Rozšíření systému správy studijní agendy

Extension of study agenda administration

Bc. František Bolf

Diplomová práce
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2009/2010

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

Jméno a příjmení:	**Bc. František BOLF**
Osobní číslo:	**A08458**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Rozšíření systému správy studijní agendy**

Zásady pro vypracování:

1. **Analýza stávajícího informačního systému.**
2. **Návrh úprav a doplnění stávajícího systému.**
3. **Vytvoření úprav systému a popis řešení.**
4. **Implementace systému.**
5. **Vyřešení zabezpečení a správy databáze.**
6. **Podpora uživatelů využívajících systém – helpdesk.**

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **RIORDAN, Rebecca M. Vytváříme relační databázové aplikace. Praha : Computer Press, 2000. 280 s. ISBN 80-7226-360-9.**
2. **LACKO, Lubomir. PHP a MySQL -- hotová řešení. Computer Press, 2006, ISBN: 80-251-1249-7.**
3. **Kolektiv autorů. PHP5, MySQL, Apache- vytváříme webové aplikace. Computer Press, 2006, ISBN: 80-251-1073-7.**
4. **PROKOPOVÁ, Zdenka. Databázové systémy MySQL+PHP. FAI UTB Zlín, s. 126, 2006, Vysokoškolská skripta. ISBN 80-7318-486-9.**
5. **SCHNEIDER, R.,D. MySQL - Oficiální průvodce tvorbou, správou a laděním databází. Grada, ISBN: 80-247-1516-3.**
6. **HAVLENKA, J. a kol. Vytváříme WWW stránky a spravujeme moderní web site. Computer Press, 2006, ISBN: 80-251-0801-5.**
7. **KOSEK, J. HTML - tvorba dokonalých WWW stránek. Grada Publishing, 2006, ISBN: 80-7169-608-0.**
8. **ULLMAN, L. PHP a MySQL. Computer Press, Brno, 2004, ISBN: 80-251-0063-4.**

Vedoucí diplomové práce: **doc. Ing. Zdenka Prokopová, CSc.**
Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce: **19. února 2010**

Termín odevzdání diplomové práce: **8. června 2010**

Ve Zlíně dne 19. února 2010

prof. Ing. Vladimír Vašek, CSc.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Cílem diplomové práce bylo vytvoření webové aplikace, sloužící ke správě státních závěrečných zkoušek. Práce vychází ze stávajícího informačního systému, který byl vytvořen v rámci bakalářské práce. Systém byl doplněn o další funkcionality jako import a export dat, zálohu databáze, registraci a správu uživatelů, generování tiskových sestav, jmenovacích dekretů apod.

Stěžejním prvkem aplikace je databáze vytvořená v prostředí MySQL. Jako programové prostředí byl vybrán skriptovací jazyk PHP. Z důvodu kladení důrazu na logické řešení prvků byly použity technologie jako XHTML, CSS a JavaScript, sloužící k přesnému definování a zobrazování jednotlivých prvků.

Klíčová slova: webová aplikace, databáze, MySQL, PHP, XHTML, CSS, JavaScript

ABSTRACT

The aim of this thesis was creating a web application for administration of final degree exams. This work is based on previous information system, which was created as a part of bachelors work. New functionalities like import and export, database backup, user registration and administration, print outputs and decree outputs were added.

Essential part of the application is a database created in MySQL, code was written in script language PHP. Due to logical component structure emphasis, technologies like XHTML, CSS and JavaScript were used to achieve precise definition and representation of each element.

Keywords: web application, database, MySQL, PHP, XHTML, CSS, JavaScript

Rád bych touto cestou poděkoval vedoucí diplomové práce doc. Ing. Zdence Prokopové, CSc. za odborné vedení, rady, připomínky a pomoc v průběhu řešení této diplomové práce.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/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 diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/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 – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/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 diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/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 diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, 7. 6. 2010 …….

podpis diplomanta

OBSAH

ÚVOD

Dnes a denně se setkáváme s obrovským množstvím informací. Pojmeme-li to odborněji, to s čím přicházíme do styku, jsou pouhá data, která pro nás mohou, ale nemusejí mít zásadní význam. Informacemi se tato data stávají až poté, co pro nás mají konkrétní význam a můžeme z nich zároveň vyvozovat požadované znalosti. Při pokusu nasbírat všechny informace sami, bychom určitě po nějaké době narazili na problém, že ke sběru a uchování dat budeme nuceni využít nějakého většího celku. Tento problém je řešitelný pomocí informačního systému, který slouží ke sběru, uchovávání a zpracování dat, a to vše může být využito více uživateli zároveň. Jak je z popisu patrné, pod pojmem informační systém si můžeme představit např. kartotéku, telefonní seznam, účetnictví či databázi. Většina dnešních systému je založena na SŘBD (systému řízení báze dat), neboli databázovém systému a není tomu jinak ani v této diplomové práci.

Před samotnou implementací informačního systému je samozřejmě důležité provést analýzu požadavků, případně analýzu původního informačního systému. Po provedení takovéto analýzy, se nám do rukou dostávají informace, které vytvářejí stěžejní prvek k tvorbě či vylepšení stávajícího informačního systému.

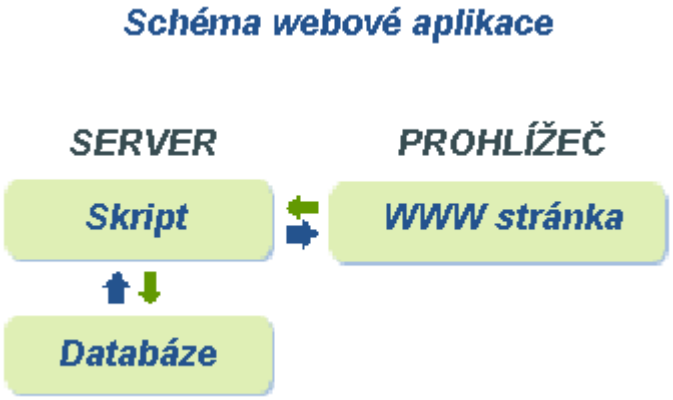
Cílem této práce je vytvoření systému pro správu studijní agendy, sloužícího k vytváření rozpisů studentů pro státní závěrečné zkoušky, a to jak v bakalářském, tak i v magisterském studijním programu. Systém je vytvořen pro zaměstnance Univerzity Tomáše Bati ve Zlíně konkrétně pro Fakultu aplikované informatiky. Práce má charakter internetové aplikace, neboť jedním z požadavků bylo umístění systému na univerzitní web. Tím je zaručeno, že po přihlášení do systému jsou uživatelé schopni spravovat svou agendu téměř kdekoliv.

Práce se skládá ze dvou částí, v první (teoretické) části se zaměřuji na popis nástrojů použitých pro vytvoření systému. V druhé (praktické) části popisuji postup vytváření a realizaci samotné webové aplikace. Závěrem zhodnocuji přínos práce a zavedení systému do provozu.

I. TEORETICKÁ ČÁST

1 TVORBA WEBOVÝCH APLIKACÍ

Dřívější aplikace typu klient-server měly každá svůj specifický klientský program, který byl nainstalován na jednotlivých uživatelských počítačích a představoval uživatelské rozhraní aplikace. Pokud došlo k aktualizaci serverové části, bylo nutno provést aktualizaci i na jednotlivých pracovních stanicích, čímž se rapidně snižovala efektivita práce zaměstnanců. Z toho důvodu byly vyvinuty webové aplikace sestávající ze třech hlavních součástí: klient, server a síť. V tomto případě je aplikace poskytována uživatelům z webového serveru, přes počítačovou síť Internet, případně intranet. Výhoda tohoto řešení spočívá v tom, že k provedení aktualizace je nutno přehrát pouze serverovou část aplikace a uživatelé si následně mohou pomocí webového prohlížeče zobrazit již aktualizovanou aplikaci. Tato vlastnost je bezesporu hlavním důvodem velké oblíbenosti webových aplikací.



Obr. 1 Schéma webové aplikace

1.1 Struktura

Webové aplikace se většinou skládají ze tří základních vrstev:

- Prezentační
- Logické
- Datové

Prezentační vrstva je obvykle reprezentována webovým prohlížečem a obsahuje funkce uživatelského rozhraní. Existuje několik prezentačních vrstev pro různé druhy zařízení, platformy a prostředí.

Logická vrstva přebírá data zadaná do prezentační vrstvy a posílá je do vrstvy datové, tvoří tzv. prostředníka. V této vrstvě dochází k transformaci dat mezi vstupně-výstupními požadavky a datovou vrstvou.

Datová vrstva obsahuje funkce pro přístup k informacím nacházejícím se v datovém úložišti. [8]

1.2 Používané technologie

Jaké technologie se při tvorbě webu použijí, záleží z velké části na výsledném obsahu. Pokud se bude jednat pouze o statické stránky, které budou mít svůj stálý obsah a nebudou často aktualizovány, vystačí si jejich autor většinou s jazykem HTML či XHTLM. Pokud ale bude mít web charakter dynamických stránek, které budou reagovat na uživatelovy pokyny, je nutné ke kódu vytvořeném v HTML či XHTLM použít skriptky naprogramované v jiných jazycích, jako PHP, ASP, JavaScript apod. Zvolení správné technologie je závislé na budoucím použití webu, samozřejmě každá technologie má svá pro a proti. Obecně můžeme programovací technologie rozdělit na ty, které se nacházejí na straně klienta a na straně serveru. Lze říci, že jejich vzájemné použití je ku prospěchu webu.

Vezměme v úvahu například situaci přidávání formuláře na webový server pro shromažďování dat ukládaných v databázi. Je zřejmé, že provedení kontroly formuláře zajišťující správnost informací zadaných uživatelem by bylo vhodnější na straně klienta, protože by nebylo nutné vložená data pro jejich kontrolu odesílat na server a zpět. Programování na straně klienta ověřování formuláře urychluje a zefektivňuje. Vkládání dat do databáze by naproti tomu bylo vhodnější provádět technologií na straně serveru za předpokladu, že je databáze umístěná na serveru.

Hlavní nevýhodou programování na straně serveru je samozřejmě nedostatek rychlosti z důvodu odesílání dat na server a zpět. Programy spuštěné na straně klienta se ve většině případů zdají uživateli poměrně rychlé, avšak programování na straně klienta má jednu vážnou nevýhodu a tou je nedostatek kontroly. Uživatel může uprostřed návštěvy zakázat podporu skriptování s čímž je nutné počítat a zavést vhodná opatření i na straně serveru. Pokud například budeme ověřovat formulář na straně klienta, je nutné kontrolu provést i na straně serveru. [5]

2 HTML

Hyper Text Markup Language, označovaný zkratkou HTML, je značkovací jazyk pro hypertext. Je jedním z jazyků pro vytváření stránek systému World Wide Web, který umožňuje publikaci dokumentů na Internetu. Definuje syntaxi a rozmístění speciálních vložených příkazů, které se v prohlížeči přímo nezobrazují, ale které řídí způsob zobrazení obsahu dokumentu, včetně textu, obrázků a ostatních podpůrných médií. Jazyk současně umožňuje vytváření interaktivních dokumentů, a to pomocí speciálních hypertextových odkazů, které propojují daný dokument s jinými dokumenty. [6], [9]

2.1 Historie HTML

Počátky jazyka HTML sahají do konce osmdesátých let minulého století, kdy v roce 1989 spolupracovali Tim Berners-Lee a Robert Caillau na propojeném informačním systému pro CERN (výzkumné centrum fyziky poblíž Ženevy ve Švýcarsku). V té době se pro tvorbu dokumentů obvykle používaly jazyky TeX, PostScript a také SGML. Berners-Lee si uvědomoval, že potřebují něco jednoduššího a v roce 1990 byl tedy navržen jazyk HTML (vycházející z rozsáhlého univerzálního značkovacího jazyka SGML) a protokol pro jeho přenos v počítačové síti HTTP. V roce 1991 CERN zprovoznil svůj web. Současně organizace NCSA (National Center for Supercomputer Applications) vybídla Marca Andreessena a Erica Binu k vytvoření prohlížeče Mosaic. Ten vznikl v roce 1993 ve verzích pro počítače IBM a Mackintosh a zaznamenal obrovský úspěch. Byl to totiž první prohlížeč s grafickým uživatelským rozhraním. [9]

2.2 Struktura HTML dokumentu

Od textového dokumentu se HTML dokument liší tím, že obsahuje kromě vlastního obsahu stránky navíc informace o vzhledu a formátování stránky. Tyto informace jsou určeny pro prohlížeče a předávají se pomocí značek uzavřených mezi znaky < >, které jsou mimo jiné nazývány jako tagy.

K tomu aby se z textového dokumentu stal HTML dokument, nám ovšem samotné značky nestačí. Stránka musí mít totiž přesně danou strukturu tvořenou specifickými formátovacími značkami, vymezujícími určité oddíly HTML dokumentu. Zde patří značky <*html*>, <*head*> a <*body*>. Ve verzi HTML 4.01 je navíc podle [10] nutné uvést Deklaraci DTD vyjadřovanou direktivou <*!DOCTYPE*>.

Jednoduchý příklad struktury dokumentu:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
        "http://www.w3.org/TR/html4/strict.dtd">
<HTML>
<HEAD>
<TITLE>Příklad titulku</TITLE>
        … ostatní elementy hlavičky …
</HEAD>
<BODY>
        … tělo dokumentu …
</BODY>
</HTML>
```

2.2.1 Deklarace DTD

Specifikace DTD (Document Type Definition) se píše úplně na začátek kódu a usnadňuje identifikaci HTML dokumentu a jeho verze. Také se zde upřesňuje vykreslovací mód, podle kterého se budou stránky zobrazovat.

Existují tři typy dokumentů:

- HTML 4.01 Strict

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
         "http://www.w3.org/TR/html4/strict.dtd">
```

- HTML 4.01 Transitional

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
         "http://www.w3.org/TR/html4/loose.dtd">
```

- HTML 4.01 Frameset

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
          "http://www.w3.org/TR/html4/frameset.dtd">
```

Dokument typu *Strict* neobsahuje zavržené a prezentační elementy (jako je *font*, *center* nebo *u*). Striktní dokument se používá k psaní pouze stoprocentně čistého zdrojového kódu. *Transitional* obsahuje všechny elementy a parametry typu *Strict*, ale navíc podporuje i starší elementy, které se nenacházejí v předchozí verzi. Dokument typu *Frameset* vychází z typu *Transitional*, ale navíc podporuje rámy.

2.2.2 Značka <html>

Slouží k ohraničení celého HTML dokumentu. Přesně udává prohlížeči, kde začíná a končí zdrojový kód zobrazované stránky. Začátek a konec dokumentu je reprezentován tagy *<html> </html>*.

2.2.3 Značka <head>

Značky *<head> </head>* slouží k vymezení záhlaví HTML dokumentu. Obsah hlavičky se v dokumentu nezobrazuje, ale jsou v ní umístěny některé důležité informace jako název stránky, popis jejího obsahu, definice stylů, metadata a mnoho dalších. Nejdůležitějšími značkami nacházejícími se v hlavičce jsou *<title> </title>*, do kterých se uzavírá nadpis stránky, sloužící k zobrazení titulku okna prohlížeče nebo záložky.

