAllByMultipleAndKatId($clanek->Kat_id);
            if($newClanky){
              $list .= '<ul class="child child">';
              foreach($newClanky as $newClanek){
                $list            .=            '<li            class="childs"><a
href="clanek.php?cl_id='.$newClanek->Id.'">'.$newClanek->Nazev.'</a></li>';
              }
              $list .= '</ul>';
            }
            $kategorie = Kategorie::LoadAllByParoveId($clanek->Kat_id);
            if($kategorie){
              $pocet++;
              $list .= "<ul class='child child".$pocet."'>";
              foreach($kategorie as $kat){
                $list .= CategoryTreeChild($list,$kat,$pocet);
              }
              $list .= "</ul>";
            }
          }
        }
      }

      if(isset($_GET['id'])){
        $newId = $_GET['id'];
        if($newId == $kat->Id){
          $clanky = Clanky::LoadAllByMultipleAndKatId($kat->Id);
          if($clanky){
            $list .= '<ul class="child child">';
            foreach($clanky as $clanek){
              $list  .=  '<li  class="childs"><a  href="clanek.php?cl_id='.$clanek-
>Id.'">'.$clanek->Nazev.'</a></li>';
            }
            $list .= '</ul>';
          }
          $kategorie = Kategorie::LoadAllByParoveId($kat->Id);
          if($kategorie){
            $pocet++;
            $list .= "<ul class='child child".$pocet."'>";
            foreach($kategorie as $kat){
              $list .= CategoryTreeChild($list,$kat,$pocet);
            }
            $list .= "</ul>";
          }
        }
      }
      return $list;
    }

    function CategoryList()
    {
      $list = "";

      $kategorie = Kategorie::LoadAllByParoveId(0);
      $mainlist = "<ul class='parent'>";
      if($kategorie){
        foreach($kategorie as $kat){
          $mainlist .= CategoryTree($list,$kat,$pocet = 0);
        }
      }
      $list .= "</ul>";
      return $mainlist;
    }

    function Main()
    {
        $hlavni = HlavniStranka::Load(0);
```

```
      echo $hlavni->Popis;
  }

  function CategoryNameList()
  {
    $id = $_GET['id'];
    $id=htmlspecialchars($id);
    $id=htmlentities($id, ENT_QUOTES);
    $kategorie = Kategorie::Load($id);
    if($kategorie){
      echo "<h2>".$kategorie->Nazev."</h2>";
      echo $kategorie->Popis;
      echo "<h4>Odkazy:</h4>";

      $clanek_list = ClanekVKategorii::LoadAllByKatId($id);
      if($clanek_list){
        foreach($clanek_list as $mlt_clanek){
          $clanek = Clanky::Load($mlt_clanek->Clanek_id);
          if($clanek){
            echo      "<a      href='clanek.php?cl_id=".$clanek->Id."'>".$clanek-
>Nazev."</a><br>";
          }
        }
      }

      $kategorie_list = Kategorie::LoadAllByParoveId($id);
      if($kategorie_list){
        foreach($kategorie_list as $kat){
          echo "<a href='kat.php?id=".$kat->Id."'>".$kat->Nazev."</a><br>";
        }
      }

      $clanky = Clanky::LoadAllByKatId($id);
      if($clanky){
        foreach($clanky as $clanek){
          echo       "<a       href='clanek.php?cl_id=".$clanek->Id."'>".$clanek-
>Nazev."</a><br>";
        }
      }
    }
  }

  function ArticleList()
  {
    $id = $_GET['cl_id'];
    $id=htmlspecialchars($id);
    $id=htmlentities($id, ENT_QUOTES);
    $clanek = Clanky::Load($id);
    if($clanek){
      echo "<h2>".$clanek->Nazev."</h2>";
      echo $clanek->Popis;
    }
  }

  function Search(){
    $hledat=$_POST['search'];
    if($hledat){
      $hledat=htmlspecialchars($hledat);
      $hledat=htmlentities($hledat, ENT_QUOTES);

      if(strlen($hledat) < 4){
        include "config/db_config.php";
        $hledatsql="SELECT * FROM clanky WHERE popis REGEXP '$hledat'";

      }
      else if(strlen($hledat) >= 4){
        include "config/db_config.php";
        $hledatsql="SELECT * FROM clanky WHERE MATCH(popis) AGAINST('$hledat') ";
```