2.2.4 Značka <body>

Hlavní obsah stránky neboli tělo je ohraničeno značkami *<body> </body>*. Mezi ně se zapisuje text, který se má na stránce zobrazit.

3 XHTML

XHTML je přetvoření jazyka HTML pomocí XML (Extensible Markup Language). XHTML řeší dva základní problémy spojení s HTML. Nutí totiž designéry dále oddělit vzhled dokumentu od jeho struktury za pomocí použití formátovacích sad. Druhým problémem je to, že XHTML přináší webovým stránkám daleko přísnější užití pravidel pro značky.

Syntaktická přesnost jazyka XHTML je jeho největším přínosem a zároveň největší slabostí, avšak všeobecně se tvorba takovéhoto dokumentu z velké části v ničem neliší od vytváření běžného dokumentu HTML.

3.1 Rozdíly XHTML oproti HTML

- V XHTML na rozdíl od HTML musí být všechny tagy ukončené a to včetně nepárových jako jsou <*meta*>, <*link*>, <*br*>, <*hr*> nebo <*img*>. Jejich zápis může mít více podob, avšak nejvhodnějším je tento <*img /*>.
- Všechny tagy a jejich atributy musejí být zapsány malými písmeny a to z toho důvodu, že jsou takto deklarovány ve specifikaci DTD a XHTML je case sensitive.
- Všechny hodnoty atributů musejí být uzavřeny do uvozovek.
- Dokument musí začínat XML deklarací.
- Při práci s rámy můžeme deklarovat XHTML 1.0 Frameset, jinak stránky deklarujeme jako XHTML 1.0 Transitional.

Citováno z [11].

4 CSS

Kaskádové styly, neboli anglicky Cascading Style Sheets (CSS) představují skutečný přelom ve způsobu formátování webových prezentací. CSS neovlivňují obsah dokumentů samotných, umožňují vytvářet čisté, přísně strukturované dokumenty, vyhovující standardům a umožňují bezproblémovou práci s obsahem stránek i jiných webových aplikací.

Jazyk byl navržen standardizační organizací W3C. CSS vznikl v roce 1996 a to hlavně v důsledku velkých nároků na HTML jazyk, které nedokázal splnit. Výhodou CSS oproti formátování v HTML je, že kód a obsah webu je uložen v souboru .html a veškerý design a formátování se načítá z jednoho souboru .css, který je většinou společný pro celý web. To znamená, že pokud máme v plánu změnu designu webu, stačí změnit pouze jeden soubor .css a změna se aplikuje na celý web. Výhodou také je, že se soubor CSS uloží do mezi-paměti prohlížeče a pokud není změněn, tak se načítá pouze jednou a zobrazení stránek se tím velmi urychlí. [7]

4.1 Verze CSS

Existují tři úrovně kaskádových stylů, první dvě patří mezi aktuální verze, zatímco třetí je stále vyvíjena. Rozdíly mezi těmito úrovněmi jsou následující:

- CSS1 definuje základní funkce stylů s omezenou podporou písmen a nastavování pozic prvků.
- CSS2 obsahuje navíc sluchové vlastnosti, stránkovaná média a lepší podporu písem a nastavování pozic prvků. Rovněž byla vylepšena spousta dalších vlastností.
- CSS3 obsahuje navíc vlastnosti určené pro prezentaci, díky čemuž je možné z webových dokumentů efektivně vytvářet prezentace (podobné prezentacím programu PowerPoint).

Verzi CSS, kterou hodláme používat, není třeba nijak definovat. Jediné na co si je potřeba dávat pozor je dostatečná kompatibilita mezi jednotlivými prohlížeči, neboť ne všechny zobrazují styly stejně. Proto je důležité před nasazením implementace do ostrého provozu otestovat chování na všech typech prohlížečů. [12]

4.2 Výhody užití CSS

Před nástupem kaskádových stylů se HTML dokumenty formátovaly pomocí značek a jejich atributů. K rozvržení stránky se používaly výhradně tabulky, což přinášelo spoustu problémů pro jiná zařízení, nežli běžné počítače. Datový objem formátovacích značek a atributů byl často vyšší, než objem vlastního obsahu, a tím docházelo ke zpomalení načítání stránek a ke zbytečnému nárůstu zatížení serverů.

Formátování dokumentů pomocí CSS tyto problémy odstraňuje. Umožňuje vytvořit čistý XHTML dokument, plně vyhovující současným standardům a hlavně přístupný všem aplikacím, jako jsou webové prohlížeče, mobilní telefony či zvuková zařízení. CSS nabízí mimo jiné mnohem širší možnosti formátování a některé vlastnosti klasický způsob formátování pomocí HTML ani neumožňuje.

Používání CSS přináší mnoho praktických výhod:

- Širší možnosti formátování.
- Snadnou tvorbu a údržbu stylů.
- Dynamickou práci se styly.
- Dopřednou kompatibilitu.
- Možnosti řízení tisku.
- Formátování XML dokumentů.

Citováno z [17].

4.3 Připojení šablony stylů k dokumentu (X)HTML

Existuje mnoho způsobů, pomocí nichž lze implementovat kaskádové styly do (X)HTML dokumentu. Níže si uvedeme nejpoužívanější typy vzájemného propojení. Zkratka (X)HTML zde reprezentuje jak jazyk HTML, tak i XHTML.

4.3.1 Značka <link>

Připojení stylu CSS k dokumentu (X)HTML pomocí značky <*link*> patří k nejpoužívanější a zároveň nejlepší možnosti. Účel této značky je propojit jeden dokument s druhým.

Příklad propojení:

```
<head>
   <link rel='stylesheet' type='text/css' href='styly.css'>
</head>
```

Tento zápis slouží k připojení externího souboru s názvem *styly.css* k dokumentu (X)HTML. Pro správné připojení stylu, je nutné umístit značku <*link*> do hlavičky dokumentu.

4.3.2 Připojení více souborů CSS

K jednomu dokumentu (X)HTML lze samozřejmě připojit více externích souborů obsahující kaskádové styly. Jako v předchozím případě, nám k tomu poslouží značka <*link*>.

4.3.3 Element *style*

Slouží k vložení informace o stylu CSS přímo do stránky, v níž pracuje. Tento způsob se avšak nedoporučuje, neboť styl nemůže odkazovat na jiné stránky a také znepřehlední výsledný dokument. Z toho vyplývá, že uložení stylů do zvláštního souboru poskytuje mnohem lepší flexibilitu.

Existují dva způsoby zápisu:

- Inline zápis

```
<p style="color: blue; text-decoration: underline">Text bude modrý a
podtržený.</p>
```

Použitím tohoto zápisu je styl aplikován pouze na text v dotyčném elementu.

- Zápis stylu na úrovni dokumentu

```
<style  type="text/css" >
     h1, h2, h3, p { text-align: left;}
     h1.zeleny { color: #00FF00;}
</style>
```

Styl se zapisuje mezi hlavičku dokumentu a je aplikován na celou stránku.

5 JAVASCRIPT

JavaScript je předním skriptovacím jazykem na straně klienta používaným ve webovém prohlížeči. Jazyk byl původně vyvinut společností Netscape pro aplikaci Navigator 2.0, během let se rozšířil a v určité formě je podporován většinou hlavních prohlížečů.

JavaScript je volně psaný skriptovací jazyk, který má využití pro úlohy, jako je ověření formulářů, implementace systémů navigace a přidání zvláštních efektů stránek.

Zde je uveden jednoduchý příklad dialogového hlášení vytvořeného pomocí jazyka JavaScript.

```
<script  type="text/javascript" >
    alert("Dialogové hlášení!");
</script>
```

5.1 Implementace jazyka JavaScript

Existují čtyři základní způsoby, jak skript zahrnout do dokumentu HTML:

- Pomocí značky <*skript*>.
- Jako připojený soubor prostřednictvím atributu *src* značky <*skript*>.
- Pomocí atributu popisovače událostí HTML, jako je *onclick*.
- Pomocí pseudo-URL skriptu *javascript:* (k syntaxi vede odkaz).

Poslední ze zde uvedených způsobů avšak není doporučováno používat, může totiž dojít k chybě načítání, pokud uživatel použití JavaScriptu zakáže.

Citováno z [5].

6 RELAČNÍ DATABÁZE A MYSQL

MySQL je výkonný a robustní relační databázový systému, který slouží k ukládání a přebírání dat. Používá SQL (Structured Query Language), což je celosvětově používaný standardní dotazovací jazyk pro databáze. Umožňuje rychle, výkonně a s minimálními náklady ukládat různé typy dat, například logické operátory, texty, čísla, obrázky, binární číslice nebo objekty BLOB. Základem databáze MySQL jsou entity a relace. Entita je vyjádřena jako věc, o které se bude v databázi uchovávat informace. Relace vyjadřuje vztahy mezi vzniklými entitami. MySQL je nyní k dispozici pod licencí Open Source, ale v případě potřeby existují i komerční licence.

Citováno z [13], [14].

Obr. 2 Logo MySQL

6.1 Uložení dat v relačních databázích

Data v relačních databázích jsou uchovávána ve formě tabulek. Jednotlivé údaje jsou uchovávány v polích, které jsou uspořádány do řádků a sloupců. Řádky zde reprezentují jednotlivé datové položky a sloupce neboli atributy udávají vlastnosti polí. Každý řádek tabulky by měl být identifikovatelný pomocí primárního klíče. Z toho vyplývá, že primární klíč je vybraný sloupec tabulky, který slouží k jednoznačné identifikaci řádků. Primární klíč je z tohoto ohledu nejdůležitějším prvkem tabulky, neboť bez něj nelze vytvořit případnou relaci s jinou tabulkou. Relace s jinou tabulkou lze vytvořit propojením primárního klíče jedné tabulky a cizího klíče tabulky druhé. Cizích klíčů může být v tabulce obsaženo vícero.

6.2 Relace mezi databázovými tabulkami

Jak jsem se již zmínil v předchozím odstavci, relace je ta vlastnost, která naší databází dodává flexibilitu a lze ji nazval alfou a omegou pro vytvoření kvalitní a efektivní databáze. Relace tedy vyjadřuje vztahy mezi jednotlivými tabulkami a je definováno několik druhů relací.

6.2.1 Relace 1:1 (jedna ku jedné)

Patří mezi nejjednodušší relační vztahy. Je vyjádřena, jako propojení jednoho řádku první tabulky s jedním řádkem tabulky druhé. V praxi se používá k rozdělení jedné tabulky na dvě části. Jako příklad si uveďme tabulku obsahující informace o zaměstnancích firmy. V tabulce budou uvedeny osobní údaje zaměstnanců, jako jméno, příjmení, bydliště apod., ale také údaje o jeho postavení ve firmě, výši platu a jiné. Takovouto tabulku můžeme jednoduše rozdělit na dvě, kde jedna bude obsahovat osobní údaje zaměstnanců a druhá bude obsahovat jejich firemní data. Ke vzájemnému propojení tabulek se použijí jednoznačné identifikační údaje zaměstnanců neboli jejich primární klíče.

6.2.2 Relace 1:N (jedna ku více)

Relace je vyjádřena, jako propojení jednoho řádku první tabulky s více řádky tabulky druhé. Jako příklad si můžeme uvést uživatele internetového obchodu provádějícího objednávky zboží. Každý uživatel totiž může provést několik objednávek, což přesně reprezentuje relaci 1:N. Uživatel má status primárního klíče a jednotlivé objednávky jsou vyjádřeny klíči cizími.

6.2.3 Relace N:M (více ku více)

Jak je z názvu patrné, relace N:M propojuje více řádků první tabulky s více řádky tabulky druhé. V praxi je tato relace ve většině případů reprezentována pomocí tabulek tří, kdy obě krajní tabulky obsahují data, a prostřední tabulka slouží jako vazební. Tento vztah je vyjádřen jako 1:N:M:1. Pro příklad si zvolím známou formuli, kdy jeden autor může napsat více knih a zároveň jedna kniha může být napsána více autory.

6.3 Normalizace

Normalizace je technika používaná pro vytvoření sady tabulek s minimální redundancí, která podporuje datové požadavky organizace. Existuje několik normálních forem, nejužívanější z nich jsou první normální forma (1NF), druhá normální forma (2NF) a třetí normální forma (3NF). Všechny tyto formy jsou založeny na vztazích mezi sloupci tabulky. Z dlouhodobého hlediska je tvorba dobře navržených tabulek klíčová k efektivně a správně fungující databázi. [1], [2], [4]

6.3.1 První normální forma (1NF)

Tabulka splňuje první normální formu tehdy, pokud všechny sloupce jsou atomické, neboli dále nedělitelné. Jeden sloupec tak nesmí obsahovat více druhů dat. Je to normální forma, která je nezbytně důležitá pro vytvoření vhodných tabulek pro relační databáze.

6.3.2 Druhá normální forma (2NF)

Pro to aby byla splněna podmínka druhé normální formy, je nezbytně nutné také splnění první normální formy a zároveň musí být každý sloupec v tabulce zcela závislý na primárním klíči. Z toho automaticky vyplývá, že tabulka, která má pouze jeden primární klíč automaticky splňuje 2NF, ale tabulka obsahující více primárních klíčů musí být rozdělena na dvě a více tabulek, tak aby byla splněna definice druhé normální formy.

6.3.3 Třetí normální forma (3NF)

Tabulka je ve třetí normální formě tehdy, pokud splňuje první a zároveň druhou normální formu a žádný sloupec, který nepatří do primárního klíče, není tranzitivně závislý na primárním klíči. Abychom zrušili tuto závislost, musíme jako v předchozím případě, rozdělit tabulku do dvou a více tabulek tak, aby všechny neklíčové atributy byly na sobě nezávislé.

6.4 Jazyk SQL a databázový systém MySQL

Jazyk SQL je nejrozšířenějším komerčním databázovým jazykem. Vznikl na základě projektu společnosti IBM, jehož cílem bylo vytvořit jazyk blízký angličtině pro práci s daty v databázích.

Databázový jazyk umožňuje uživateli:

- Vytvářet struktury databází a tabulek.
- Provádět údržbu dat, např. vkládání, modifikaci a vymazávání dat.
- Provádět jednoduché i složité dotazy.

Jazyk SQL má dvě hlavní součásti:

- Jazyk pro definici dat (Data Definition Language, DDL). Pomocí příkazů z této skupiny můžeme definovat, vytvářet, měnit a rušit různé databázové struktury. Patří zde příkazy typu: *CRATE, ALTER, DROP*.
- Jazyk pro manipulaci dat (Data Manipulation Language, DML). Jak již z názvu vyplývá, patří do této skupiny příkazy pro manipulaci s daty, tedy příkazy pro výběr, vkládání, aktualizaci a mazání dat, což jsou tyto typy příkazů: *SELECT, INSERT, UPDATE, DELETE*.

Citováno z [2], [4].

6.4.1 Tvorba databází a tabulek

K vytváření databází a tabulek se většinou používá program pro správu databází, zde můžeme jmenovat např. program phpMyAdmin. Méně využívanou možností je psaní příkazů ručně, pomocí klienta mysql, neboť je tato možnost pracná a náchylná na chybné formulování příkazů.