```
      }
      $vypishledat=mysql_query($hledatsql);
      $hledatcisla=mysql_num_rows($vypishledat);

      $hledat=html_entity_decode($hledat);
      if ($hledatcisla) {
        echo "<h3>Zadaný výraz: ".$hledat."</h3>";
        echo "<strong>Nalezeno ".$hledatcisla." výsledků</strong> <br />";
        echo "<br />";
        while ($vypsathled=mysql_fetch_array($vypishledat)) {
          echo '<a
href="clanek.php?cl_id='.$vypsathled['id'].'">'.$vypsathled['nazev'].'</a>';
          echo "<br />";
        }
      }
      if ($hledatcisla == 0) {
        echo "<h3>Zadaný výraz: ".$hledat."</h3>";
        echo "<strong>Nalezeno 0 výsledků</strong> <br />";
        echo "<br />";
      }
    } else{
      echo "<h3>Zadaný výraz: ".$hledat."</h3>";
      echo "<strong>Nalezeno 0 výsledků</strong> <br />";
      echo "<br />";
    }
  }
?>
```

PŘÍLOHA P VI: INCLUDE/CLANEKVKATEGORIICLASS.PHP

```
<?php

class ClanekVKategorii {

	//objekt DB pripojeni
	protected static $Conn;

	//ORM promenne
	protected $id;
	protected $clanek_id;
	protected $kategorie_id;

	//konstruktor
	public function __construct(){

		try {
			//pripojeni k DB
			self::$Conn                    =                    new
PDO('mysql:host=localhost;dbname=d15275_bp2015', 'root', '');
			//hlaseni chyb DB
			self::$Conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

		} catch (PDOException $e){
			//odchytime vyjimku a vypiseme chybu pripojeni
			echo 'Error: '.$e->getMessage();
		}


	}

	//Load - nacte jeden radek z databaze
	public static function Load($intId){

		//nova instance tridy
		$Contact = new ClanekVKategorii();

		//statement - dotaz nad DB
		$stmt = self::$Conn->prepare('SELECT * FROM clanek_v_kategorii WHERE
id = :id');
		$stmt->setFetchMode(PDO::FETCH_CLASS, 'ClanekVKategorii');
		//provedeme pripraveny dotaz
		$stmt->execute(array('id'=>$intId));

		return $Contact = $stmt->fetch();
	}

	//LoadAll - nacte vsechny radky z tabulky
	public static function LoadAll(){

		//nova instance tridy
		$Contact = new ClanekVKategorii();

		//pole pro objekty
		$arrObjects = array();

		$stmt = self::$Conn->query('SELECT * FROM clanek_v_kategorii');
		$stmt->setFetchMode(PDO::FETCH_CLASS, 'ClanekVKategorii');

		while($Contact = $stmt->fetch()){

			array_push($arrObjects, $Contact);
		}

		return $arrObjects;
```

```
	}

  public static function LoadAllByKatId($intId){

		//nova instance tridy
		$Contact = new ClanekVKategorii();

		//pole pro objekty
		$arrObjects = array();

		$stmt = self::$Conn->query("SELECT * FROM clanek_v_kategorii WHERE
kategorie_id = '$intId' order by id desc");
		$stmt->setFetchMode(PDO::FETCH_CLASS, 'ClanekVKategorii');

		while($Contact = $stmt->fetch()){

			array_push($arrObjects, $Contact);
		}

		return $arrObjects;
	}

	//Save - ulozeni instance objektu do DB
	public function Save(){
		try {
			if($this->id){
	  $stmt   =   self::$Conn->prepare('UPDATE   clanek_v_kategorii   SET   cla-
nek_id=:clanekId, kategorie_id=:kategorieId WHERE id=:id');
	  $stmt->execute(array(
	    ':id' => $this->id,
	    ':clanekId' => $this->clanek_id,
				':kategorieId' => $this->kategorie_id,
	  ));

	  echo 'UPDATED!!!';