Prvním krokem je vytvoření samotné databáze. Syntaxe příkazu není nijak složitá.

```
CREATE DATABASE nazev_databaze;
```

Po zadání výše uvedeného příkazu je nutné vytvořenou databázi nastavit jako aktivní.

```
USE nazev_databaze;
```

Nyní již můžeme libovolně vytvářet tabulky.

```
CREATE TABLE nazev_tabulky ( definice_tabulky );
```

Hlavní částí příkazu je definice tabulky, sloužící ke specifikaci (názvu, datového typu a atributů) jednotlivých sloupců tabulky.

MySQL obsahuje spoustu datových typů, které lze v základu rozdělit jako [14]:

- Číselné (*INTEGER, DECIMAL, NUMERIC, FLOAT, DOUBLE*, …)

- Textové (*CHAR, VARCHAR, BLOB, TEXT, ENUM, SET*, …)
- Datum a čas (*DATETIME, DATE, TIME, TIMESTAMP*, …)

6.4.2 Práce s daty

S příkazy pracujícími s daty se vývojáři databázových aplikací setkávají nejčastěji. Lze pomocí nich vkládat, aktualizovat, mazat a samozřejmě vybírat data. [14]

Příkaz SELECT

Je zdaleka nejpoužívanějším příkazem jazyka SQL. Slouží k výběru dat z jedné nebo i více tabulek. Jeho základní syntaxí pro zobrazení všech dat z databázové tabulky je:

```
SELECT * FROM nazev_tabulky;
```

Samozřejmě můžeme zobrazit pouze vybrané sloupce tak, že jejich názvy zapíšeme do příkazu namísto hvězdičky. Příkaz *SELECT* podporuje spoustu klauzulí, sloužících ke zpřesnění či filtrování výsledků. Mezi nejzákladnější patří: *WHERE, GROUP BY, HAVING, ODER BY*.

Příkaz INSERT

Slouží k vložení nového záznamu do tabulky. Syntaxe vypadá následovně:

```
INSERT INTO nazev_tabulky ( seznam_sloupcu ) VALUES ( seznam_hodnot );
```

Obdobně jako příkaz *INSERT* funguje také příkaz *REPLACE*, rozdíl je v tom, že když dojde ke kolizi již existujícího klíče, je původní záznam nahrazen novým.

Příkaz UPDATE

Pomocí příkazu *UPDATE* můžeme měnit obsah již existujících záznamů.

```
UPDATE nazev_tabulky SET jmeno_sloupce = hodnota WHERE podminka;
```

Klauzule *WHERE* není povinná, avšak pokud bychom ji neuvedli, aktualizovaly by se všechny záznamy tabulky.

Příkaz DELETE

Příkaz DELETE slouží k odstranění záznamů z tabulky. Příklad syntaxe je následovný:

```
DELETE FROM nazev_tabulky WHERE podminka;
```

I zde je klauzule *WHERE* nepovinná, ale pokud není uvedena, dojde ke smazání všech řádků tabulky.

7 PHP

PHP nebo li *Hypertext Preprocesor* je skriptovací programovací jazyk určený především pro programování dynamických internetových stránek a aplikací. Nejčastěji se začleňuje přímo do struktury jazyka (X)HTML. To znamená, že PHP umožňuje vkládat vlastní skripty (krátké úseky kódu, ale i celé programy) přímo do hypertextových stránek.

PHP skripty jsou většinou prováděny na straně serveru, k uživateli je přenášen až výsledek jejich činností. Velmi dobře se hodí pro spolupráci s databázemi, zpracování formulářů a náročnější úlohy jako manipulace s grafikou či soubory PDF. [15]

7.1 Historie PHP

Základem PHP byla v roce 1995 jednoduchá sada skriptů Rasmuse Lerdorfa v jazyce Perl sloužící ke sledování návštěvnosti jeho stránek. Časem začal Lerdorf psát mnohem obsáhlejší implementaci v jazyce C naplňující větší požadavky na funkčnost včetně databázové konektivity. Následně vydal první verzi, samozřejmě jako otevřený zdrojový kód, nazvanou PHP/FI (Personal Home Page/Forms Interpreter). V roce 1997 byla distribuována druhá verze (PHP/FI 2.0), kterou již používalo několik tisíc lidí z celého světa. Hlavním vývojářem byl stále Lerdorf.

Od roku 1998 nastal nový věk PHP. Objevila se verze 3 a jednalo se o kompletní přepis verze PHP/FI 2.0, o který se postarali Andi Gutmans a Zeev Suraski. Aby využili rostoucí uživatelské báze, rozhodli se Lerdorf, Gutmans a Suraski uvolnit tento výtvor pod názvem PHP.

Poté začali Gutmans a Suraski s vývojem PHP 4, jejíž první oficiální verzi uvolnili v květnu 2000. Jazyk PHP 4 nabízel mnohem vyšší výkonnost a přinesl nové technologie, jako session, bufferování výstupu a bezpečnější možnosti zpracování uživatelského vstupu.

Zatím poslední verzí je PHP 5, která byla vydána roku 2004. Nové objektově orientované funkce jazyka PHP usnadnily správu především rozsáhlých projektů. Přínosné je i nové propojení s databází MySQL s názvem *mysqli*, které je samozřejmě také objektově orientované, dále jsou tu nové funkce pro úpravu dat ve formátu XML, zavedení *exceptions* apod. [16]

7.2 Princip PHP

Programovací jazyk PHP byl vyvinut k tomu, aby se statické stránky obohatily o dynamické skriptování. Dříve se pro tyto operace používaly programy volané přes rozhraní Common Gateway Interface (CGI), zajišťující spuštění nového procesu na webovém serveru, avšak tento proces byl velmi náročný na čas. Naproti tomu u PHP se funkce načítají jako součást webového serveru, a jsou tedy ke spuštění přivedeny přímo v kontextu procesu webového serveru. To přináší enormní zrychlení, hlavně u frekventovaných stránek.

Pokud je správně nakonfigurovaný webový server, pak každý soubor s příponou *.php* zpracovává interpreter jazyka PHP. Obsah souboru se interpreterem zpracuje a provedou se v něm zadané instrukce. Výsledek se vrátí webovému serveru a ten poté zašle obsah klientovi (většinou internetovému prohlížeči). K nejdůležitějším instrukcím jazyka PHP patří značky pro začátek a konec PHP kódu. Pokud je tedy interpreter v kódu nenajde, pak žádný příkaz nevykoná a vrátí text zdrojového kódu v nezměněné podobě. [3]

7.2.1 Identifikace PHP kódu

Existují v zásadě dva spolehlivé způsoby, jak správně identifikovat PHP kód v souboru HTML.

Prvním způsobem je tzv. standardní zápis tvořený značkami *<?php* a *?>*, označující začátek a konec kódu PHP. Jako příklad si uvedeme kód pro výpis hodnoty proměnné.

```
<?php echo $x; ?>
```

Druhým způsobem je tzv. dlouhý zápis, který je ovšem díky své délce používám sporadicky.

```
<script language="php">
    echo $x;
</script>
```

Samozřejmě existují i jiné způsoby zápisu vkládání PHP kódu do HTML dokumentu, ovšem jejich použití není doporučováno. Mezi ně můžeme zařadit například tzv. krátký zápis tvořený značkami *<?* a *?>*.

8 KNIHOVNA FPDF

FPDF je třída napsaná v jazyce PHP, která slouží k vytváření PDF dokumentů přímo za pomocí jazyka PHP, což znamená, že ke své činnosti nepotřebuje používat knihovnu PDFLib. Jejím tvůrcem je Olivier Plathey, který ji rozšířil pod freeware licencí. To že je knihovna zdarma reprezentuje také první písmeno v jejím názvu F (freeeware) a každý si tak může měnit její funkčnost podle své potřeby.

Mezi hlavní rysy knihovny FDPF patří:

- Volba jednotek, formátu papíru a okraje.
- Záhlaví a zápatí stránky.
- Automaticky nová stránka.
- Automatické zalomení řádku a zarovnání textu.
- Podpora obrázků typu (JPEG, PNG a GIF).
- Bravy
- Odkazy
- Podpora kódování.
- Komprese stránek.

Třída může vytvářet dokumenty v mnoha jazycích a to včetně mimoevropských. Předpokladem ke správnému zobrazení jazyků jako azbuka, řečtina, thajština, ale také čeština je ovšem vytvoření vlastního písma s požadovanou znakovou sadou.

Ačkoliv výkon třídy FPDF je nižší, než u knihovny PDFLib, z důvodu vytváření dokumentů přímo z PHP, je její potenciál velmi obrovský a v její oblíbenost hovoří velký počet rozšíření a nástaveb.

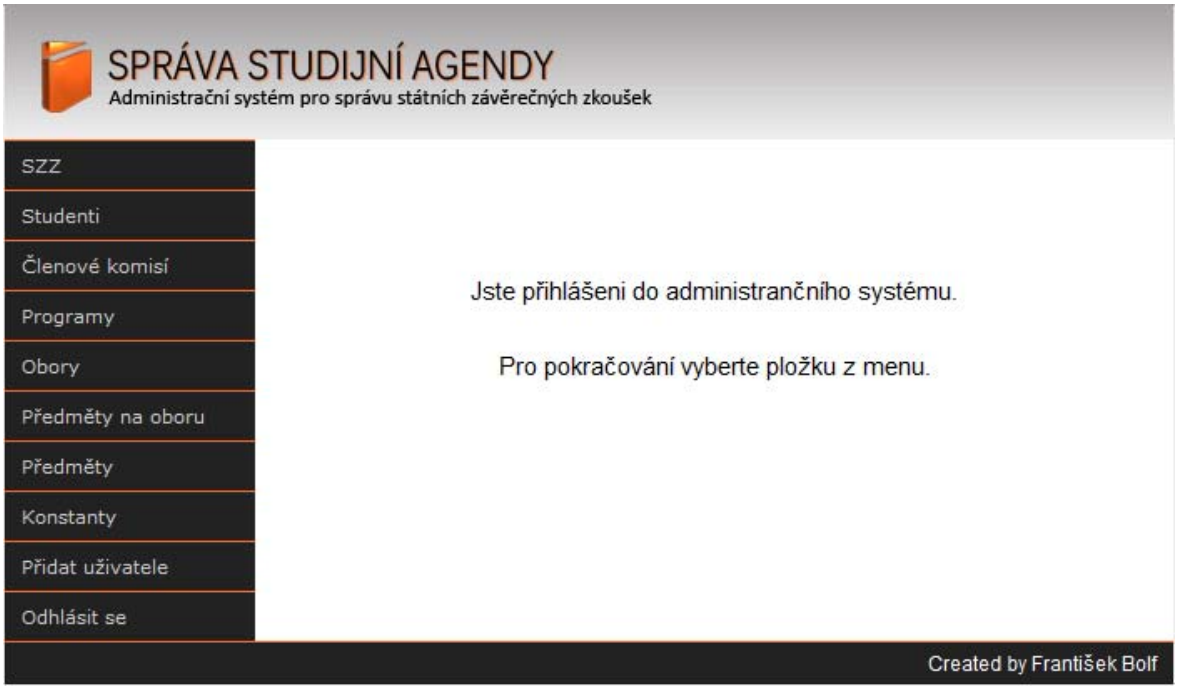
Citováno z [18].

II. PRAKTICKÁ ČÁST

9 ANALÝZA STÁVAJÍCÍHO INFORMAČNÍHO SYSTÉMU

Hlavní částí diplomové práce byla realizace systému sloužícího ke správě studijní agendy. Studijní agenda slouží primárně k vytváření rozpisů pro státní závěrečné zkoušky a tvorbě dalších neméně důležitých funkcionalit souvisejících se samotnými rozpisy a studenty. Systém je zásadně určen pro zaměstnance Univerzity Tomáše Bati ve Zlíně, konkrétně pro Fakultu aplikované informatiky. Jeho primární funkce bude nastíněna níže.

Nejprve bylo nutné provést analýzu stávajícího informačního systému. Důkladně provedená analýza společně s návrhem úprav nám zajistí kvalitní základ, který využijeme při další práci. Je tedy nutné tuto část projektu nezanedbat a přesně určit, jakým směrem se jeho vývoj bude ubírat. Jak jsem již zmínil v úvodu, při realizaci informačního systému jsem vycházel ze své bakalářské práce [7]. Z té byla převzata databázová struktura a aplikační rozhraní vytvořené pomocí programovacího jazyka PHP. Ovšem i tyto části doznaly v rozšíření systému zásadních změn tak, aby splňovaly požadovaná kritéria.



Obr. 3 Hlavní stránka původního systému

Stěžejním prvkem původního systému byla databáze vytvořená v MySQL. Obsahovala celkem 14 tabulek sloužících ke správě dat. Její struktura vycházela z požadavků kladených na předcházející systém realizovaný v programu Microsoft Access. Ovšem pro

stávající rozšířený systém bylo potřeba požadavky upravit, a to převážně v oblasti týkající se množství zpracovávaných dat. Došlo tedy k celkovému nárůstu objemu databáze.

K databázi bylo možno přistupovat pomocí aplikačního rozhraní, které vyžadovalo přihlášení do systému. Systém rozlišoval dva typy uživatelských účtů, a to administrátorský a uživatelský. Ovšem oba přístupy do systému měly téměř identická práva týkající se práce s daty. Administrátor měl možnost registrovat nové uživatele, avšak nemohl jim libovolně omezovat pole působnosti. V rozšířeném systému bude tedy potřeba jasně definovat pole působnosti jednolitých uživatelů a přidělování uživatelských práv dát do pravomocí administrátora. Tím dojde k jednoznačnému určení, jaká data bude moci daný uživatel přidávat, upravovat či odstraňovat.

V neposlední řadě je potřeba se zaměřit také na vzhled aplikace. Přece jenom dobře členěné uživatelské rozhraní dopomáhá k jednouchému a intuitivnímu ovládání aplikace. Na *Obr. 3* je znázorněná hlavní stránka původního systému. Prvním podmětem, který se dostaví po zhlédnutí výše uvedeného obrázku, je poměrně nevyvážené barevné řešení. Chtělo by to celkově více uhladit barvy, aby nebyly tak křiklavé. Celkem nešťastně je řešeno menu, které kromě funkčních záložek sloužících k ovládání systému obsahuje také položku náležící uživatelskému účtu. V tomto případě se jedná o záložku „*Odhlásit se*“, která není oddělena od zbytku menu a poměrně se ztrácí. V nejlepším případě by se tyto funkční odkazy týkající se uživatelského účtu měly nacházet na samostatném odděleném místě.

9.1 Shrnutí analýzy

Z analýzy jasně vyplývá, že pro rozšíření systému bude potřeba zapracovat na téměř všech částech původní aplikace. Je nutné přepracovat databázovou strukturu tak, aby vyhovovala novým nárokům na systém. S tím samozřejmě souvisí úprava zdrojového kódu, který by měl obsahovat objektové části, spolu s použitím nových a propracovanějších metod používaných pro přístup k databázi. Je důležité, aby spojení s databází fungovalo spolehlivě a rychle, neboť systém pracuje převážně s databázovými údaji. Důležitou součástí je také tvorba jednoduchého a intuitivního uživatelského rozhraní, které nebude obsahovat rušivé prvky. Samozřejmostí je přepracování vzhledu aplikace s přesně definovaným layoutem.

10 NÁVRH ÚPRAV A DOPLNĚNÍ STÁVAJÍCÍHO SYSTÉMU

Stávající systém bylo potřeba náležitě upravit a také zabezpečit, neboť jednou z hlavních podmínek bylo jeho umístění na univerzitní web. Systém tedy musí dokázat odolat útokům zvenčí i zevnitř. Bylo nutné doplnit ochranu proti SQL Injection a nebo proti případné záměně URL adresy. Nyní se pokusím nastínit hlavní úpravy, ke kterým došlo.

Prvotní úpravou byla změna vzhledu aplikace takovým směrem, aby se celkový design přiblížil vzhledu univerzitního webu. Hlavním úmyslem bylo vytvořit uživatelské rozhraní s jednoduchým a přívětivým ovládáním. Cílem bylo rozmístit ovládací prvky do tří hlavních částí. Do záhlaví stránky umístit prvky přímo související s uživatelským účtem. Tím bude zajištěna jejich snadná přístupnost a viditelnost. Vlevo poté vytvořit menu, které bude obsahovat záložky sloužící k přímému ovládání systému. Menu musí být jasně čitelné a logicky uspořádané. Poslední částí je hlavní pracovní plocha sloužící k zobrazování a práci s daty umístěná napravo od hlavního menu.

Další úpravy se týkají samotné databáze, která se musela přizpůsobit uživatelským nárokům. Úpravou některých stávajících tabulek a přidáním nových, došlo ke změně struktury databáze. S těmito úpravami samozřejmě úzce souvisejí programové inovace. Přístup k databázi se přepracoval na objektový a došlo ke zpřehlednění programového kódu.

Nyní se dostáváme ke změnám týkajících se funkce systému. Bylo potřeba zavést striktní rozdělení přístupových práv různých uživatelů. Uživatel může mít buď administrátorská, nebo uživatelská práva. Administrátor má na starosti kompletní správu systému, kdežto uživatel má od administrátora přesně vytyčeno pole působnosti. To umožňuje jednotlivým uživatelům přistupovat pouze k určeným datům. S uživatelskými právy přímo souvisí vytváření záloh, kde administrátor má jako jediný možnost vytvářet zálohu celé databáze, a také ji v případě potřeby importovat. Kdežto uživatel provádí zálohu pouze z těch dat, ke kterým má povolen přístup. Důležitou novinkou je zavedení možnosti generování dekretů, sloužících k vytváření tiskových výstupů přímo z aplikace. Tiskových sestav je v systému několik a ještě se k jejich vytváření později vrátíme.

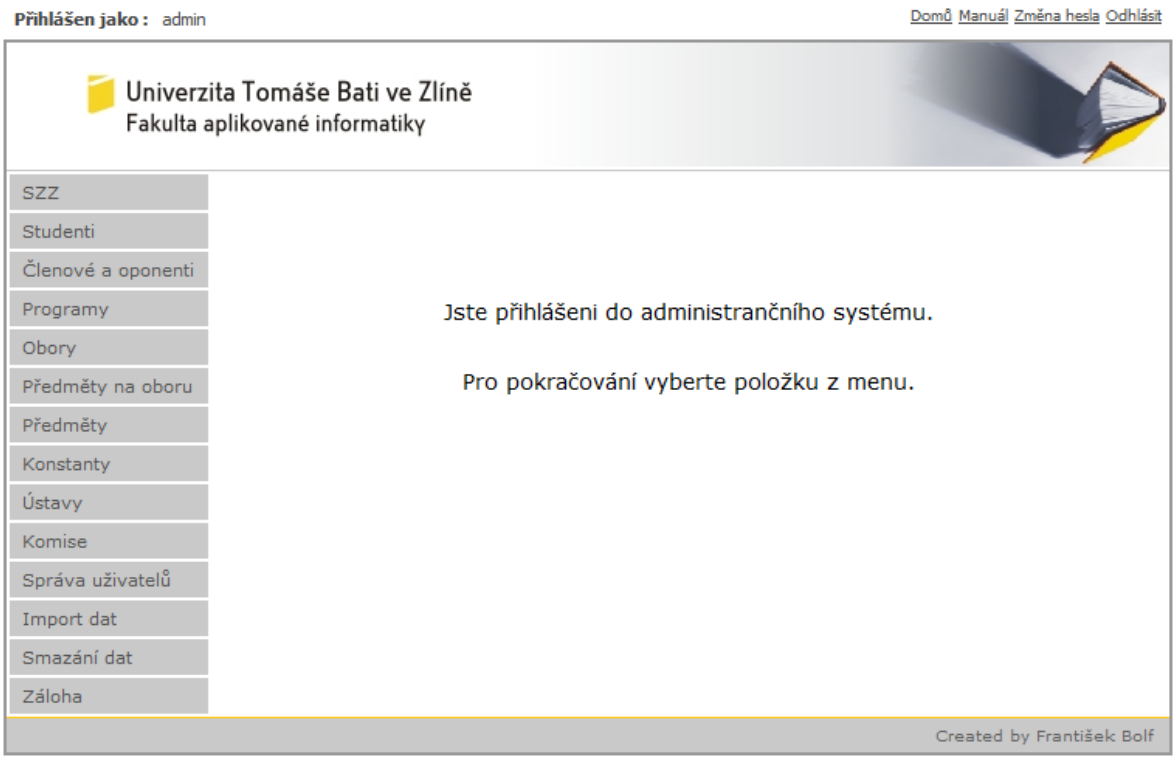
To je stručný výpis nejdůležitějších inovací, ke kterým se samozřejmě později vrátíme a podrobně si popíšeme jejich funkce a možnosti.

11 REALIZACE SYSTÉMU

Zde se pokusím nastínit hlavní části diplomové práce související se samotným postupem realizace systému. Nebudu zde přímo popisovat uživatelské ovládání jednotlivých částí programu, k tomu slouží uživatelské manuály, které jsou taktéž součástí systému.

11.1 Grafická úprava aplikace

Jednou z prvotních změn bylo provedení grafické úpravy vzhledu celého systému. Jelikož se jedná o systém, který bude využíván Univerzitou Tomáše Bati ve Zlíně, konkrétně Fakultou aplikované informatiky, snažil jsem se vytvořit kompletní vzhled tak, aby korespondoval s barevnou kompozicí použitou na univerzitních stránkách *http://web.fai.utb.cz/*. Celkový layout je vytvořen pomocí kaskádových stylů a rozmístěn do dvou souborů *.css. Oba zmíněné soubory se nacházejí ve složce *_css* a jsou načítány v hlavičce každé stránky.



Obr. 4 Hlavní stránka rozšířeného systému – administrátorské rozhraní

Rozvržení jednotlivých prvků na stránce odpovídá jasnému schématu. V záhlaví stránky je uvedeno přihlašovací jméno uživatele, a také jsou zde samostatně umístěny ovládací prvky

uživatelského účtu tak, aby byly odděleny od záložek menu a nerušily celkový vzhled. Menu je ponecháno na levé straně stránky a volný prostor situovaný napravo od něj slouží k zobrazování požadovaných dat. Celkový design podle mě nyní působí jednoduchým, ale uhlazenějším dojmem a jasně definuje funkce jednotlivých položek.

Jakou funkci jednotlivé položky plní, zde popisovat nebudu, podrobný popis je uveden v manuálech a to jak pro uživatelské, tak i administrátorské rozhraní.

11.2 Úprava struktury databáze

Následující důležitá úprava se týká samotné databáze. Jelikož byly na systém kladeny vyšší nároky, došlo k navýšení počtu databázových tabulek z původních 14 na celkových 19. Kromě toho byla většina tabulek upravena, tak aby splňovala nové požadavky. Většina úprav se projevila zvýšením počtu sloupců v jednotlivých tabulkách, někde byla ale potřeba upravit i datové typy, převážně z důvodů možnosti importu dat generovaných systémem STAG.

Kódování jednotlivých tabulek databáze je nastaveno na cp1250. S tím samozřejmě souvisí kódování jednotlivých stránek uživatelského rozhraní, které bylo pro vzájemnou kompatibilitu nastaveno na windows-1250. Proč jsem zvolil tento typ kódování a ne například v dnešní době nejrozšířenější UTF-8 se pokusím vysvětlit. Jednoduše protože nebylo potřeba. Zvolený typ kódování dokáže pracovat se všemi českými i slovenskými znaky a co je hlavní, podporuje ho i knihovna FPDF, kterou používám ke generování tiskových sestav. Například kódování UTF-8 tato knihovna nepodporuje a při výpisu tiskových sestav by docházelo ke špatnému zobrazování databázových dat. Jediným řešením by bylo každý údaj složitě překódovat. Nehledě na tuto nevýhodu i data, která jsou do systému načítána z externích souborů generovaných systémem STAG nemají kódování UTF-8. Je tedy patrné, že ke změně kódování není důvod.

11.2.1 Původní tabulky

Tabulky, jejichž tvar nebylo potřeba měnit, jsou 3. Aby bylo možno pochopit celkovou strukturu databáze, a také spojitost mezi jednotlivými daty nastíním u každé tabulky její primární funkci.

Tabulka „program“

Tabulka slouží k uchovávání dat o jednotlivých studijních programech. Každý záznam je definován sloupci *id_program* a *nazev_programu* a první údaj je zároveň primárním klíčem.

Tab. 1 Tabulka „program“

Sloupec	Typ
id_program	varchar(15)
nazev_programu	varchar(60)

Tabulka „typ_studia“

Pomocí záznamů této tabulky lze přesně definovat typ studia. Používá se k rozlišení bakalářského a magisterského typu studia.

Tab. 2 Tabulka „typ_studia“

Sloupec	Typ
id_typ_studia	tinyint(1) *Auto Increment*
typ_studia	varchar(10)

Tabulka „rozpis_studentu“

Tato tabulka je určena k uchovávání dat souvisejících s rozpisy jednotlivých studentů pro dané státnicové termíny. Který den a ke které komisi má student nastoupit je uvedeno ve sloupcích *id_szz* a *datum*. Sloupce s názvem *poradi* a *cas_od* slouží k přesnému určení studentova nástupu ke státním závěrečným zkouškám.

Tab. 3 Tabulka „rozpis_studentu“

Sloupec	Typ
id_rozpis_st	bigint(10) *Auto Increment*
id_szz	int(5)
os_cislo_student	varchar(30)
cas_od	time
poradi	varchar(3)
datum	date

11.2.2 Upravené tabulky

V této části popíši tabulky, které byly převzaty z původní databáze, ale z důvodu jiných nároků na systém došlo k přepracování jejich struktury. U některých jsou změny pouze kosmetického charakteru, u jiných jsou změny poměrně markantní.

Tabulka „komise“

Každý termín státních závěrečných zkoušek je veden pod svou barvou. Jaké barvy lze k jednotlivým termínům přiřadit udávají hodnoty uložené v této tabulce. Každá barva je kromě svého identifikačního čísla a názvu také reprezentována složkami RGB, které slouží k upřesnění barevného odstínu.

Tab. 4 Tabulka „komise“

Sloupec	Typ
id_komise	int(2) *Auto Increment*
nazev_komise	varchar(20)
slozka_r	int(3)
slozka_g	int(3)
slozka_b	int(3)

Tabulka „konstanty“

Data uvedená v této tabulce jsou používána při tiskových výstupech. Slouží k určení základních údajů souvisejících s univerzitou, fakultou a aktuálním akademickým rokem. Novým údajem je zde sloupeček se jménem děkana. Data této tabulky mohou být v systému upravovány pouze administrátorem.

Tab. 5 Tabulka „konstanty“

Sloupec	Typ
id_konstanty	int(3) *Auto Increment*
ak_rok	varchar(9)
fakulta	varchar(50)
univerzita	varchar(50)
dekan	varchar(100)

Tabulka „predmety“

Tabulka obsahuje všechny předměty, které mají souvislost se státními závěrečnými zkouškami. Každý předmět musí být označen identifikačním údajem, zkratkou a názvem.

Z důvodu rozdílných zkratek předmětů pro studenty kombinovaného studia bylo nutné do tabulky doplnit nový sloupec *zkratka_komb*. Tento údaj je nepovinný a slouží k ukládání odpovídajících zkratek předmětů používaných u kombinovaných studentů.

Tab. 6 Tabulka „predmety“

Sloupec	Typ
id_predmet	varchar(10)
zkratka	varchar(30)
zkratka_komb	varchar(30) *NULL*
nazev_predmetu	varchar(70)

Tabulka „obor“

V této tabulce jsou uloženy data o studijních oborech používaných na Fakultě aplikované informatiky. Každý obor je identifikován svým id označením a názvem. Mimo to obsahuje dva upřesňující záznamy, které specifikují nadřazený program a typ studia. Doplňujícím údajem je sloupec *posudek*, vyjadřující datum, do kdy se má pro daný studijní obor odesílat oponentní posudek.

Tab. 7 Tabulka „obor“

Sloupec	Typ
id_obor	varchar(15)
id_program	varchar(15)
nazev_obor	varchar(70)
typ_studia	varchar(10)
posudek	date *NULL*

Tabulka „obor_predmety“

Strukturu této tabulky jsem zde záměrně neuvedl, z důvodů její rozsáhlosti. V této tabulce lze přiřadit jednotlivým oborům jejich státnicové a dílčí předměty. Každý obor může obsahovat maximálně pět státnicových a šestnáct dílčích předmětů. Státnicové předměty jsou ty, ze kterých se skládají státní závěrečné zkoušky a dílčí předměty jsou vybrané předměty absolvované během studia.

Tabulka „szz“

Patří k jedné z nejdůležitějších tabulek v databázi a slouží k uchovávání dat o termínech státních závěrečných zkoušek. Každý termín je specifikován typem studia, barvou komise,

studijním programem a rozmezím dnů, ve kterých se státnice budou konat. Tabulka je doplněna o údaj *szz_od* používaný při generování tiskových sestav a udávající začátek státních závěrečných zkoušek.

Tab. 8 Tabulka „szz“

Sloupec	Typ
id_szz	int(5) *Auto Increment*
typ_studia	varchar(10)
id_komise	int(2)
id_program	varchar(15)
datum1	date *NULL*
datum2	date *NULL*
datum3	date *NULL*
datum4	date *NULL*
szz_od	time *NULL*

Tabulka „rozpis_komisi“

Ke každému státnicovému termínu je samozřejmě potřeba přiřadit zkušební komisi, a právě tato tabulka komisi specifikuje. Každá komise může obsahovat maximálně jednoho předsedu, dva místopředsedy, jednoho tajemníka a šest členů. Jednotliví zkoušející jsou do tabulky uváděni pod svými identifikačními čísly.