	} else{
			$stmt = self::$Conn->prepare('INSERT INTO clanek_v_kategorii
(clanek_id, kategorie_id) VALUES (:clanekId,:kategorieId)');
			$stmt->execute(array(
				':clanekId' => $this->clanek_id,
				':kategorieId' => $this->kategorie_id,
			));

			return self::$Conn->lastInsertId();
	}

		} catch (PDOException $e){

			echo 'Error: '.$e->getMessage();
		}

	}

  //Delete - smazani instance objektu do DB
	public function Delete(){
		try {

			$stmt = self::$Conn->prepare('DELETE FROM clanek_v_kategorii
WHERE id = :id');
			$stmt->execute(array(
				':id' => $this->id
			));

	echo 'DELETED! <br />';
			return $this->id;

		} catch (PDOException $e){
```

```
                echo 'Error: '.$e->getMessage();
        }

    }

    //getter - ziskani vlastnosti objektu
    public function __get($strName){

        switch ($strName) {

    case 'Id':
                    return $this->id;
                    break;

    case 'Clanek_id':
                    return $this->clanek_id;
                    break;

                case 'Kategorie_id':
                    return $this->kategorie_id;
                    break;

        }
    }

    //setter - nastaveni vlastnosti objektu
    public function __set($strName, $mixValue){

        switch($strName){

                case 'Clanek_id':
                    $this->clanek_id = $mixValue;
                    break;

                case 'Kategorie_id':
                    $this->kategorie_id = $mixValue;
                    break;

        }

    }

}
?>
```

PŘÍLOHA P VII: INCLUDE/CLANKYCLASS.PHP

```
<?php
class Clanky {
	//objekt DB pripojeni
	protected static $Conn;
	//ORM promenne
	protected $id;
	protected $nazev;
	protected $kat_id;
	protected $popis;
	protected $multi_clanek;

	//konstruktor
	public function __construct(){

		try {
			//pripojeni k DB
			self::$Conn = new
PDO('mysql:host=localhost;dbname=d15275_bp2015', 'root', '');
			//hlaseni chyb DB
			self::$Conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

		} catch (PDOException $e){
			//odchytime vyjimku a vypiseme chybu pripojeni
			echo 'Error: '.$e->getMessage();
		}

	}
	//Load - nacte jeden radek z databaze
	public static function Load($intId){

		//nova instance tridy
		$Contact = new Clanky();

		//statement - dotaz nad DB
		$stmt = self::$Conn->prepare('SELECT * FROM clanky WHERE id = :id');
		$stmt->setFetchMode(PDO::FETCH_CLASS, 'Clanky');
		//provedeme pripraveny dotaz
		$stmt->execute(array('id'=>$intId));

		return $Contact = $stmt->fetch();
	}
	//LoadAll - nacte vsechny radky z tabulky
	public static function LoadAll(){

		//nova instance tridy
		$Contact = new Clanky();

		//pole pro objekty
		$arrObjects = array();

		$stmt = self::$Conn->query('SELECT * FROM clanky');
		$stmt->setFetchMode(PDO::FETCH_CLASS, 'Clanky');

		while($Contact = $stmt->fetch()){

			array_push($arrObjects, $Contact);
		}

		return $arrObjects;
	}

  public static function LoadAllByKatId($intId){

		//nova instance tridy
```

```
            $Contact = new Clanky();

            //pole pro objekty
            $arrObjects = array();

            $stmt = self::$Conn->query("SELECT * FROM clanky WHERE kat_id =
'$intId'");
            $stmt->setFetchMode(PDO::FETCH_CLASS, 'Clanky');

            while($Contact = $stmt->fetch()){

                  array_push($arrObjects, $Contact);
            }

            return $arrObjects;
      }

  public static function LoadAllByMultipleAndKatId($intId){

            //nova instance tridy
            $Contact = new Clanky();

            //pole pro objekty
            $arrObjects = array();

            $stmt = self::$Conn->query("SELECT * FROM clanky WHERE mul-
ti_clanek='false' AND kat_id = '$intId'");
            $stmt->setFetchMode(PDO::FETCH_CLASS, 'Clanky');

            while($Contact = $stmt->fetch()){

                  array_push($arrObjects, $Contact);
            }

            return $arrObjects;
      }

      //Save - ulozeni instance objektu do DB
      public function Save(){
            try {
                  if($this->id){
        $stmt = self::$Conn->prepare('UPDATE clanky SET nazev=:nazev,
kat_id=:katId, popis=:popis, multi_clanek=:multiClanek WHERE id=:id');
        $stmt->execute(array(
          ':id' => $this->id,
          ':nazev' => $this->nazev,
                        ':katId' => $this->kat_id,
                        ':popis' => $this->popis,
                        ':multiClanek' => $this->multi_clanek
        ));