Tab. 9 Tabulka „rozpis_komisi“

Sloupec	Typ
id_rozpis_komisi	int(6) *Auto Increment*
id_szz	int(5)
predseda	int(6)
mistopredseda1	int(6)
mistopredseda2	int(6) *NULL*
tajemnik	int(6)
clen1	int(6)
clen2	int(6) *NULL*
clen3	int(6) *NULL*
clen4	int(6) *NULL*
clen5	int(6) *NULL*
clen6	int(6) *NULL*

Tabulka „studenti“

Jak název napovídá, slouží tato tabulka k ukládání dat o studentech. O každém studentovi jsou vedeny osobní údaje, jako tituly, jméno, příjmení, a samozřejmě údaje související se studiem a studijními prospěchy. V tabulce došlo převážně k úpravám tykajících se datových typů jednolitých sloupců. Jelikož se data do této tabulky ve většině případů načítají ze souboru vygenerovaného systémem STAG, bylo nutné strukturu přepracovat tak, aby docházelo k bezproblémovému importu a následné práci s daty.

Tab. 10 Tabulka „studenti“

Sloupec	Typ
os_cislo_student	varchar(30)
id_program	varchar(15)
id_obor	varchar(15)
titul	varchar(10) *NULL*
jmeno	varchar(50)
prijmeni	varchar(50)
titul_za	varchar(15) *NULL*
str_skola	varchar(80)
na_fai_od	varchar(15)
prumer_1rocnik	varchar(10) *NULL*
prumer_2rocnik	varchar(10) *NULL*
prumer_3rocnik	varchar(10) *NULL*
prumer_4rocnik	varchar(10) *NULL*
prumer_5rocnik	varchar(10) *NULL*
prumer_Bc	varchar(10) *NULL*
prumer_Mgr	varchar(10) *NULL*
nazev_prace	varchar(150)
vedouci_prace	varchar(100)
oponent_prace	varchar(100) *NULL*
znamka_vedouci	varchar(1) *NULL*
znamka_oponent	varchar(1) *NULL*
prumer	varchar(10) *NULL*
forma_studia	varchar(20)
vyznamenani	varchar(3)

Tabulka „student_predmety“

Tato tabulka je v databázi nejrozsáhlejší a slouží k ukládání známek jednotlivých státnicových a dílčích předmětů. Kromě toho je zde možné ukládat pro jednotlivé studenty

známky ze státních závěrečných zkoušek. Tabulka obsahuje také sloupec, sloužící k upřesnění volitelného státnicového předmětu pro daného studenta. Tento údaj se do databáze ukládá až dodatečně, a pouze k těm studentům, kteří mají do státnicových okruhů zařazen volitelný předmět.

Tabulka „zkousejici“

Tabulka obsahuje informace o všech členech komisí a oponentech prací, kteří jsou v systému používáni. O každém zkoušejícím jsou uvedeny osobní záznamy a údaje související s univerzitou, na které působí. Tyto záznamy jsou následně využívány při generování dekretů. Posledním sloupcem tabulky je *typ*, který slouží pro přidělení funkce v systému.

Tab. 11 Tabulka „zkousejici“

Sloupec	Typ
id_zkousejici	int(6) *Auto Increment*
tituly_pred	varchar(40) *NULL*
jmeno	varchar(50)
prijmeni	varchar(50)
tituly_za	varchar(40) *NULL*
pohlavi	varchar(3)
univerzita	varchar(100) *NULL*
fakulta	varchar(80) *NULL*
ulice	varchar(60) *NULL*
cp	varchar(10) *NULL*
mesto	varchar(100) *NULL*
psc	varchar(6) *NULL*
stat	varchar(30) *NULL*
typ	varchar(20)

Tabulka „uzivatele“

Do tabulky se ukládají všichni uživatelé, kteří mohou do systému přistupovat. Kromě osobních údajů jako titul, jméno a příjmení je zde uveden také email, na který jsou zaslány uživateli přihlašovací údaje. Do systému se uživatel přihlašuje pomocí údajů uvedených ve sloupcích *uziv_jmeno* a *heslo*. Heslo se samozřejmě do tabulky ukládá již v kódovaném tvaru. Neméně důležitý je poslední sloupec *prava*, který slouží k rozlišení administrátorského a uživatelského účtu. Je-li tento příznak nastaven na hodnotu 1, jedná

se o administrátorský účet, pokud je nastaven na hodnotu 0, jedná se o běžný uživatelský účet.

Tab. 12 Tabulka „uzivatele“

Sloupec	Typ
id_uzivatel	int(3) *Auto Increment*
titul	varchar(10) *NULL*
jmeno	varchar(50)
prijmeni	varchar(50)
email	varchar(80)
uziv_jmeno	varchar(40)
heslo	varchar(40)
prava	tinyint(4)

11.2.3 Nové tabulky

Nové tabulky, které byly do systému přidány, slouží převážně k zefektivnění práce s daty. Tento krok bylo nutné provést na popud zvýšení systémových požadavků.

Tabulka „predmety_rozdeleni“

Jak jsme si již říkali, každý studijní obor je reprezentován určitým počtem státnicových a dílčích předmětů. Je ovšem nutné určit, ke kterému státnicovému předmětu náleží které dílčí, a právě k tomuto účelu slouží tabulka „predmety_rozdeleni“. Rozdělení předmětů je nutné provést z důvodu správného generování některých tiskových výstupů.

Tab. 13 Tabulka „predmety_rozdeleni“

Sloupec	Typ
id_predmety_rozdeleni	int(6) *Auto Increment*
id_obor	varchar(15)
dil_predmet	varchar(30)
st_predmet	varchar(30)

Tabulka „szz_rozpis_oboru“

Jedním z nových požadavků na systém byla možnost vytvářet termíny státních závěrečných zkoušek pro více studijních oborů najednou. Pro realizaci výše zmíněného požadavku slouží tato tabulka. Sloupec *id_szz* určuje daný státnicový termín a k němu přiřazuje studijní obor v podobě sloupce *id_obor*.

Tab. 14 Tabulka „szz_rozpis_oboru“

Sloupec	Typ
id_szz_rozpis_oboru	int(5) *Auto Increment*
id_szz	int(5)
id_obor	varchar(15)

Tabulka „ustav“

Tabulka slouží k ukládání dat o ústavech nacházejících se na Fakultě aplikované informatiky. O každém ústavu jsou uchovávány data o jeho zkratce, názvu a jménu ředitele. Samozřejmě je každý ústav identifikován unikátním číslem *id_ustav*.

Tab. 15 Tabulka „ustav“

Sloupec	Typ
id_ustav	int(3) *Auto Increment*
zkratka_ustavu	varchar(30)
nazev_ustavu	varchar(100)
reditel	varchar(100)

Tabulka „rozpis_ustavu“

Tato tabulka bezprostředně souvisí s tabulkou „ustav“. Slouží k rozdělení patřičných oborů mezi dané ústavy. Každý studijní obor totiž patří pod určitý ústav, a k jejich rozřazení slouží právě tabulka „rozpis_ustavu“.

Tab. 16 Tabulka „rozpis_ustavu“

Sloupec	Typ
id_rozpis_ustavu	int(3) *Auto Increment*
id_ustav	int(3)
id_obor	varchar(15)

Tabulka „uzivatele_prava“

Každý uživatel má v systému přidělená daná práva, která mu přesně vymezují jeho pole působnosti. Přidělování práv má na starosti administrátor, který tak činí pomocí povolení přístupu k jednotlivým studijním oborům. Každý uživatel má tedy právo přistupovat pouze k těm datům, která souvisejí s povolenými studijními obory. V této tabulce jsou uložena data, která souží k nastavení přístupových práv jednotlivým uživatelům pomocí studijních oborů.

Tab. 17 Tabulka „uzivatele_prava“

Sloupec	Typ
id_uzivatele_prava	int(4) *Auto Increment*
id_uzivatel	int(3)
id_obor	varchar(15)

11.3 Použité třídy

Pro naprogramování systému byl zvolen jazyk PHP. Byla použita nejnovější verze PHP 5, která umožňuje psát kód za pomocí objektově orientovaného programování. Samozřejmě jsem této možnosti využil a v programovém kódu jsou používány dvě třídy. První třída s názvem MyDb slouží pro připojení k databázi a práci s ní, druhá třída s názvem FPDF je využívána ke generování PDF souborů. Každou třídu si nyní podrobněji popíšeme.

11.3.1 Třída MyDb

Tato třída slouží pro připojení k databázi a provádění základních úkonů s jejími daty. Všechny operace jsou prováděny pomocí rozhraní *mysqli*. Zmíněné rozhraní jsem se rozhodl použít z toho důvodu, že pro přístup k databázi MySQL využívá objektově orientovaného programování. Mimo tuto výhodu umožňuje daleko efektivněji používat příkazy SQL, jako například hromadné provádění dotazů spouštěné metodou *mysqli_multi_query*. Pro použití uvedeného rozhraní je nutné, aby bylo na serveru nainstalováno PHP 5 a MySQL 4.1 nebo novější. Obě tyto podmínky univerzitní server splňuje, a tak nebyl problém v jeho použití.

Třída obsahuje metody pro připojení k serveru MySQL, provádění dotazů nebo escapovaní speciálních znaků. Celý zdrojový kód třídy je samostatně uložen v adresáři *classes* pod názvem *myDb.php*.

Konstruktor třídy

Konstruktor je automaticky volán při vytvoření instance třídy MyDb a tudíž je do něj vhodné umístit kód pro připojení k MySQL serveru. Pokud dojde k zavolání konstruktoru, jsou ze souboru *password.php* načteny údaje, zajišťující spojení s databází. Jestliže připojení proběhne v pořádku, dojde k nastavení kódování dat a konstruktor vrátí objekt *mysqli*. V opačném případě je spojení ukončeno a dojde k vypsání chyby.

```
function __construct() {
    require_once('./includes/password.php');
    $this->mysqli = @new mysqli($host, $username, $password,
                                $db_name);

    // je pripojeni vporadku?
    if (mysqli_connect_errno()) {
        $this->tiskChyby("Nepodařilo se navázat připojení! (" .
                         mysqli_connect_error() . ")");
        $this->mysqli = FALSE;
        exit();
    }
    $this->mysqli->query ("SET NAMES 'cp1250'");
}
```

Metoda *queryArray*

Tato metoda je jednou z nejpoužívanějších v celém programovém kódu. Slouží k provedení SQL příkazu *SELECT*, kterým jsou vybírána data z databáze. Metoda přebírá jeden parametr v podobě řetězce s SQL dotazem a vrací v případě úspěchu asociativní pole s daty. Pro spuštění SQL příkazu se uvnitř funkce používá metoda *query*, která v případě úspěchu vrací hodnotu objektu. Pomocí metody *num_rows* je otestován počet získaných řádků a následně jsou jednotlivá data uložena do asociativního pole. Pokud dojde během spuštění metody *query* k chybě, je vrácena hodnota *FALSE* a zároveň vypsána chyba.

```
function queryArray($sql) {
    if ($result = $this->mysqli->query($sql)) {
        if ($result->num_rows) {
            $result_array = array();
            while($row = $result->fetch_array())
                $result_array[] = $row;
            return $result_array;
        }
        else
            return FALSE;
    }
    else {
        $this->tiskChyby($this->mysqli->error);
        return FALSE;
    }
}
```

K výběru dat z databáze ovšem neslouží pouze tato metoda. Třída obsahuje také metody *queryObjectArray* a *queryRow*. Obě fungují na stejném principu jako výše popsaná metoda *queryArray*, rozdíl je pouze ve tvaru vrácených dat. Prvně zmíněná metoda vrací data jako objekt záznam a druhá vrací záznam v podobě jednoduchého pole.

Metoda *querySingleItem*

Je to jednoduchá metoda sloužící ke spuštění SQL dotazu *SELECT*, vracející pouze jednu položku. Funguje obdobně jako předchozí metody, liší se pouze v chování při výskytu chyby nebo vrácení prázdného záznamu. Nevrací totiž hodnotu 0, ale -1. Na to si je potřeba dávat pozor v programové části a testovat úspěšné provedení dotazu na hodnotu -1.

```
function querySingleItem($sql) {
    if ($result = $this->mysqli->query($sql)) {
        if ($row = $result->fetch_array()) {
            $result->close();
            return $row[0];
        }
        else {
            // dotaz nevratil zadna data
            return -1;
        }
    }
    else {
        $this->tiskchyby($this->mysqli->error);
        return -1;
    }
}
```

Metoda *execute*

Metoda je používaná k vykonávání SQL dotazů, které nemají žádnou návratovou hodnotu. Lze mezi ně zařadit například SQL příkazy *INSERT*, *UPDATE*, *DELETE* apod. Metoda *execute* je velmi jednoduchá, přebírá jeden parametr ve tvaru řetězce s SQL dotazem, který vykonává pomocí metody *real_query*. Pokud dojde k úspěšnému provedení příkazu je navrácena hodnota *TRUE*, v opačném případě dojde k výpisu chyby a navrácení hodnoty *FALSE*.

```
function execute($sql) {
    if ($this->mysqli->real_query($sql)) {
        return TRUE;
    }
    else {
        $this->tiskchyby("Nemohu vykonat SQL příkaz! (" .
                         $this->mysqli->error . ")");
        return FALSE;
    }
}
```

Metoda *escape*

Tato metoda neslouží pro připojení k databázi, ale i přesto patří k nejvyužívanějším v celém programu. Používá se k escapovaní nebezpečných znaků jako jsou (“, ‘, \ ) apod.

Metoda nejdříve testuje, zda je na serveru zapnuta funkce *magic_quotes_gpc* sloužící k automatickému escapovaní znaků. Pokud funkce aktivována je, dojde pouze ke zkrácení vstupního řetězce o počáteční a koncové bíle znaky a v původním znění je navrácen. Pokud ovšem funkce aktivována není, mohou se v přijatém řetězci nacházet nebezpečné znaky, které je potřeba eliminovat zavoláním metody *real_escape_string*. Samozřejmě i tento řetězec je ve finále zkrácen o počáteční a koncové bíle znaky. Některé servery mají funkci *magic_quotes_gpc* automaticky aktivovánu a pokud by nedocházelo k jejímu testování, byl by vstupní řetězec escapován také pomocí metody *real_escape_string* a výsledná data by obsahovala nepřesné údaje, což je nepřípustné. Metoda *escape* je ve většině případů volána před vkládáním dat do databáze, neboť ta pocházejí z různých formulářů a jejich obsah může být nedůvěryhodný.

```
function escape($text) {
    return (get_magic_quotes_gpc() ? trim($text) :
            trim($this->mysqli->real_escape_string($text)));
}
```

Samozřejmě obsahuje zmíněná třída i jiné metody, avšak zde jsem se snažil popsat alespoň ty nejvíce používané.

Použití třídy *MyDb*

Jednotlivé metody jsme si již popsali a nyní nám zbývá pouze drobná ukázka použití této třídy v praxi. Samotná implementace je velice jednoduchá, neboť většinu věcí za nás dělá třída sama.