        echo 'UPDATED!!!';
      } else{
                  $stmt = self::$Conn->prepare('INSERT INTO clanky (nazev,
kat_id, popis, multi_clanek) VALUES (:nazev,:katId,:popis,:multiClanek)');
                  $stmt->execute(array(
                        ':nazev' => $this->nazev,
                        ':katId' => $this->kat_id,
                        ':popis' => $this->popis,
                        ':multiClanek' => $this->multi_clanek
                  ));

                  return self::$Conn->lastInsertId();
      }
            } catch (PDOException $e){

                  echo 'Error: '.$e->getMessage();
            }
```

```
	}

  //Delete - smazani instance objektu do DB
	public function Delete(){
		try {

			$stmt = self::$Conn->prepare('DELETE FROM clanky WHERE id =
:id');
			$stmt->execute(array(
				':id' => $this->id
			));
	echo 'DELETED! <br />';
			return $this->id;

		} catch (PDOException $e){

			echo 'Error: '.$e->getMessage();
		}

	}
	//getter - ziskani vlastnosti objektu
	public function __get($strName){

		switch ($strName) {

	case 'Id':
				return $this->id;
				break;

	case 'Nazev':
				return $this->nazev;
				break;

			case 'Kat_id':
				return $this->kat_id;
				break;

			case 'Popis':
				return $this->popis;
				break;

	case 'multiClanek':
				return $this->multi_clanek;
				break;
		}
	}
	//setter - nastaveni vlastnosti objektu
	public function __set($strName, $mixValue){

		switch($strName){

			case 'Nazev':
				$this->nazev = $mixValue;
				break;
			case 'Kat_id':
				$this->kat_id = $mixValue;
				break;
			case 'Popis':
				$this->popis = $mixValue;
				break;
	case 'multiClanek':
				$this->multi_clanek = $mixValue;
				break;
		}
	}
}
?>
```

PŘÍLOHA P VIII: INCLUDE/KATEGORIECLASS.PHP

```
<?php
class Kategorie {
	//objekt DB pripojeni
	protected static $Conn;
	//ORM promenne
	protected $id;
	protected $parove_id;
	protected $nazev;
	protected $popis;
	//konstruktor
	public function __construct(){
		try {
			//pripojeni k DB
			self::$Conn = new
PDO('mysql:host=localhost;dbname=d15275_bp2015', 'root', '');
			//hlaseni chyb DB
			self::$Conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

		} catch (PDOException $e){
			//odchytime vyjimku a vypiseme chybu pripojeni
			echo 'Error: '.$e->getMessage();
		}
	}
	//Load - nacte jeden radek z databaze
	public static function Load($intId){
		//nova instance tridy
		$Contact = new Kategorie();

		//statement - dotaz nad DB
		$stmt = self::$Conn->prepare('SELECT * FROM kategorie WHERE id =
:id');
		$stmt->setFetchMode(PDO::FETCH_CLASS, 'Kategorie');
		//provedeme pripraveny dotaz
		$stmt->execute(array('id'=>$intId));

		return $Contact = $stmt->fetch();
	}

  //LoadbyParoveId - nacte jeden radek z databaze podle parove_id
	public static function LoadByParoveId($intId){

		//nova instance tridy
		$Contact = new Kategorie();

		//statement - dotaz nad DB
		$stmt = self::$Conn->prepare('SELECT * FROM kategorie WHERE paro-
ve_id = :id');
		$stmt->setFetchMode(PDO::FETCH_CLASS, 'Kategorie');
		//provedeme pripraveny dotaz
		$stmt->execute(array('id'=>$intId));

		return $Contact = $stmt->fetch();
	}
	//LoadAll - nacte vsechny radky z tabulky
	public static function LoadAll(){
		//nova instance tridy
		$Contact = new Kategorie();
		//pole pro objekty
		$arrObjects = array();
		$stmt = self::$Conn->query('SELECT * FROM kategorie');
		$stmt->setFetchMode(PDO::FETCH_CLASS, 'Kategorie');

		while($Contact = $stmt->fetch()){
```

```
                array_push($arrObjects, $Contact);
            }

            return $arrObjects;
      }

  //LoadAll - nacte vsechny radky z tabulky
      public static function LoadAllByParoveId($intId){

            //nova instance tridy
            $Contact = new Kategorie();

            //pole pro objekty
            $arrObjects = array();

            $stmt = self::$Conn->query("SELECT * FROM kategorie WHERE paro-
ve_id='$intId'");
            $stmt->setFetchMode(PDO::FETCH_CLASS, 'Kategorie');

            while($Contact = $stmt->fetch()){

                  array_push($arrObjects, $Contact);
            }

            return $arrObjects;
      }

      //Save - ulozeni instance objektu do DB
      public function Save(){
            try {

                  $stmt = self::$Conn->prepare('INSERT INTO kategorie (paro-
ve_id, nazev, popis) VALUES (:paroveId,:nazev,:popis)');
                  $stmt->execute(array(
                        ':paroveId' => $this->parove_id,
         ':nazev' => $this->nazev,
                        ':popis' => $this->popis
                  ));

        echo          "<p          class='center'><strong          style='color:
#f00000;'>Vytvořeno...</strong></p>";
                  return self::$Conn->lastInsertId();

            } catch (PDOException $e){

                  echo 'Error: '.$e->getMessage();
            }

      }

  public function Update($intId){
            try {
       $stmt = self::$Conn->prepare('UPDATE kategorie SET parove_id=:paroveId
nazev=:nazev, popis=:popis WHERE id=:id');
        $stmt->execute(array(
          ':id' => $this->id,
                        ':paroveId' => $this->parove_id,
          ':nazev' => $this->nazev,
                        ':popis' => $this->popis
        ));

        echo 'UPDATED!!!';

            } catch (PDOException $e){

                  echo 'Error: '.$e->getMessage();
            }
```

```
	}

  //Delete - smazani instance objektu do DB
	public function Delete(){
		try {

			$stmt = self::$Conn->prepare('DELETE FROM kategorie WHERE id
= :id');
			$stmt->execute(array(
				':id' => $this->id
			));

	echo 'DELETED! <br />';
			return $this->id;

		} catch (PDOException $e){

			echo 'Error: '.$e->getMessage();
		}

	}

	//getter - ziskani vlastnosti objektu
	public function __get($strName){

		switch ($strName) {

	case 'Id':
				return $this->id;
				break;

	case 'Parove_id':
				return $this->parove_id;
				break;

			case 'Nazev':
				return $this->nazev;
				break;

			case 'Popis':
				return $this->popis;
				break;

		}
	}

	//setter - nastaveni vlastnosti objektu
	public function __set($strName, $mixValue){

		switch($strName){

			case 'Parove_id':
				$this->parove_id = $mixValue;
				break;

			case 'Nazev':
				$this->nazev = $mixValue;
				break;

			case 'Popis':
				$this->popis = $mixValue;
				break;

		}

	}
}
?>
```

PŘÍLOHA P IX: INCLUDE/HLAVNISTRANKACLASS.PHP

```
<?php

class HlavniStranka {

	//objekt DB pripojeni
	protected static $Conn;

	//ORM promenne
	protected $popis;

	//konstruktor
	public function __construct(){

		try {
			//pripojeni k DB
		  self::$Conn = new PDO('mysql:host=localhost;dbname=d15275_bp2015',
'root', '');
			//hlaseni chyb DB
			self::$Conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

		} catch (PDOException $e){
			//odchytime vyjimku a vypiseme chybu pripojeni
			echo 'Error: '.$e->getMessage();
		}


	}

	//Load - nacte jeden radek z databaze
	public static function Load($intId){

		//nova instance tridy
		$Contact = new HlavniStranka();

		//statement - dotaz nad DB
		$stmt = self::$Conn->prepare('SELECT * FROM hlavni_stranka WHERE id
= :id');
		$stmt->setFetchMode(PDO::FETCH_CLASS, 'HlavniStranka');
		//provedeme pripraveny dotaz
		$stmt->execute(array('id'=>$intId));

		return $Contact = $stmt->fetch();
	}

	//LoadAll - nacte vsechny radky z tabulky
	public static function LoadAll(){

		//nova instance tridy
		$Contact = new HlavniStranka();

		//pole pro objekty
		$arrObjects = array();

		$stmt = self::$Conn->query('SELECT * FROM hlavni_stranka');
		$stmt->setFetchMode(PDO::FETCH_CLASS, 'HlavniStranka');

		while($Contact = $stmt->fetch()){

			array_push($arrObjects, $Contact);
		}

		return $arrObjects;
	}
```

```
	//Save - ulozeni instance objektu do DB
	public function Save(){
		try {

	  $stmt = self::$Conn->prepare('UPDATE hlavni_stranka SET popis=:popis');
	  $stmt->execute(array(
				':popis' => $this->popis
	  ));

	  echo "<p class='center'><strong style='color:
#f00000;'>Upraveno...</strong></p>";

		} catch (PDOException $e){

			echo 'Error: '.$e->getMessage();
		}

	}

  //Delete - smazani instance objektu do DB
	public function Delete(){
		try {

			$stmt = self::$Conn->prepare('DELETE FROM hlavni_stranka
WHERE id = :id');
			$stmt->execute(array(
				':id' => $this->id
			));

	      echo 'DELETED! <br />';
			return $this->id;

		} catch (PDOException $e){

			echo 'Error: '.$e->getMessage();
		}

	}

	//getter - ziskani vlastnosti objektu
	public function __get($strName){

		switch ($strName) {

			case 'Popis':
				return $this->popis;
				break;

		}
	}

	//setter - nastaveni vlastnosti objektu
	public function __set($strName, $mixValue){

		switch($strName){

			case 'Popis':
				$this->popis = $mixValue;
				break;

		}

	}

}

?>
```

PŘÍLOHA P X: INCLUDE/UZIVATELECLASS.PHP

```
<?php

class Uzivatele {

	//objekt DB pripojeni
	protected static $Conn;

	//ORM promenne
	protected $id;
	protected $jmeno;
	protected $heslo;

	//konstruktor
	public function __construct(){

		try {
			//pripojeni k DB
			self::$Conn = new
PDO('mysql:host=localhost;dbname=d15275_bp2015', 'root', '');
			//hlaseni chyb DB
			self::$Conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

		} catch (PDOException $e){
			//odchytime vyjimku a vypiseme chybu pripojeni
			echo 'Error: '.$e->getMessage();
		}


	}

	//Load - nacte jeden radek z databaze
	public static function Load($intId){

		//nova instance tridy
		$Contact = new Uzivatele();

		//statement - dotaz nad DB
		$stmt = self::$Conn->prepare('SELECT * FROM uzivatele WHERE id =
:id');
		$stmt->setFetchMode(PDO::FETCH_CLASS, 'Uzivatele');
		//provedeme pripraveny dotaz
		$stmt->execute(array('id'=>$intId));

		return $Contact = $stmt->fetch();
	}

	//LoadByJmeno - nacte jeden radek z databaze podle jmena
	public static function LoadByJmeno($lblJmeno){

		//nova instance tridy
		$Contact = new Uzivatele();

		//statement - dotaz nad DB
		$stmt = self::$Conn->prepare('SELECT * FROM uzivatele WHERE jmeno =
:jmeno');
		$stmt->setFetchMode(PDO::FETCH_CLASS, 'Uzivatele');
		//provedeme pripraveny dotaz
		$stmt->execute(array('jmeno'=>$lblJmeno));

		return $Contact = $stmt->fetch();
	}

	//LoadAll - nacte vsechny radky z tabulky
	public static function LoadAll(){
```

```
            //nova instance tridy
            $Contact = new Clanky();

            //pole pro objekty
            $arrObjects = array();

            $stmt = self::$Conn->query('SELECT * FROM uzivatele');
            $stmt->setFetchMode(PDO::FETCH_CLASS, 'Uzivatele');

            while($Contact = $stmt->fetch()){

                  array_push($arrObjects, $Contact);
            }

            return $arrObjects;
      }

      //Save - ulozeni instance objektu do DB
      public function Save(){
            try {
                  if($this->id){
        $stmt = self::$Conn->prepare('UPDATE uzivatele SET jmeno=:jmeno, hes-
lo=:heslo WHERE id=:id');
        $stmt->execute(array(
          ':id' => $this->id,
          ':jmeno' => $this->jmeno,
                        ':heslo' => $this->heslo
        ));

        echo 'UPDATED!!!';

      } else{
                  $stmt = self::$Conn->prepare('INSERT INTO uzivatele (nazev,
kat_id, popis, multi_clanek) VALUES (:nazev,:katId,:popis,:multiClanek)');
                  $stmt->execute(array(
                        ':jmeno' => $this->jmeno,
                        ':heslo' => $this->heslo
                  ));

                  return self::$Conn->lastInsertId();
      }

            } catch (PDOException $e){

                  echo 'Error: '.$e->getMessage();
            }

      }

  //Delete - smazani instance objektu do DB
      public function Delete(){
            try {

                  $stmt = self::$Conn->prepare('DELETE FROM uzivatele WHERE id
= :id');
                  $stmt->execute(array(
                        ':id' => $this->id
                  ));

      echo 'DELETED! <br />';
                  return $this->id;

            } catch (PDOException $e){

                  echo 'Error: '.$e->getMessage();
            }
```

```
	}

	//getter - ziskani vlastnosti objektu
	public function __get($strName){

		switch ($strName) {

			case 'Jmeno':
				return $this->jmeno;
				break;

			case 'Heslo':
				return $this->heslo;
				break;

		}
	}

	//setter - nastaveni vlastnosti objektu
	public function __set($strName, $mixValue){

		switch($strName){

			case 'Jmeno':
				$this->jmeno = $mixValue;
				break;

			case 'Heslo':
				$this->heslo = $mixValue;
				break;

		}

	}

}
?>
```

PŘÍLOHA P XI: CSS/MAIN.CSS

```
body{
  width: 100%;
  height: 100%;
  margin 0;
  padding 0;
  font-family: Trebuchet MS;
  background-image: url(../img/bg.png);
}

.main_site{
  width: 1000px;
  margin: 0 auto;
}

.banner{
  height: 20px;
  text-align: center;
  font-style: italic;
  background-color: #FDC82F;
  color: #FDC82F;
}

.main_content{
  min-height: 500px;
  overflow:auto;
}

.left_side{
  width: 25%;
  float: left;
  height: 100%;
  display: block;
}

.left_side h3{
  text-align: center;
}

.parent{
  padding: 0;
  margin: 20px auto;
}

.parents{
       padding-top: 10px;
  padding-left: 10px;
  height: 30px;
       overflow: hidden;
       display: block;
       background: #fff;
       box-shadow: 1px 1px 2px rgba(0,0,0,0.2);
       margin-bottom: 4px;
       border-left: 10px solid #FDC82F;
}

.parents a{
  text-align: left;
  display: block;
  width: 100%;
  height: 100%;
  color: #333;
  position:relative;
  text-decoration: none;
}
```

```
.parents:hover{
  background: #FDC82F;
  font-weight: bold;
}

.child{
  padding: 0;
  margin: 5px auto;
  padding-left: 20px;
}

.childs{
  padding-left: 20px;
  list-style-type: none;
  margin: 5px auto;
}

.childs a{
text-decoration: none;
color: black;
}

.childs:hover{
  font-weight: bold;
}

.right_side{
  width: 72%;
  float: right;
  min-height: 500px;
  background-color: white;
  -webkit-border-radius: 7px;
  -moz-border-radius: 7px;
  border-radius: 7px;
}

.right_side article{
  padding-left: 20px;
  padding-right: 20px;
}

.right_side p{
  text-align: left;
}

.right_side h4{
  margin: 0;
  padding: 2px;
}

footer{
  height: 60px;
  -webkit-border-radius: 7px;
  -moz-border-radius: 7px;
  border-radius: 7px;
  background-color: #FDC82F;
  text-align: center;
  font-style: italic;
}

footer p{
  margin: 0;
  padding: 0;
  padding-top: 20px;
}

/* Formular */
```

```
#searchForm{
  float: right;
  margin-top: auto;
  margin-bottom: auto;
}

#searchForm input{
       float: left;
       font-size: 13px;
       font-weight: 400;
       height: 38px;
  width: 300px;
       margin: 0;
       padding: 0 10px;
  -webkit-border-radius: 0 7px 7px 0;
  -moz-border-radius: 0 7px 7px 0;
  border-radius: 7px 0 0 7px;
  border: 1px solid grey;
}

#searchForm button{
  -webkit-border-radius: 0 7px 7px 0;
  -moz-border-radius: 0 7px 7px 0;
  border-radius: 0 7px 7px 0;
  border: 1px solid grey;
  font-size: 13px;
  color: #ffffff;
  background: #595959;
  padding: 10px 20px 10px 20px;
  text-decoration: none;
  height: 40px;
}
```