Nejprve je nutné vložit do skriptu soubor s deklarací třídy. Ten se nachází v adresáři *classes* pod názvem *myDb.php*. Poté můžeme vytvořit novou instanci třídy s názvem *$db*.

```
require_once('classes/myDb.php');
$db = new MyDb();
```

Následujícím krokem je vytvoření samotného SQL dotazu. V tomto případě se jedná o jednoduchý dotaz vypisující všechny uživatele, kteří jsou v systému registrováni. Řetězec je následně předán jako parametr do metody *queryArray*, která provede zmíněný dotaz a v případě úspěchu vrátí asociativní pole, které je procházeno a postupně vypisováno.

```
$sql = "SELECT uziv_jmeno FROM uzivatele ORDER BY uziv_jmeno";
if ($result = $db->queryArray($sql)) {
    foreach($result as $row) {
        echo $row["uziv_jmeno"]."<br />";
    }
}
```

Z výše uvedeného zápisu je patrné, že práce s třídou MyDb je velice efektivní, neboť všechny potřebné příkazy jsou obsaženy uvnitř třídy a k provedení daných dotazů stačí pouze volat její členské metody.

11.3.2 Třída FPDF

FPDF je převzatá třída sloužící ke generování PDF souborů. Je kompletně napsaná v jazyce PHP a společně s jejím jednoduchým použitím se jednoznačně vybízí k implementaci v tomto projektu. K tomu abychom třídu mohli použít, je nutné si stáhnout její zdrojový kód a umístit jej do projektu. Třída ovšem neumí pracovat s českými znaky, a tak je před jejím prvním použitím nutné vytvořit vlastní sadu fontů. Podrobný návod k vytvoření fontů je uveden na oficiálních stránkách třídy (http://www.fpdf.org/) v záložce „Tutorials“. Pouze doplním, že k vytvoření vlastní sady fontů je potřeba použít jako základ funkční typ písma, vhodné je například písmo z operačního systému Windows a samozřejmě v průběhu vytváření zvolit správný typ kódování, v mém případě to bylo cp1250. Po těchto krocích nám již nic nebrání v plnohodnotném využívání vlastností třídy.

Použití třídy *MyDb*

Nyní bych se pokus nastínit krátkou ukázku použití třídy v praxi. Kód pochází ze skriptu sloužícího ke generování tiskových sestav pro jednotlivé komise státních závěrečných zkoušek. Záměrně není uveden celý zdrojový kód, neboť slouží tato část pouze k ukázce.

Nejprve je nutné v kódu definovat cestu k fontům, které hodláme používat. Nezbytné je také vložení souboru s třídou FPDF.

```
define('FPDF_FONTPATH','./font/');
require('classes/fpdf.php');
```

Třída FPDF při vytváření každé nové stránky automaticky zobrazuje záhlaví a zápatí, která jsou defaultně prázdná. K zobrazení slouží metody *Header* a *Footer*. Pokud tedy metody vyplníme svými daty, dojde k zobrazování těchto údajů při vykreslení každé nové stránky, což je velice výhodné a samozřejmě toho využijeme. Jednoduše tedy vytvoříme potomka třídy FPDF, do jehož odvozené metody *Header* vložíme logo univerzity. To provedeme pomocí metody *Image*, která kromě načtení obrázku ze souboru umožňuje upřesnit jeho pozici na stránce a také velikost.

```
class PDF extends FPDF {
    //Page header
    function Header()
```

```
    {
        //Logo
        $this->Image('_img/logo_fai.png',10,10,104,24.5);
        $this->Ln(38);
    }
}
```

Následuje vytvoření nové instance třídy PDF. Její konstruktor může přejímat tři parametry definující formát, délkovou jednotku a orientaci vygenerované stránky. Naše stránka je formátu A4 s orientací na výšku a délkovou jednotkou v mm, což jsou defaultní parametry a není tedy vyžadováno zadání žádných parametrů. Následuje přidání všech námi vygenerovaných fontů, které hodláme ve skriptu použít.

```
$pdf=new PDF();

$pdf->AddFont('Times','','times.php');
$pdf->AddFont('Timesi','','timesi.php');
$pdf->AddFont('Timesbd','','timesbd.php');
$pdf->AddFont('Timesbi','','timesbi.php');
```

Nyní nám zbývá pouze přidat stránku a nastavit pomocí metody *SetFont* požadovaný font společně s velikostí písma. Pro výpis textu můžeme použít metodu *Cell*, která přebírá parametry jako délka řádku, výška řádku, výsledný text, zobrazení rámečku, pozici kurzoru po vypsání textu a zarovnání textu. Samozřejmě se při výpisu dat používají různé metody, je ovšem zbytečné je zde uvádět, neboť se jedná pouze o ukázku. Podrobný popis jednotlivých metod je uveden v manuálu na oficiálních stránkách.

```
$pdf->AddPage();

$pdf->SetFont('Times','',18);
$pdf->Cell(190,10,"Komise pro státní závěrečné zkoušky a obhajoby "
                  .$out_stud_program1,0,1,'C');
$pdf->Cell(30);
$pdf->Cell(45,14,"Předseda:",0,0);

$pdf->SetFont('Timesbd','');
$pdf->Cell(80,14,$tiskPredseda,0,1);
```

Pro zobrazení výsledného PDF souboru nám stačí zavolat metodu *Output*.

```
$pdf->Output();
```

11.4 Vybrané části kódu

Z důvodu poměrně velké rozsáhlosti systému není možné popsat podrobně každou část programového kódu. V této části práce se tedy zaměříme na důkladnější popis důležitých a zároveň zajímavých částí kódu.

11.4.1 Import dat

Jedná se o poměrně důležitou funkci, která v původním systému chyběla. Slouží k nahrávání dat o studentech společně s jejich známkami. Data jsou načítána z externích souborů, která obsahují údaje vygenerované systémem STAG. Funkce přináší markantní ulehčení práce všem uživatelům, jelikož není potřeba zadávat údaje o studentech dvakrát. Dříve totiž bylo nutné vlastnoručně vyplnit údaje do STAGu, a také do systému pro správu studijní agendy. Nyní stačí údaje vyplnit pouze jednou do STAGu, a poté vygenerovat soubory, ze kterých se údaje po pár kliknutích do systému automaticky nahrají. Samotný import je rozdělen na tři části, a to na import studentů, update studentů a import známek. Nejdříve je nutné samozřejmě importovat data o studentech.

Podrobný popis jak data importovat je uveden v administrátorském manuálu, my se zde zaměříme na programovou část tohoto importu.

Import studentů

Nyní si popíšeme níže uvedenou část kódu sloužící pro import studentů. Po načtení souboru probíhá několik kontrol, zjišťujících zda importovaný soubor není příliš veliký nebo má správnou koncovku či nedošlo při jeho načítání k nenadálé chybě. Poté se soubor nahrává na server do adresáře *import*, ze kterého je později načítán. Upload souboru je realizován funkcí *move_uploaded_file*. Pokud bychom tento krok neprovedli, nebylo by možné, při zapnutém *Safe Mode* soubor číst. V zápětí se zjišťuje, zda provést smazání původních studentů a jejich známek z databáze. Provádí se tedy testování zaškrtávacího políčka „*odstranit stávající studenty a známky*“. Následně stačí určit cestu ke zdrojovému souboru na serveru, který poslouží pro import.

```
$soubor_nazev = $_FILES['data']['name'];
$soubor_cesta = '/import/';
$max_soubor_velikost = 10485760;
if ($_FILES["data"]["error"] > 0) {
    echo "<p class='warningText'>Error: " . $_FILES["file"]["error"]
           . "</p><br />";
}
```

```
if ($_FILES['data']['size'] > $max_soubor_velikost) die (
    "<p class='warningText'>Soubor je příliš velký!</p>");

$koncovka = end(explode(".",$soubor_nazev));
if ($koncovka == "txt" || $koncovka == "csv") { }
else {
    die("<p class='warningText'>Je možné uploadovat pouze soubory s
         koncovkou txt nebo csv.</p>");
}
if (!is_file($_FILES['data']['tmp_name'])) die ("<p
    class='warningText'>Žádný soubor jste neuploadovali !!!</p>");

if (move_uploaded_file($_FILES["data"]["tmp_name"], "./import/".
    $soubor_nazev))
{
    if (!empty($_POST['delStudenti'])) {
        $smazat_tabulku = $db->execute("TRUNCATE TABLE studenti");
        $smazat_tabulku1 = $db->execute("TRUNCATE TABLE
            student_predmety");
        if (!$smazat_tabulku && !$smazat_tabulku1) {
            die("<p class='warningText'>Přepsání tabulky se
                 nezdařilo, zkontrolujte prosím validitu formátu
                 vstupních dat.</p>");
        }
    }
    $soubor = $_SERVER["DOCUMENT_ROOT"].$soubor_cesta.$soubor_nazev;
```

Nyní se dostáváme k samotnému importu dat. Pracovat je možno pouze se soubory s koncovkou **.csv* nebo **.txt*, a to z toho důvodu, že jednotlivá data musejí být oddělena středníkem a každý nový záznam musí začínat na novém řádku. Pro splnění těchto podmínek jsou tedy nejvhodnější již zmíněné typy souborů. Pro samotný import je použit SQL příkaz *LOAD DATA LOCAL INFILE,*načítající data přímo z definovaného souboru. Obsahuje několik argumentů, které je potřeba ke správné funkci zavést. V tomto případě jsou data ukládána do tabulky *studenti*, jednotlivé záznamy musejí být odděleny středníkem (*FIELDS TERMINATED BY ';'*), a každý nový záznam musí začínat na dalším řádku (*LINES TERMINATED BY '\n'*). Argument *REPLACE* přepíše záznamy se stejným primárním klíčem, toho využíváme při aktualizací údajů. Na závěr je možné pomocí argumentu *SET* nastavit, se kterými daty se mají provádět další operace.

```
if ($db->execute("LOAD DATA LOCAL INFILE '".$soubor."' REPLACE INTO
TABLE studenti FIELDS TERMINATED BY ';' LINES TERMINATED BY '\n'
(os_cislo_student, titul, prijmeni, jmeno, titul_za, str_skola,
na_fai_od, forma_studia, id_program, id_obor, @varPrumer1,
@varPrumer2, @varPrumer3, @varPrumer4, @varPrumer5, @varPrumerBc,
@varPrumerMgr, @varPrumerCelk, vyznamenani, @varPrace, vedouci_prace,
znamka_vedouci, oponent_prace, @varZnamkaOponent) SET prumer_1rocnik
= TRIM(REPLACE(@varPrumer1,'.',',')), prumer_2rocnik =
TRIM(REPLACE(@varPrumer2,'.',',')), prumer_3rocnik =
TRIM(REPLACE(@varPrumer3,'.',',')), prumer_4rocnik =
```

```
TRIM(REPLACE(@varPrumer4,'.',',')), prumer_5rocnik =
TRIM(REPLACE(@varPrumer5,'.',',')), prumer_Bc =
TRIM(REPLACE(@varPrumerBc,'.',',')), prumer_Mgr =
TRIM(REPLACE(@varPrumerMgr,'.',',')), prumer =
TRIM(REPLACE(@varPrumerCelk,'.',',')), nazev_prace =
TRIM(REPLACE(@varPrace,'\nl{}',' ')), znamka_oponent =
TRIM(REPLACE(@varZnamkaOponent,'\r',''));")) { ?>
```

Zároveň s importem studentů probíhá ukládání dat do tabulky *student_predmety*. V té má každý student přiřazeny státnicové a dílčí předměty podle svého studijního oboru. Provádění tohoto kroku je důležité pro pozdější import známek.

Import studentů končí výpisem textu o úspěšnosti či neúspěšnosti uložení dat do databáze.

```
    <p class="addRecord">Data byla IMPORTOVÁNA!</p>
    <p class="addRecordText">Pro návrat klikněte <a href=
            "import_dat.php" class="odkaz">zde</a> nebo vyberte
             položku z menu.</p>
<?php }
}
else { ?>
   <p class="warningText">Chyba: Data se nepodařilo zpracovat!</p>
    <p class="warningText">Pro návrat klikněte <a href="<?php echo
            $_SERVER['HTTP_REFERER']; ?>" class="odkaz">zde</a>.</p>
<?php }
}
```

Update studentů

Update studentů funguje na obdobném principu jako import studentů. Rozdíl je pouze v tom, že při importu studentů dochází k přepsání celého záznamu, kdežto update studentů slouží k přepsání pouze nejčastěji se měnících údajů. Které konkrétní záznamy jsou přepisovány, je uvedeno v administrátorském manuálu.

Jelikož update probíhá téměř identicky jako import, není potřeba jej dopodrobna popisovat. Pro získání dat je používán stejný soubor, jako pro import studentů. Soubor je po načtení procházen a požadovaná data jsou vyseparována a následně použita k vytvoření několika SQL příkazů *UPDATE*. Ty jsou následně zpracovány pomocí metody *multi_query*, která provede aktualizaci.

Import známek

Bezprostředně s importem studentů souvisí také import známek. Soubor s daty je načten obdobně jako v předchozím případě a všechny údaje jsou po řádcích uloženy do pole. Následně je pole procházeno v cyklu a pomocí funkce *explode* dochází k

vzájemnému oddělení jednotlivých údajů a uložení záznamů do pole *$row_array*. Tím máme předpřipravená data ze souboru, pro jejich vkládání do databáze.

Níže uvedená část kódu slouží k vytváření aktualizačního SQL dotazu. Pomocí funkce *array_search* je hledána pozice předmětu, ke kterému má být přiřazena známka. Pokud je předmět nalezen, je mu přiřazen poziční klíč, jehož hodnota je následně využita pro vytvoření SQL příkazu *UPDATE* vkládajícího známku k danému předmětu. Celý tento cyklus se opakuje, dokud nejsou projité všechny předměty a známky načtené ze souboru.

```
    $predmetKlic = array_search($zkratkaPredm, $predmetyArray);
    if (is_numeric($predmetKlic)) {
        $predmetKlic += 1;
        $sql .= "UPDATE student_predmety SET znamka_dp$predmetKlic =
                '$row_array[3]' WHERE os_cislo_student = '$row_array[0]'
                AND dil_predmet$predmetKlic = '$zkratkaPredm';";
    }
```

Tímto nám vznikne jeden velký textový řetězec s názvem *$sql*, který obsahuje všechny aktualizační SQL dotazy. Celý řetězec je následně předán jako parametr metodě *multi_query*, která provede všechny aktualizační dotazy najednou. Pokud vše proběhne v pořádku, je zobrazena informace o úspěšném provedení importu, v opačném případě je zobrazena chyba.

```
    if ($db->odkazNaMysqli()->multi_query($sql)) { ?>
        <p class="addRecord">Data byla IMPORTOVÁNA!</p>
        <p class="addRecordText">Pro návrat klikněte <a href=
                "import_dat.php" class="odkaz">zde</a> nebo vyberte
                 položku z menu.</p>
<?php }
    else { ?>
        <p class="warningText">Chyba: Data se nepodařilo zpracovat!</p>
        <p class="warningText">Pro návrat klikněte <a href="<?php echo
                $_SERVER['HTTP_REFERER']; ?>" class="odkaz">zde</a>.</p>
<?php }
```

11.4.2 Update záznamů

Systém obsahuje poměrně hodně záznamů, které jsou na sobě vzájemně závislé. Jedná se například o identifikační čísla oborů, předmětu apod. Tyto údaje jsou vedeny v několika různých tabulkách a při aktualizaci jedné hodnoty je nutné, aby se změny provedly také ve zbývajících částech databáze. Pro tento účel se přímo nabízí použití triggerů neboli česky spouští. Jedná se o uloženou proceduru, která se spouští v souvislosti s provedením nějakého akčního dotazu na tabulce. Akčním dotazem jsou zde myšleny SQL příkazy

INSERT, *UPDATE* nebo *DELETE*. Triggery jsou podporovány od verze MySQL 5.0.2 a univerzitní server používá MySQL klienta verze 5.0.32. Z toho je patrné, že použití triggerů v systému by bylo možné, ovšem je zde ještě jeden problém. Vytvářet triggery mohou pouze uživatelé se SUPER právy, což my samozřejmě nejsme. Vytvářet triggery s normálními právy je možno až od verze MySQL 5.1.6, což MySQL klient nainstalován na univerzitním serveru nesplňuje a tím pádem je použití triggerů zavrhnuto. Existují avšak i jiné poměrně jednoduché řešení, kterými lze tuto situaci řešit.

Jak již jsem dříve zmínil, rozhraní *mysqli* obsahuje metodu *multi_query*, která slouží k vykonávání hromadných dotazů. Proč tedy nevyužít této možnosti, když stačí pouze vytvořit několik aktualizačních dotazů, které budou splňovat stejnou funkci, jako použití triggeru. Níže jsou uvedeny všechny dotazy, které je potřeba provézt při změně identifikačního čísla oboru.

```
$sql = "UPDATE obor SET id_obor = '$editIdOboru', nazev_obor =
        '$editNazevOboru', id_program = '$editProgram', typ_studia =
        '$editTypStudia' WHERE id_obor = '$oborH';";
$sql .= "UPDATE obor_predmety SET obor = '$editIdOboru' WHERE obor =
        '$oborH';";
$sql .= "UPDATE predmety_rozdeleni SET id_obor = '$editIdOboru' WHERE
        id_obor = '$oborH';";
$sql .= "UPDATE rozpis_ustavu SET id_obor = '$editIdOboru' WHERE
        id_obor = '$oborH';";
$sql .= "UPDATE studenti SET id_obor = '$editIdOboru' WHERE id_obor =
        '$oborH';";
$sql .= "UPDATE student_predmety SET id_obor = '$editIdOboru' WHERE
        id_obor = '$oborH';";
$sql .= "UPDATE szz_rozpis_oboru SET id_obor = '$editIdOboru' WHERE
        id_obor = '$oborH';";
$sql .= "UPDATE uzivatele_prava SET id_obor = '$editIdOboru' WHERE
        id_obor = '$oborH'";
```

Poté stačí pouze zavolat metodu *multi_query* a předat ji jako parametr textový řetězec obsahující výše uvedené SQL dotazy.

```
$db->odkazNaMysqli()->multi_query($sql);
```

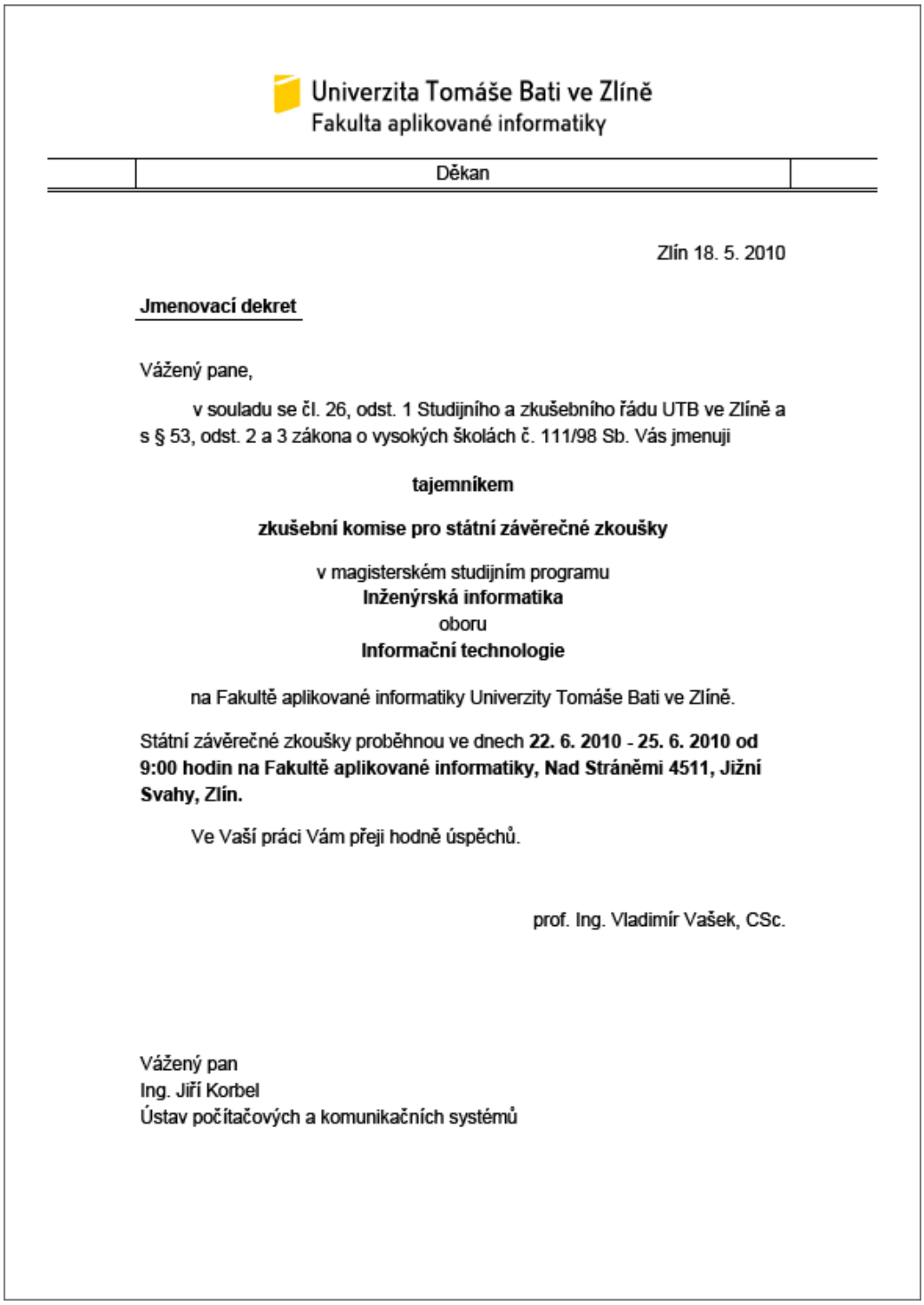
11.4.3 Tiskové výstupy

Jednou z nejdůležitějších funkcí, které byly do rozšířeného systému implementovány, je možnost generování výstupních tiskových sestav. V podstatě se dá říct, že veškeré rozpisy studentů, komisí apod. jsou vytvářeny pro to, aby je bylo možno následně vygenerovat do textové podoby. Generování všech tiskových výstupů je prováděno pomocí třídy FPDF, jejíž funkce byla popsána výše.

Systém umožňuje vytvářet následující typy tiskových výstupů:

- Jmenovací dekret do funkce oponenta práce – slouží k vytvoření tiskového výstupu, který je následně zasílán danému oponentovi bakalářské či diplomové práce s žádostí o oponentní posudek.
- Jmenovací dekret do zvolené funkce zkušební komise – slouží ke generování dekretů pro předsedy, místopředsedy, tajemníky a členy zkušebních komisí státních závěrečných zkoušek. Tyto tiskové výstupy jsou zasílány jednotlivým zkoušejícím a udávají funkci, kterou bude daný zkoušející v komisi zastávat.
- Tisk komise – slouží k vytvoření tiskového výstupu, ve kterém je rozepsána kompletní komise pro daný státnicový termín.
- Rozpisy studentů – v těchto tiskových výstupech jsou pro dané státnicové termíny rozepsáni jednotliví studenti. Každý student má přesně udáno, do které státnicové komise patří, který den a v jakou hodinu nastupuje ke státním závěrečným zkouškám.
- Tisk pavouků – slouží ke generování tiskových výstupů, ve kterých jsou o každém studentovi vyobrazeny podrobnější informace týkající se jeho studia a studijního prospěchu. Tento výstup úzce souvisí s rozpisy studentů.
- Tisk štítků – do tohoto výstupu jsou vypsány zvolené kontaktní adresy jednotlivých oponentů. Tyto štítky jsou následně používány k rozesílání jmenovacích dekretů pro oponenty.

Jak vytvářet jednotlivé tiskové výstupy je podrobně popsáno v uživatelských manuálech.



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Děkan

Zlín 18. 5. 2010

Jmenovací dekret

Vážený pane,

v souladu se čl. 26, odst. 1 Studijního a zkušebního řádu UTB ve Zlíně a s § 53, odst. 2 a 3 zákona o vysokých školách č. 111/98 Sb. Vás jmenuji

tajemníkem

zkušební komise pro státní závěrečné zkoušky

v magisterském studijním programu
Inženýrská informatika
oboru
Informační technologie

na Fakultě aplikované informatiky Univerzity Tomáše Bati ve Zlíně.

Státní závěrečné zkoušky proběhnou ve dnech **22. 6. 2010 - 25. 6. 2010 od 9:00 hodin na Fakultě aplikované informatiky, Nad Stráněmi 4511, Jižní Svahy, Zlín.**

Ve Vaší práci Vám přeji hodně úspěchů.

prof. Ing. Vladimír Vašek, CSc.

Vážený pan
Ing. Jiří Korbel
Ústav počítačových a komunikačních systémů

Obr. 5 Ukázka jmenovacího dekretu do funkce tajemníka zkušební komise

11.4.4 Odeslání emailu

Jednou z věcí, která se automaticky provede při registraci nového či úpravě stávající uživatele je odeslání přihlašovacích údajů na jeho email. Správu uživatelů společně

s udělováním uživatelských práv má na starosti administrátor systému. Je tedy pouze na něm, kdo do systému bude mít povolen přístup a s jakým omezením.

Poté co administrátor vyplní údaje pro registraci nového uživatele a následně odešle formulář, je automaticky generován email ve tvaru uvedeném níže. K tomu aby mohl být email úspěšně odeslán je vyžadována definice několika povinných argumentů. Prvním argumentem je emailová adresa příjemce reprezentována proměnou *$to*, což je zároveň emailová adresa daného uživatele. Dále je nutné vyplnit předmět emailu, který je uložen do proměnné *$subject* a následně kódován, aby bylo zaručeno jeho správné zobrazování v hlavičce emailu. Třetím argumentem je samotný text emailu. Text je uložen do proměnné *$message* a lze jej formátovat pomocí HTML značek. Tyto tři argumenty jsou k odeslání emailu vyžadovány.

```
// email
$to = $addEmail;
$subject = 'Registrační údaje pro správu szz';
$subject1 = "=?windows-1250?B?".base64_encode($subject)."?=";
$message = '
    <html>
    <head>
        <title>Registrační údaje pro správu szz</title>
    </head>
    <body>
        <p>Vítejte v aplikaci pro správu státních závěrečných
           zkoušek.</p>
        <p>Do aplikace se můžete přihlásit zde: <a href="
           '.$serverName.'">'.$serverName.'</a></p>
        <p>Prosím schovejte si tento e-mail. Vaše přihlašovací údaje
           do aplikace jsou:</p>
        <p>-----------------------------------------------------</p>
        <p></p>
        Uživatelské jméno: <b>'.$addUzivJmeno.'</b><br />
        Heslo: <b>'.$addPasswd.'</b><br />
        <br />
        <br />
        <p>Údaje o Vašem účtu jsou následující:</p>
        <p>-----------------------------------------------------</p>
        <p></p>
        Titul: '.$addTitul.'<br />
        Jméno: '.$addJmeno.'<br />
        Příjmení: '.$addPrijmeni.'<br />
        <p></p>
        <p>-----------------------------------------------------</p>
        <p>Máte povolen přístup k těmto studijním oborům:</p>
        <p></p>
        '.$prava.'
    </body>
    </html>
';
```

Čtvrtým sice nepovinným, ale neméně důležitým argumentem jsou hlavičky emailu, jejichž obsah je uložen do proměnné *$headers*. V hlavičkách jsou upřesněny údaje jako typ zprávy, kódování, ale také emailová adresa odesílatele.

```
// To send HTML mail, the Content-type header must be set
$headers  = 'MIME-Version: 1.0' . "\r\n";
$headers .= 'Content-type: text/html; charset=windows-1250' . "\r\n";

// Additional headers
$headers .= 'From: admin <'.$serverAdmin.'>' . "\r\n";
```

Pro odeslání emailu slouží funkce *mail*, přebírající v tomto případě čtyři výše zmíněné parametry. Zda dojde k úspěšnému odeslání emailu je následně uvedeno pomocí výstupního textu.

```
if( @mail($to, $subject1, $message, $headers) ) { ?>
    <p class="addRecordText">E-mail s přihlašovacími údaji byl
              úspěšně odeslán.</p>
<?php }
else { ?>
    <p class="warningText">Chyba: E-mail s přihlašovacími údaji se
              nepodařilo odeslat.</p>
<?php }
```

Obdobnou strukturu má také email upozorňující na aktualizaci záznamu. Liší se od výše zmíněného postupu pouze o pár drobností v předmětu a textu zprávy.

11.5 Uživatelské manuály

Jedním z požadavků diplomové práce bylo vytvoření uživatelských manuálů, sloužících k popisu jednotlivých částí systému. Jde primárně o popis ovládání systému a informace o funkcích jednotlivých záložek menu.

Jsou k dispozici dvě verze uživatelských manuálů, a to administrátorský a uživatelský. Manuály jsou přístupné po přihlášení do systému v záhlaví stránky. Jaká verze manuálu pro Vás bude přístupná, záleží na tom, pod jakým uživatelským účtem se přihlásíte.

12 IMPLEMENTACE SYSTÉMU

Jedním z požadavků diplomové práce byla implementace systému na univerzitní web. Tento krok bylo potřeba provézt co nejdříve, aby se systém náležitě prověřil v ostrém provozu. Každou část programu jsem samozřejmě důkladně testoval na různé krizové situace, avšak až samotní uživatelé dokážou zhodnotit jeho funkčnost.

Pro zajištění domény a dalších částí souvisejících s implementací systému na univerzitní web bylo potřeba navštívit oddělení informačních systému Univerzity Tomáše Bati. Zde byla pro systém přidělena doména, přístup k MySQL databázi a samozřejmě FTP přístup pro web. Vyřízení těchto záležitostí bylo takřka okamžité a nyní zbývalo umístit systém na web. Správa databáze probíhá pomocí webového rozhraní phpMyAdmin, a tudíž nebyl žádný problém s nahráním databáze společně s daty. Stačilo provést import souboru s kompletní strukturou databáze implementovaného systému. Při nahrávání obsahu webu také nedošlo k žádnému problému. Stačilo pouze správně nastavit FTP klient a pomocí něj provést nahrání všech adresářů a souboru přes FTP protokol. Jedinou úpravou, ke které po nahrání došlo, byla změna práv u adresářů *import*, *zaloha_admin* a *zaloha_user*. Do těchto adresářů jsou totiž zapisována data při importu či záloze údajů, a bylo nutné nastavit práva i pro zápis. Po provedení těchto operací byl systém kompletně nahrát na univerzitní web a nachází se pod doménou *ssa.fai.utb.cz*.

Kompletní nahrání systému bylo provedeno 13. 4. 2010 a od té doby probíhá intenzivní testování a ladění celého systému. Na testování se samozřejmě nepodílím pouze já jako vývojář, důležitou roli hrají také budoucí uživatelé a administrátor.

13 ZABEZPEČENÍ A SPRÁVA DATABÁZE

Jelikož systém pracuje s choulostivými informacemi, je nutné, aby byl vstup do aplikace chráněn. Bez přihlášení tedy není možné se systémem vůbec pracovat. Správu uživatelů, kteří mají do systému přístup, má na starosti administrátor. Ten má také možnost vytyčovat jednotlivým uživatelům pole působnosti, a to pomocí udělování práv pro přístup k jednotlivým studijním oborům. Tímto je zajištěno, že jsou danému uživateli zobrazována pouze ta data, ke kterým má přístup a nezasahuje do dat druhých uživatelů.

Podrobný popis jak vytvářet a spravovat uživatelské účty je vysvětleno v administrátorském manuálu.

13.1 Záloha databáze

Zálohování dat z databáze je rozděleno do dvou částí. Pokud se do systému přihlásíme pod administrátorským účtem, máme možnost provádět zálohu celé databáze. Avšak pokud se do systému přihlásíme pod klasickým uživatelským účtem, máme možnost provádět zálohu pouze svých dat. To znamená pouze dat, ke kterým máme povolen přístup.

Pokud dojde k problému s celou databází, je možné importovat zálohu, kterou provedl administrátor, neboť ta obsahuje celou strukturu databáze i s daty. Jestliže dojde k problému s daty pouze jednoho uživatele, má tento uživatel možnost importovat pouze svá zálohovaná data. Při importu této zálohy nedochází k přehrání dat jiných uživatelů.

13.1.1 Export a import celé databáze

Jak jsem se již zmínil, exportovat a importovat celou databázi má možnost pouze administrátor. Postup provádění záloh a jejich importů je uveden v manuálu a tedy se jimi nebudeme zabývat. My si popíšeme programové části, které k provádění těchto úkonů slouží.

Export databáze

Pro export databáze slouží funkce *zalohuj*, nacházející se v adresáři *functions* pod názvem *backup.php*. Funkce přebírá jako parametry název aktuální databáze, název souboru, pod kterým se vyexportovaná data uloží na server a instanci třídy *MyDb*. Níže si uvedeme jak záloha databáze a dat probíhá.

Záloha je tvořena pomocí SQL příkazů, s nimiž lze obnovit celou databázi společně s daty. Jednotlivé příkazy jsou spojovány do jednoho velkého řetězce s názvem *$text*. Každý řádek je tedy reprezentován jedním SQL příkazem. Níže uvedená část kódu slouží k zálohování struktury jednotlivých tabulek.

```
if ($result = $db->queryArray("SHOW TABLE STATUS FROM `$database`")){
    foreach($result as $data) {
    $PRI = $UNI = $MUL = "";

    $text .= "\n\n--\n-- Odstraneni tabulky $data[0]\n--\n\n";
    $text .= "DROP TABLE IF EXISTS `$data[0]`;\n";

    $text .= "\n--\n-- Struktura tabulky $data[0]\n--\n\n";
    $text .= "CREATE TABLE `$data[0]`(\n";

    if ($result = $db->queryArray("SHOW COLUMNS FROM $data[0]")) {
        $e = true;
        foreach($result as $dataa) {
        if($e){
            $e = false;
        }
        else
            $text .= ",\n";
        $null = ($dataa[2] == "NO")? "NOT NULL":"NULL";
        $default = !empty($dataa[4])? " DEFAULT '$dataa[4]'":false;

        if($default == " DEFAULT 'CURRENT_TIMESTAMP'")
            $default = " DEFAULT CURRENT_TIMESTAMP";
        if($dataa[3] == "PRI") $PRI[] = $dataa[0];
        if($dataa[3] == "UNI") $UNI[] = $dataa[0];
        if($dataa[3] == "MUL") $MUL[] = $dataa[0];
        $extra = !empty($dataa[5])? " ".$dataa[5]:false;
        $text .= "`$dataa[0]` $dataa[1] $null$default$extra";
        }
    }

    $primary = keys("PRIMARY KEY",$PRI);
    $unique = keys("UNIQUE KEY",$UNI);
    $mul = keys("INDEX",$MUL);

    $text .= "$primary$unique$mul\n) ENGINE=$data[Engine]
              COLLATE=$data[Collation];\n\n";
    unset($PRI,$UNI,$MUL);
```

Následující část kódu slouží pro zálohování dat existujících tabulek. Data jsou jednoduše vypisována z tabulek a ukládána do textového řetězce.

```
    #data
    $text .= "--\n-- Data tabulky $data[0]\n--\n\n";
    if ($result = $db->queryArray("SELECT * FROM $data[0]")) {
        foreach($result as $fetch) {
        $values = "";
        $pocet_sloupcu = count($fetch)/2;
```

```
            $i = 0;
            while($i < $pocet_sloupcu){
                if($i < $pocet_sloupcu-1)
                    $carka = ",";
                else
                    $carka = "";
                $values .= "'".mysql_escape_string($fetch[$i])."'$carka";
                $i++;
            }
            $text .= "\nINSERT INTO `$data[0]` VALUES($values);";
            unset($values);
            }
        }
        }
    }
```

Na závěr je potřeba celou strukturu databáze společně s daty uložit do souboru. Jednoduše se tedy provede uložení celého textového řetězce s názvem *$text* do souboru, jehož název přejímala funkce *zalohuj* jako parametr.

```
if(!empty($soubor)){
    $fp = @fopen("zaloha_admin/".$soubor,"w+");
    $fw = @fwrite($fp,$text);
    @fclose($fp);
}
```

Import databáze

Import databáze probíhá velice jednoduše. K jeho provedení je nutné vložit soubor, který byl vytvořen v rámci exportu databáze. Ten je po odeslání formuláře načten a postupně zpracováván po řádcích, na které je volána metoda *execute* sloužící pro provádění SQL dotazů. Po provedení všech dotazů načtených ze souboru je import kompletně dokončen.

13.1.2 Export a import uživatelských částí databáze

Exportovat a importovat data svých částí databáze má možnost každý přihlášený uživatel. Není potřeba zde uvádět popis kódu, který výše zmíněné operace provádí, neboť princip je obdobný jako u exportu a importu celé databáze. Rozdíl je pouze v tom, že jednotliví uživatelé nevytvářejí kompletní zálohy databáze, ale mají přesně určeno, která data budou zálohována. Import těchto dat je obdobný, jako u předchozí kapitoly.

ZÁVĚR

Cílem diplomové práce bylo rozšířit, doplnit a odzkoušet systém pro správu studijní agendy, který byl vytvořen v rámci bakalářské práce. Rozšířená aplikace má nahradit „pomocný systém“ vytvořený v programu Microsoft Access, který byl používán dodnes, avšak postrádal potřebnou funkčnost a flexibilitu.

Po důkladné analýze stávajícího systému byly navrženy úpravy, které se týkaly databázové struktury, uživatelského rozhraní a samozřejmě programového kódu. Uživatelské rozhraní je vytvořeno pomocí technologií XHTML a CSS a celkový design stránek je přepracován tak, aby korespondoval se vzhledem univerzitních stránek. Oproti původní verzi se také změnilo rozmístění jednotlivých ovládacích prvků. Z důvodu většího množství zpracovávaných dat došlo k rozšíření MySQL databáze. Ta doznala změn v podobě vytvoření nových a úpravy stávajících databázových tabulek. Programový kód napsaný pomocí PHP a JavaScriptu doznal největších změn. Kromě vytvoření nových modulů došlo k úpravě či rozšíření těch stávajících.

Systém obsahuje několik nových funkcí, které zajišťují veškerou činnost související se správou studijní agendy. Do aplikace je možno přistupovat pod administrátorským či uživatelským účtem a každému uživateli je umožněna práce s daty, ke kterým má povolen přístup. Svá data může libovolně spravovat, vytvářet pomocí nich rozpisy pro státní závěrečné zkoušky a samozřejmě generovat několik druhů tiskových výstupů. Jednotliví uživatelé mohou svá data zálohovat a v případě potřeby je zpět obnovovat. Správu celého systému má na starosti administrátor, který přiděluje uživatelská práva, může provádět export a import celé databáze, a také má na starosti načítání dat z externích souborů, které obsahují údaje generované systémem STAG. Jde o základní údaje o studentovi a jeho studiu, které student podává k Přihlášce ke Státní závěrečné zkoušce. Získané údaje se používají pouze pro účely přípravy a zpracování Státních závěrečných zkoušek.

Aby mohl být systém plně využíván, bylo potřeba jej umístit na univerzitní web. K tomuto účelu byla vytvořena doména *ssa.fai.utb.cz*, na které je systém od 13. 4. 2010 přístupný. Od této chvíle je systém v „ostrém provozu“ a samozřejmě důkladně testován uživateli.

Na závěr lze konstatovat, že všechny body zadání byly splněny a pro použitelnost vytvořené aplikace hovoří fakt, že jsou pomocí ní letos vytvářeny všechny rozpisy a dekrety související se správou Státních závěrečných zkoušek.

ZÁVĚR V ANGLIČTINĚ

The aim of this thesis was to extend, complete and test study agenda administration system, which has been previously created in the bachelor thesis. This extended application should replace previous system created in Microsoft Access, which has been used till today, however lacking needed functionality and flexibility.

After thorough analysis of existing system, rework of database structure, user interface and program code were proposed. User interface is created using XHTML and CSS technology and web site design is redesigned to correspond with the university web site layout. Compared to the previous version, the layout of controls was also changed. Because of large scale of processed data, whole MySQL database was expanded. That included creating new and alternating existing database tables. Program code, written in PHP and JavaScript, was almost completely changed. Besides adding new modules, existing ones were altered or expanded.

System brings several new functions, which provide all activities related to study agenda administration. Application can be accessed under the admin or user account, allowing users to work with authorized data. Each user can freely manage their data, allowing them to create schedules for the final degree exams or generate several kinds of print outputs. Every individual user can backup his data and restore them back. System is managed by the administrator, who assigns user rights. He can also import and export the entire database and is responsible for retrieving data from external files, generated by the STAG system. These include basic information about the student and his study progress, that student attaches to the application form to the final degree exam. Obtained data are used only for purpose of processing and preparing the final degree exams.

In order to fully utilize presented system, it was required to upload it to university web. For this purpose, the domain *ssa.fai.utb.cz* was created and is accessible since 13. 4. 2010. From this moment on, present system is in “full operation” state and is being attested by final users.

All initial tasks were accomplished and for the fact of applicability of presented system, all the schedules and decrees of this year’s final degree exams were created using this system.

SEZNAM POUŽITÉ LITERATURY

[1] PROKOPOVÁ, Zdenka. *Databázové systémy MySQL+PHP*. FAI UTB Zlín, 2006. 126 s. Vysokoškolská skripta. ISBN 80-7318-486-9.

[2] LACKO, Luboslav. *Oracle : Správa, programování a použití databázového systému*. Brno : Computer Press, 2007. 576 s. ISBN 978-80-251-1490-2.

[3] KOFLER, Michael; ÖGGL, Bernd. *PHP 5 a MySQL 5 : Průvodce webového programátora*. Brno : Computer Press, 2007. 608 s. ISBN 978-80-251-1813-9.

[4] CONOLLY, Thomas; BEGG, Carolyn; HOLOWCZAK, Richard. *Mistrovství – Databáze : Profesionální průvodce tvorbou efektivních databází*. Brno : Computer Press, 2009. 584 s. ISBN 978-80-251-2328-7.

[5] POWELL, Thomas A. *Web design : Kompletní průvodce*. Brno : Computer Press, 2004. 844 s. ISBN 80-722-6949-6.

[6] MUSCIANO, Chuck; KENNEDY, Bill. *HTML a XHTML : Kompletní průvodce*. Praha : Computer Press, 2000. 632 s. ISBN 80-7226-407-9.

[7] BOLF, František. *Správa studijní agendy*. Zlín, 2008. 50 s. Bakalářská práce. UTB Zlín, Fakulta aplikované informatiky.

[8] VACH, David. *ITexpert.cz* [online]. 2006 [cit. 2010-03-08]. Třívrstvá architektura. Dostupné z WWW: <http://www.itexpert.cz/trivrstva-architektura/>.

[9] *Wikipedie* [online]. [cit. 2010-03-08]. HyperText Markup Language. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Html>.

[10] *W3C* [online]. [cit. 2010-03-08]. The global structure of an HTML document. Dostupné z WWW: <http://www.w3.org/TR/html401/struct/global.html#h-7.1>.

[11] *Wikipedie* [online]. [cit. 2010-03-08]. Extensible HyperText Markup Language. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Extensible_HyperText_Markup_Language>.

[12] SCHAFER, Steven M. *HTML, XHTML a CSS : Bible pro tvorbu WWW stránek*. Praha : Grada, 2009. 648 s. ISBN 978-80-247-2850-6.

[13] WELLING, L.; THOMSON, L. *PHP a MySQL : rozvoj webových aplikací.* 2. vydání. Praha : Soft Press, 2004. ISBN 80-86497-60-7.

[14] WELLING, Luke; THOMSON, Laura. *MySQL : Průvodce základy databázového systému*. Brno : Computer Press, 2005. 256 s. ISBN 80-251-0671-3.

[15] *Wikipedie* [online]. [cit. 2010-03-27]. PHP. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Php>.

[16] ROSEBROCK, Eric; FILSON, Eric. *Linux, Apache, MySQL a PHP : Instalace a konfigurace prostředí pro pokročilé webové aplikace*. Praha : Grada, 2005. 344 s. ISBN 80-247-1260-1.

[17] CYROŇ, Miroslav. *CSS - kaskádové styly : praktický manuál*. Praha : Grada, 2005. 340 s. ISBN 80-247-1420-5.

[18] *FPDF* [online]. [cit. 2010-03-28]. Dostupné z WWW: <http://www.fpdf.org/>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

MySQL	Databázový systém.
PHP	Skriptovací jazyk využívající se k tvorbě WWW stránek.
JavaScript	Objektově orientovaný skriptovací jazyk.
HTML	Hyper Text Markup Language – značkovací jazyk pro hypertext.
XHTML	Extensible Hypertext Markup Language – značkovací jazyk pro tvorbu WWW stránek.
CSS	Cascading Style Sheets – kaskádové styly.
FPDF	Knihovna sloužící ke tvorbě PDF souborů.
ASP	Active Server Pages – skriptovací platforma společnosti Microsoft.
HTTP	Hypertext Transfer Protocol.
DTD	Document Type Definition – definice typu dokumentu.
XML	Extensible Markup Language – obecný značkovací jazyk.
W3C	World Wide Web Consortium.
SQL	Structured Query Language - dotazovací databázový jazyk.
URL	Uniform Resource Locator.
FTP	File Transfer Protocol.
PDF	Portable Document Format – přenosný formát dokumentů.
phpMyAdmin	Nástroj umožňují správu databáze MySQL.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH