Internetová aplikace pro vypisování bakalářských a diplomových prací

Internet application for publishing bachelor and diploma theses

Bc. Aleš Tichý

Diplomová práce
2007

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2006/2007

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Aleš TICHÝ**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Internetová aplikace pro vypisování bakalářských a diplomových prací**

Zásady pro vypracování:

1) Vypracujte literární rešerši na téma přihlašovací a rezervační systémy na Internetu.
2) Navrhněte strukturu aplikace řešící vypisování témat BP a DP učiteli a následně zapisování studentů na jednotlivá vypsaná témata.
3) Vytvořte program řešící dílčí úkoly (přístupová práva, vypisování témat, kontroly podmínek zápisu, statistiky, tisk výstupních sestav apod.)
4) Umístěte celý systém na univerzitní web.
5) Zpracujte manuál pro uživatele systému.

Rozsah práce:

Rozsah příloh:

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

Seznam odborné literatury:

1) Jakub Mach: PHP pro úplné začátečníky, Computer Press, Praha 2002.
2) Jiří Kosek: PHP – tvorba interaktivních internetových aplikací, Grada Publishing 1999.
3) J. Castagnetto, H. Rawat, S. Schumann, Ch. Scollo, D. Veliath: PHP – programujeme profesionálně, Computer Press, Praha 2002.
4) Larry Ullman: PHP a MySQL – Názorný průvodce tvorbou dynamických WWW stránek, Computer Press, červen 2004.
5) Rastislav Škultéty: JavaScript – Kapesní přehled, Computer Press, leden 2006.
6) Marek Prokop: CSS kaskádové styly pro webdesignéry (2. vydání), Computer Press, únor 2005.
7) George Schlossnagle: Pokročilé programování v PHP 5, Zoner Press, září 2004.
8) C. Darie, B. Brinzarea, F. Chereche-Toa, M. Bucica: AJAX a PHP -- tvoříme interaktivní webové aplikace profesionálně, Zoner Press 2006.

Vedoucí diplomové práce: **Ing. Tomáš Sysala, Ph.D.**
Ústav automatizace a řídicí techniky

Datum zadání diplomové práce: **13. února 2007**

Termín odevzdání diplomové práce: **28. května 2007**

Ve Zlíně dne 13. února 2007

prof. Ing. Vladimír Vašek, CSc.
děkan

L.S.

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Cílem této diplomové práce bylo vytvoření webové aplikace pro správu bakalářských a diplomových prací. V teoretické části této práce se čtenář seznámí s přihlašovacími a rezervačními systémy a s technologiemi, které byly pro vytvoření aplikace použity. Praktická část je zaměřena přímo na vytvořený systém a mimo struktury databázových tabulek popisuje také vytvořené třídy a některé zajímavé skripty, které se v aplikaci nacházejí.

Klíčová slova: Internetová aplikace, webové technologie, přihlašovací systém, rezervační systém, skript.

ABSTRACT

The objective of this diploma thesis was creating a web application for administration of bachelor and diploma theses. Within the theoretical part of this thesis the reader is informed about entry and reservation systems, as well as with technologies, which were used for creating the application. The practical part is focused directly on the created system. Besides the data tables, it also describes created classes and some interesting scripts that can be found in the application.

Keywords: Internet application, web technologies, entry system, reservation system, script.

Tímto bych chtěl velmi poděkovat vedoucímu své diplomové práce panu Ing. Tomáši Sysalovi, Ph.D. za odborné vedení a spolupráci během řešení mé práce. Dále bych chtěl poděkovat Bc. Martinu Bartákovi za poskytnutí notebooku v době nefunkčnosti mého stolního počítače. V neposlední řadě děkuji své rodině za poskytnuté zázemí a podporu v mém studiu.

Prohlašuji, že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků, je-li to uvolněno na základě licenční smlouvy, budu uveden jako spoluautor.

Ve Zlíně

…….
Podpis diplomanta

OBSAH

ÚVOD

Vývoj moderních technologií je v posledním půlstoletí fenomenální. Zvláště se to týká informačních technologií, kde je rozvoj asi nejvíce patrný. Technologie, které byly před třiceti lety nemyslitelné jsou dnes běžnou součástí našich životů. Příkladem mohou být např. pračky s šestým smyslem, které využívají neuronových sítí nebo navigační systémy pro automobily s vazbou na GPS. Patrný je také rozvoj počítačových sítí a především Internetu. Ten dnes již není pouze prostředníkem k získávání informací, ale firmy přes něj nabízejí své služby, peněžní ústavy správu klientských kont a elektronické peněžní transakce nebo letecké společnosti palubní lístky na lety svých letových flotil. To vše v plně grafických rozhraních, které dnešní technologie umožňují.

Velké oblibě se také těší intranetové systémy. Nejrůznější firmy a organizace si nechávají dělat tyto systémy tzv. na míru. Ty potom zjednodušují a zefektivňují jejich práci. Úlohy, které dříve vyžadovaly složité papírování nebo zdlouhavé hledání v kartotékách a archivech, jsou dnes otázkou několika kliknutí a léty zažitý znechucený výraz na tvářích nejrůznějších administrativních pracovníků se tak pomalu vytrácí.

Záležitostí několika posledních let je rozšíření těchto systémů do škol. Tyto školy pak provozují buď jeden obrovský nebo několik menších více či méně provázaných systémů. Moderní vzdělávací ústavy dnes nabízejí elektronické přihlášky ke studiu, systémy pro přihlašování na zkoušky nebo odevzdávání vypracovaných protokolů a další. Stále ovšem existují případy, na které jsou stávající aplikace „krátké“ a proto vznikají nároky na systémy nové. Takovým systémem by měla být i internetová aplikace, která je náplní této diplomové práce.

I. TEORETICKÁ ČÁST

1 PŘIHLAŠOVACÍ A REZERVAČNÍ SYSTÉMY NA INTERNETU

Od té doby co internet není výsadou několika vyvolených a za připojení k němu se již neplatí horentní sumy se přihlašovací a rezervační systémy staly běžnou součástí lidských životů. Lze dokonce tvrdit, že většina uživatelů po připojení na internet během prvních deseti minut použije jeden nebo několik takových systémů. Jako příklad je možné uvést návštěvu emailové schránky nebo kontrolu odpovědí v internetových diskusích aj.

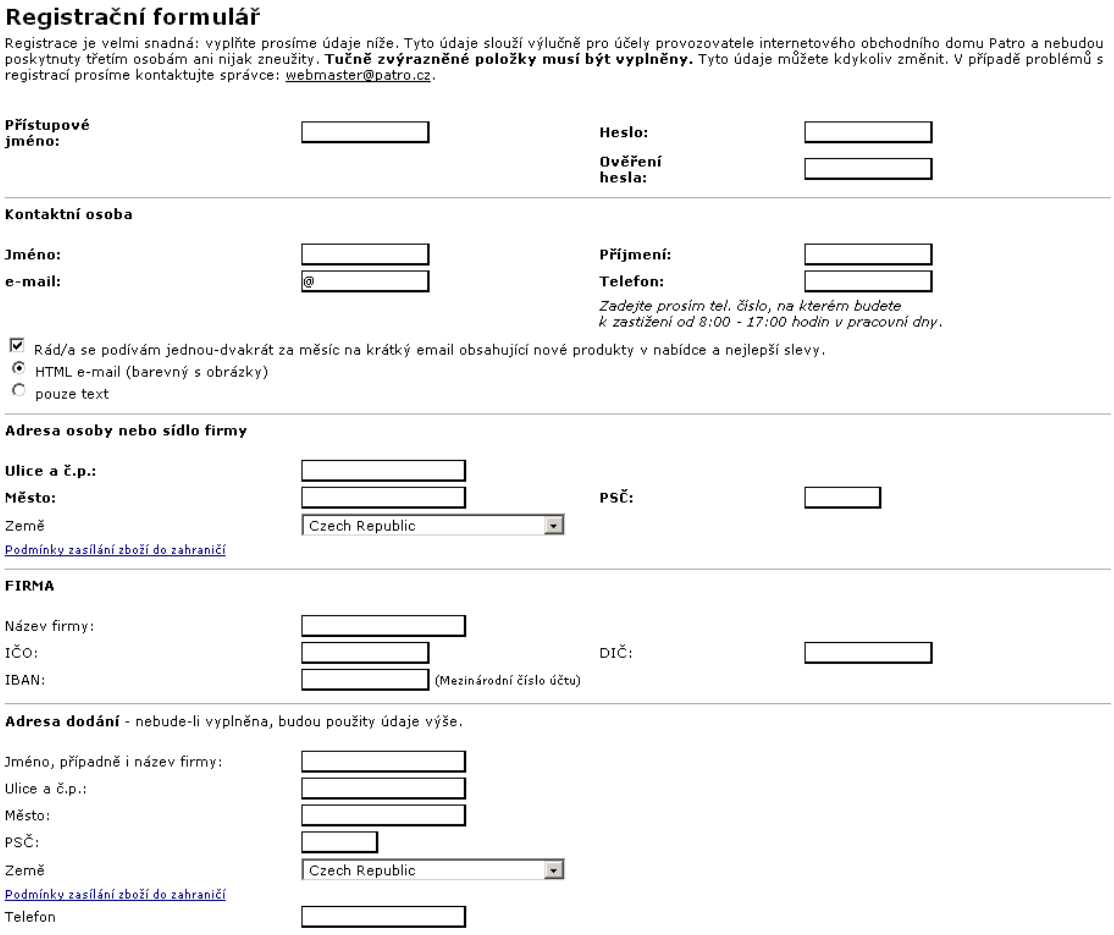
Ačkoliv jak rezervační, tak přihlašovací systémy jsou pojmy obecně známé, není vůbec na škodu popsat si, jaké možnosti tyto systémy nabízejí a jak vlastně fungují.

1.1 Přihlašovací systémy

Přihlašovací systémy slouží obecně k identifikaci, autentizaci či autorizaci uživatele v systému. Takovým systémem může být již dříve zmíněné elektronické bankovnictví nějakého peněžního ústavu, kde uživatel po přihlášení může spravovat svůj účet, internetový obchod, kde může nakupovat, emailová schránka, kde má přehled nad svou korespondencí, ale přihlásit se můžeme i k FTP serveru a spravovat tak soubory vlastních internetových stránek nebo sdílet data.

1.1.1 Registrace

Registrace je jednorázový proces, který předchází vytvoření uživatelského účtu. Zpravidla se jedná o formulář, ve kterém je nutno vyplnit osobní informace a na základě těch je pak účet vytvořen. Mohou však nastat i případy kdy registrace není nutná. To se týká zejména těch případů, kdy informace byly získány již předtím jinou formou, např. při založení bankovního účtu v peněžním ústavu, a služba je tak zřizována automaticky nebo dodatečně na žádost zákazníka. Některé služby při registraci nevyžadují osobní informace vůbec nebo si je žádným způsobem neověřují. V těchto případech často stačí pro registraci pouze emailová adresa, na kterou je pak doručen nějaký typ aktivační zprávy. Tento způsob registrace je typický např. pro diskusní fóra.

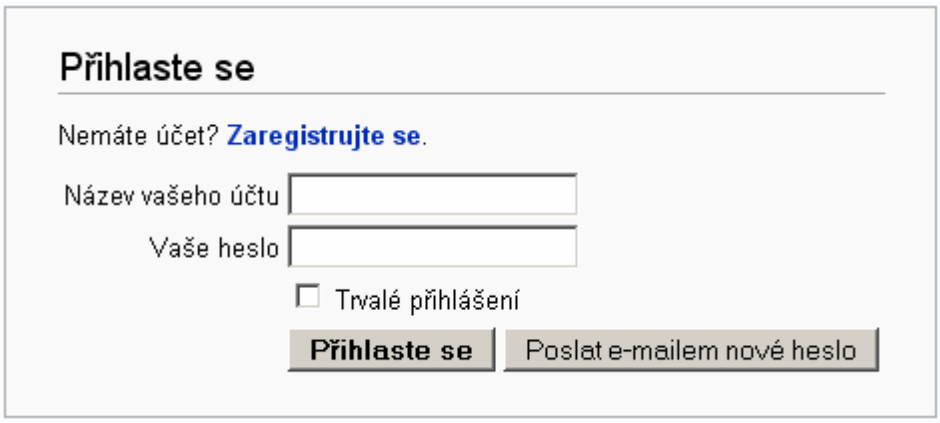


Obrázek 1 - Ukázka registračního formuláře internetového obchodního domu Patro

1.1.2 Přihlášení

Jak již vyplývá ze samotného názvu, hlavní funkcí přihlašovacích systémů je přihlášení. Přihlášení do většiny internetových systémů probíhá pomocí uživatelského jména a hesla. Ve výjimečných případech je nutné zadávat ještě jiné parametry nebo je parametr pouze jeden. Přihlašovací jméno (login) slouží k odlišení uživatele od ostatním uživatelů. Uživatelské jméno je obvykle obecně známé (bývá např. sestaveno ze skutečného jména uživatele), heslo k příslušnému uživatelskému jménu zná pouze uživatel sám.

Samotný princip přihlášení pak funguje následovně: Uživatel zadá již zmíněné jméno a heslo a zmáčkne tlačítko k odeslání těchto údajů. Tyto údaje pak slouží jako vodítko pro hledání záznamu v databázi. Pokud je záznam nalezen a údaje se shodují, pošle se nazpátek kladná odpověď a uživateli je zpřístupněn jeho účet. V opačném případě je uživatel vyzván k opakovanému vložení přihlašovacích údajů.



Obrázek 2 - Přihlašovací formulář wikipedie.org

1.1.3 Bezpečnost

Ačkoliv samotný proces přihlášení se zdá být bezpečným, může docházet k situacím kdy jsou vaše data odposlouchávána a to někde na cestě mezi klientem a serverem. Vaše data tak mohou být potom zpřístupněna třetí osobě. Aby k těmto situacím nedocházelo, je třeba zajistit, aby přenášená data nebyla pouze holým textem (jak je tomu u běžně používaného protokolu HTTP), ale aby komunikace byla šifrována. K těmto účelům se používá protokol HTTPS, který šifruje data pomocí šifrovací vrstvy SSL.

HTTPS

HTTPS je nadstavba počítačového protokolu HTTP, která poskytuje zvýšenou bezpečnost před odposloucháváním či podvržením dat. HTTPS není přímo zvláštní protokol. Data jsou přenášena pomocí HTTP, ale jsou šifrována pomocí SSL (Secure Sockets Layer) nebo TLS (Transport Layer Security – předchůdce SSL), což zaručuje ochranu proti útokům. HTTPS implicitně komunikuje prostřednictvím TCP portu 443. HTTP komunikuje na portu 80.

Pro komunikaci pomocí HTTPS musí nejdříve server vlastnit certifikát. Takovýto certifikát může být vytvořen např. pomocí balíku OpenSSL, který je běžně ke stažení na internetu. Tento certifikát musí být dále podepsán tzv. certifikační autoritou, která zaručí, že vlastník certifikátu se nevydává za nikoho jiného. Organizace si mohou taktéž podepsat certifikáty samy, protože certifikační autority si za podpis certifikátu většinou nechávají dobře zaplatit. V těchto případech však internetové prohlížeče při načítání stránek varují uživatele před nedůvěryhodným podpisem, protože bývají vybaveny jen podpisovými certifikáty největších podpisových autorit (např. VeriSign). [8]

SSL

SSL (Secure Sockets Layer) je protokol, resp. vrstva vložená mezi vrstvu transportní (např. TCP/IP) a aplikační (např. HTTP), která poskytuje zabezpečení komunikace šifrováním a autentizaci komunikujících stran. Po vytvoření SSL spojení (session) je komunikace mezi serverem a klientem již šifrovaná a tedy zabezpečená.

Ustavení SSL spojení funguje na principu asymetrické šifry, kdy každá z komunikujících stran má dvojici šifrovacích klíčů - veřejný a soukromý. Veřejný klíč je možné zveřejnit a pokud tímto klíčem kdokoliv zašifruje nějakou zprávu, je zajištěno, že ji bude moci rozšifrovat jen majitel použitého veřejného klíče svým soukromým klíčem. [7]

Vytvoření SSL spojení tzv. SSL handshake pak probíhá následovně:

1. Klient pošle serveru požadavek na SSL spojení, spolu s různými doplňujícími informacemi (verze SSL, nastavení šifrování atd.).
2. Server pošle klientovi odpověď na jeho požadavek, která obsahuje stejný typ informací a hlavně certifikát serveru.
3. Podle přijatého certifikátu si klient ověří autentičnost serveru. Certifikát také obsahuje veřejný klíč serveru.
4. Na základě dosud obdržených informací vygeneruje klient základ šifrovacího klíče, kterým se bude kódovat následná komunikace. Ten zakóduje veřejným klíčem serveru a pošle mu ho.
5. Server použije svůj soukromý klíč k rozšifrování základu šifrovacího klíče. Z tohoto základu vygenerují jak server, tak klient hlavní šifrovací klíč.
6. Klient a server si navzájem potvrdí, že od teď bude jejich komunikace šifrovaná tímto klíčem. Fáze handshake tímto končí.
7. Je ustaveno zabezpečené spojení šifrované vygenerovaným šifrovacím klíčem.
8. Aplikace od teď dál komunikují přes šifrované spojení. Například POST požadavek na server se do této doby neodešle.

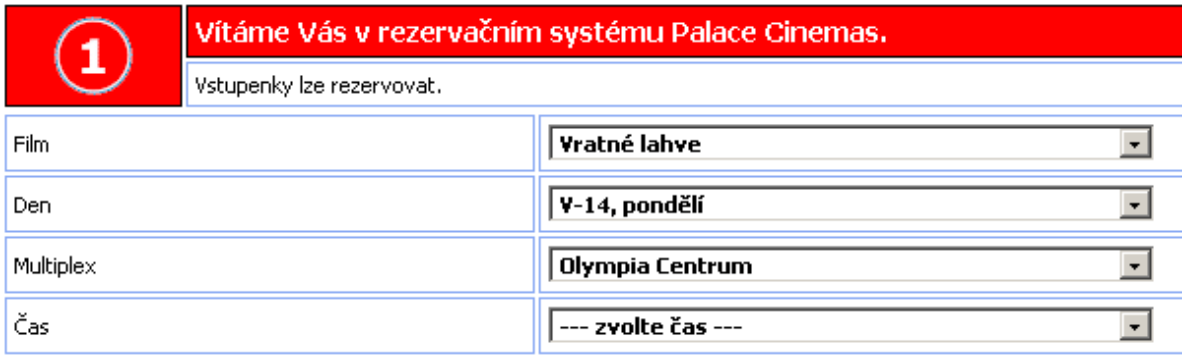
1.2 Rezervační systémy

Rezervační systémy na internetu nahrazují tradiční telefonický způsob vytváření rezervací a nejsou již ani u nás žádnou vymožeností. V dnešní době je možné si přes internet rezervovat lístky do kin nebo divadel, místa v restauracích, pokoje v hotelích, ale nejznámějším a nejlepším příkladem použití takových systémů je pravděpodobně rezervace letenek. Cestovní agentury a letecké společnosti dokonce podporují rezervace letenek touto cestou a to nabídkou zvýhodněných cen. Tyto letenky pak často bývají levnější o stovky, ale i tisíce korun. Elektronicky zakoupenou letenku je buď možné si vyzvednout dodatečně při nástupu cesty, nebo namísto papírové letenky obdrží cestující e-mailem kód, který slouží jako elektronická letenka (tento systém se nazývá e-ticketing). Kód si může cestující vytisknout doma na své tiskárně nebo si ho jen opsat či pamatovat.

1.2.1 Rezervační systém Palace Cinemas

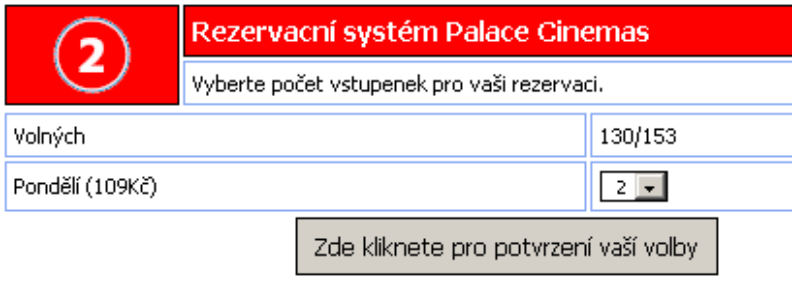
Pro lepší představu o rezervačních systémech je vhodné uvést si nějakou praktickou ukázku. Takovou ukázkou může být např. rezervace lístků do kina Palace Cinemas Olympia v Brně.

Prvním krokem k rezervaci lístků je výběr filmu. Naším filmem budou třeba Vratné láhve. Klikneme tedy na Rezervaci vstupenek a v nově otevřeném okně vyplníme základní informace jako promítací den a čas. Název filmu a kino už bývá přednastaveno podle předchozího pohybu na stránkách.



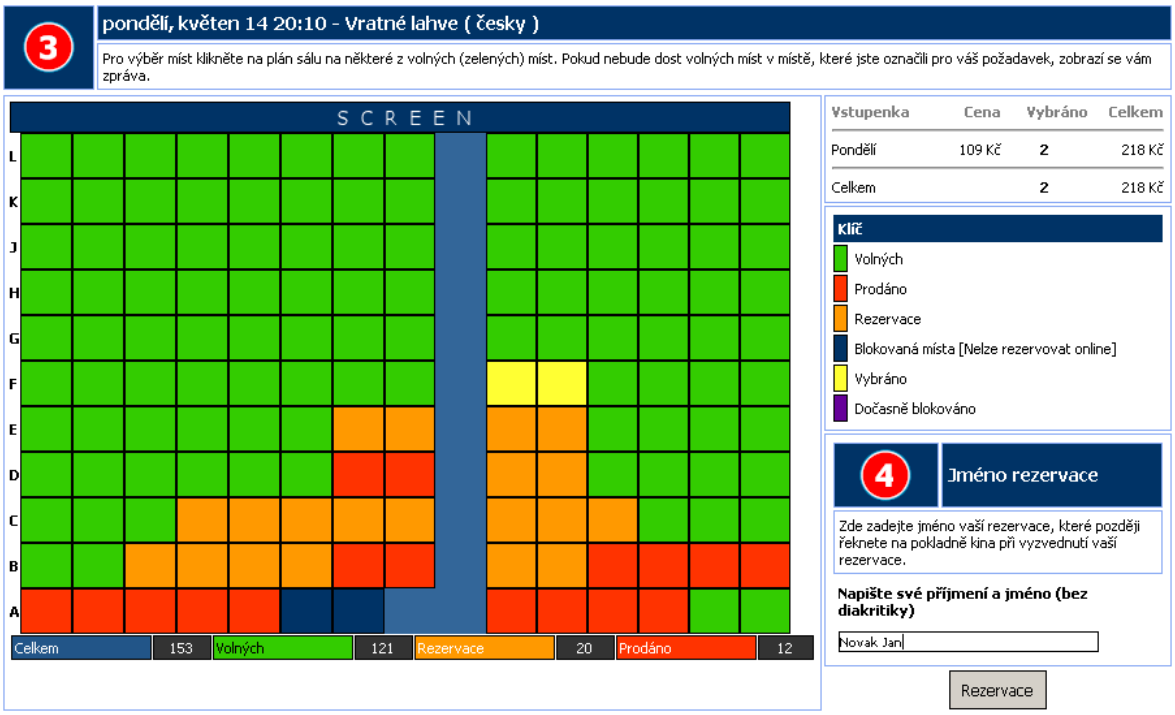
Obrázek 3 – Úvodní tabulka rezervačního systému Palace Cinemas

Po vyplnění vysílacího času se zobrazí číslo kinosálu a další tabulka nás vyzve ke stanovení počtu vstupenek a potvrzení naší volby.



Obrázek 4 – Výběr počtu vstupenek

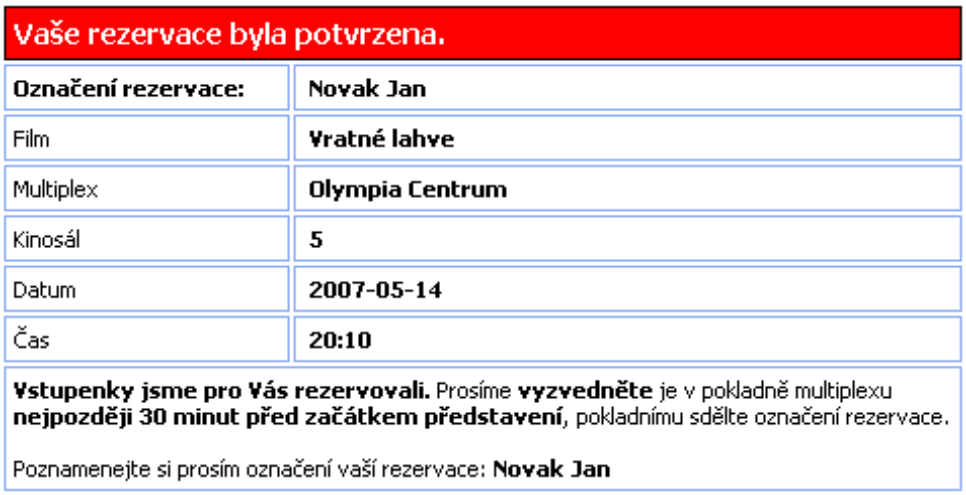
Po potvrzení se zobrazí schématický plán kina, kde si můžete vybrat místa k sezení. Vybraná místa jsou označena žlutou barvou (viz Obrázek 5). Poté napíšete jméno vaší rezervace a zmáčknete tlačítko Rezervace.



Obrázek 5 – Výběr míst k sezení

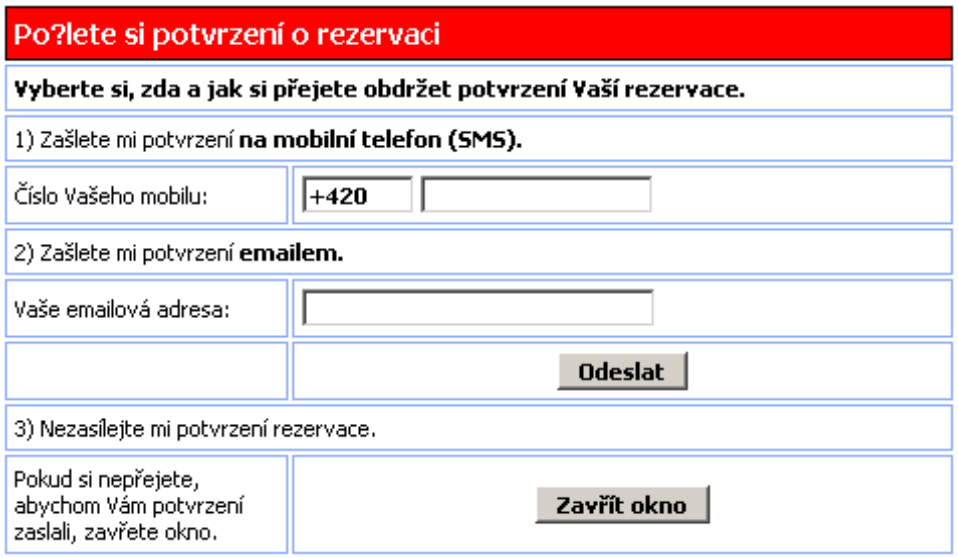
V posledním kroku se vám pak zobrazí potvrzení o vaší rezervaci s upozorněním o vyzvednutí vstupenek nejpozději 30 minut před začátkem promítání a možnost poslat si potvrzení o rezervaci a to buď formou SMS nebo emailem.

Nyní jsou vaše místa rezervována a vy si můžete v klidu vychutnat film.



Vaše rezervace byla potvrzena.	
Označení rezervace:	**Novak Jan**
Film	**Vratné lahve**
Multiplex	**Olympia Centrum**
Kinosál	**5**
Datum	**2007-05-14**
Čas	**20:10**
Vstupenky jsme pro Vás rezervovali. Prosíme **vyzvedněte** je v pokladně multiplexu **nejpozději 30 minut před začátkem představení**, pokladnímu sdělte označení rezervace. Poznamenejte si prosím označení vaší rezervace: **Novak Jan**	

Obrázek 6 - Potvrzení rezervace lístků



Po?lete si potvrzení o rezervaci	
Vyberte si, zda a jak si přejete obdržet potvrzení Vaší rezervace.	
1) Zašlete mi potvrzení **na mobilní telefon (SMS).**	
Číslo Vašeho mobilu:	+420
2) Zašlete mi potvrzení **emailem.**	
Vaše emailová adresa:	
	Odeslat
3) Nezasílejte mi potvrzení rezervace.	
Pokud si nepřejete, abychom Vám potvrzení zaslali, zavřete okno.	Zavřít okno

Obrázek 7 - Tabulka pro možnost zaslání potvrzení registrace

1.2.2 Technologie

Rezervační systémy zpravidla nepoužívají, žádné speciální technologie. O zajištění bezpečnosti komunikace se stará stejně jako u přihlašovacích systémů šifrovaný přenos (tedy HTTPS s SSL). Webové rozhraní pak bývá naprogramováno v PHP a JavaScriptu. Jako databázové systémy bývají nasazovány MySQL u menších systémů řádově do několika set tisíc záznamů. Pro větší systémy jako jsou systémy pro rezervace a správu letenek to pak bývá Oracle.

2 TECHNOLOGIE POUŽITÉ PŘI NÁVRHU APLIKACE

Pro vytvoření tohoto webového portálu bylo třeba použít několika vzájemně provázaných technologií. Použité technologie byly vybrány s ohledem na aktuální vývojové trendy, ale i jejich bezproblémovou dostupnost.

Tyto technologie lze v zásadě rozdělit na dvě skupiny a to:

- **webové technologie**
- **ostatní technologie**

2.1 Webové technologie

Webové technologie se bezprostředně podílejí na funkci celé aplikace. Starají se jak o estetickou stránku internetové aplikace, tak o celkový běh aplikace. Názvy technologií v následujícím výčtu nejsou žádnou novinkou, ale jejich obliba a všeobecná rozšířenost jen dokazuje jejich funkčnost.

Konkrétně se tedy jedná o:

- HTML
- CSS
- JavaScript
- PHP 5 (a knihovna FPDF)
- MySQL

Následující popis jednotlivých technologií je koncipován tak, aby byl spíše obecný a dával čtenáři odpovědi na otázky týkající se obecné rozšířenosti a principu fungování, než aby byl nějakým druhem praktické příručky. Takových je ostatně na knižním trhu i na internetu dostatek.

2.1.1 HTML

HTML je zkratka z anglického HyperText Markup Language. HTML je jedním z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci stránek na Internetu. Jazyk je podmnožinou dříve vyvinutého rozsáhlého univerzálního značkovacího

jazyka SGML (Standard Generalized Markup Language) a jeho vývoj byl ovlivněn vývojem webových prohlížečů, které zpětně ovlivňovaly definici jazyka.

Historie HTML

Počátky HTML jsou situovány na konec osmdesátých let. 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 používal TeX, Postscript a také SGML. Berners-Lee si uvědomoval, že potřebují něco jednoduššího a v roce 1990 byl tedy navržen jazyk HTML a protokol pro jeho přenos v síti HTTP (HyperText Transfer Protocol - přenosový protokol hypertextu). V roce 1991 CERN zprovoznil svůj web. Současně NCSA (National Center for Supercomputer Applications) vybídlo Marca Andreessena a Erica Binu k vyvinutí prohlížeče Mosaic. Ten byl vyvinut v roce 1993 pro počítače PC a Macintosh a měl obrovský úspěch. Mosaic byl prvním prohlížečem s grafickým uživatelským rozhraním a později se stal základem pro Internet Explorer.

Koncepce

Jazyk HTML je od verze 2.0 aplikací SGML. Je charakterizován množinou značek a jejich atributů. Značky jsou uzavřeny do úhlových (ostrých) závorek a toto uskupení se nazývá tag. Mezi tagy se uzavírají části textu dokumentu a tím se určuje význam (sémantika) obsaženého textu. Takto uzavřená část dokumentu tvoří tzv. element (prvek) dokumentu. Součástí obsahu elementu mohou být další vnořené elementy.

Tagy můžeme rozdělit podle typu na:

- **párové** – většina tagů. Rozlišujeme tag počáteční a koncový. Koncový tag má před názvem značky lomítko. Mezi tyto tagy je možné uzavřít část textu. Mezi představitele těchto tagů patří např. odstavec (tagy a) nebo nadpis první úrovně (tagy <h1> a </h1>).
- **nepárové** – mezi nepárové tagy patří např. zalomení řádku (tag
) nebo vykreslení vodorovné čáry (tag <hr>).

Nebo podle významu na:

- **Strukturální značky** - rozvrhují strukturu dokumentu. Příkladem jsou odstavce (<p>), nadpisy (<h1>, <h2>). Dodávají dokumentu formu.
- **Popisné (sémantické) značky** - popisují povahu obsahu elementu. Příklad nadpis (<title>) nebo adresa (<address>). Současný trend je orientován právě na sémantické značky, které usnadňují automatizované zpracovávání dokumentů a vyhledávání informací v záplavě dokumentů na webu. Vyvrcholením této snahy je v současné době jazyk XML.
- **Stylistické značky** - určují vzhled elementu při zobrazení. Typickým příkladem je značka pro tučné písmo (<b>).

Většina tagů také obvykle obsahuje menší či větší počet atributů, které upřesňují vlastnosti elementu. Takovým atributem může být například odkaz (tag <a>), jehož atribut href říká, kam se uživatel po kliknutí na něj dostane (v tomto případě jsou to stránky internetového vyhledávače Google).

```
1 <a href="http://www.google.com/">Google</a>
```

Dokument může mimo normální tagy obsahovat další prvky:

- **Direktivy** - začínají znaky „<!“, jsou určeny pro zpracovatele dokumentu (prohlížeč)
- **Komentáře** - pomocné texty pro programátora, nejsou součástí obsahu dokumentu a nezobrazují se (prohlížeč je ignoruje).
- **Kód skriptovacích jazyků**
- **Definice událostí a kód pro jejich obsluhu**

Struktura dokumentu

Dokument v jazyku HTML má předepsanou strukturu:

- **Deklarace DTD** - je povinná až ve verzi 4.01, je uvedena direktivou <!DOCTYPE
- **Kořenový element** - element html (tagy <html> a </html>) reprezentuje celý dokument. Je nepovinný, ale je doporučeno ho používat.

- **Hlavička elementu** - jsou to metadata, která se vztahují k celému dokumentu. Definují např. název dokumentu, jazyk, kódování, klíčová slova, popis, použitý styl zobrazení. Hlavička je uzavřena mezi tagy <head> a </head>.
- **Tělo dokumentu** - obsahuje vlastní text dokumentu. Vymezuje se tagy <body> a </body>

Příklad struktury dokumentu:

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2 <html>
3 <!-- komentář -->
4    <head>
5       <title>Titulek stránky</title>
6    </head>
7    <body>
8       <h1>Nadpis stránky</h1>
9      <hr>
10       <p>Toto je tělo dokumentu</p>
11   </body>
12 </html>
```

Funkce internetového prohlížeče

Po zadání internetové adresy a jejím odeslání internetový prohlížeč obdrží dokument. Ten je prohlížečem načítán a rozkládán (parsován) na jednotlivé elementy. Každému elementu je poté přiřazen styl (způsob zobrazení). Styly mohou být uvedeny ve stylovém předpisu. Vlastnosti stylů, které nejsou předepsány, doplní prohlížeč podle implicitního stylu, který má zabudován.[6]

2.1.2 CSS

CSS je zkratka pro anglický název Cascading Style Sheets, česky tabulky kaskádových stylů. Častěji se však používá prosté označení kaskádové styly. Je to jazyk pro popis způsobu zobrazení stránek napsaných v jazycích HTML, XHTML nebo XML. Jazyk byl navržen standardizační organizací W3C a zatím byly vydány dvě verze, specifikace CSS1 a CSS2 (plus CSS 2.1). V současné době se pracuje na verzi CSS3.

Hlavním smyslem kaskádových stylů je umožnit návrhářům oddělit vzhled dokumentu od jeho struktury a obsahu. Původně to měl umožnit už jazyk HTML, ale v důsledku

nedostatečných standardů a konkurenčního boje výrobců prohlížečů se vyvinul jinak. Starší verze HTML obsahují celou řadu elementů, které nepopisují obsah a strukturu dokumentu, ale i způsob jeho zobrazení. Z hlediska zpracování dokumentů a vyhledávání informací není takový vývoj žádoucí.

Výhody CSS

Používání kaskádových stylů oproti samotnému HTML přináší v praxi řadu výhod:

- **rozsáhlejší možnosti** – CSS nabízí rozsáhlejší formátovací možnosti než samotné HTML. Např. pro formátování bloku textu – tj. určení vzdálenosti od jejich elementu či okraje stránky nenabízí HTML nic. CSS má vlastnosti padding a margin. V HTML by bylo potřeba vytvořit složitou konstrukci vnořených tabulek.
- **konzistentní styl** – na všech stránkách webové prezentace by měly být všechny nadpisy stejné úrovně, seznamy, zdůrazněné části textu apod. stejného stylu. S použitím formátovacích možností HTML je to obtížné – u každého objektu v každém dokumentu se vzhled objektu stále znovu nastavuje. S použitím CSS je to velmi jednoduché. Vytvoří se soubor stylu, který se připojuje k HTML dokumentu. Ve všech dokumentech jsou pak objekty stejného vzhledu.
- **oddělení struktury a stylu**
- **dynamická práce se styly** – provést změnu stylu webu, který pro formátování vzhledu využívá jen možnosti HTML, znamená najít a nahradit všechny značky a změnit atributy mnoha dalších značek. V případě používání CSS znamená změna stylu webu přepsání jediného souboru – souboru stylů. Internetový prohlížeč Mozilla Firefox dokonce nabízí rozšíření, které umožňuje editovat CSS a přitom sledovat změny ve vzhledu stránky v reálném čase a to bez nutnosti obnovovat obsah stránky (např. pomocí tlačítka F5).
- **formátování XML dokumentů**
- **větší kompatibilita alternativních webových prohlížečů**
- **kratší doba načítání stránky**

Výhodou CSS oproti starému 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 máte v plánu změnu designu webu, stačí změnit pouze jeden soubor .css a změna se aplikuje na celý web. Také se soubor CSS uloží do mezipaměti prohlížeče a pokud není změněn, tak se načítá pouze jednou a zobrazení stránek se velmi urychlí. Mohou také existovat různé styly pro různá výstupní zařízení. Webdesigner má tak možnost prostřednictvím CSS stylů dokumentu určit, jak bude vypadat na papíře, při projekci či na PDA apod. Specifikace CSS nezapomínají dokonce ani na zrakově postižené - je možno napsat styly pro hlasový syntetizátor nebo hmatovou čtečku Braillova písma. Je také možnost upravit formátování podle prohlížeče, kterým si uživatel danou stránku zobrazuje. Jednoduše si vytvoříte více souborů .css (např. styl1.css a styl2.css) a podle prohlížeče, který si o stránku požádá, připojíte jiný soubor. Tím se dá eliminovat problém různé interpretace kódu jednotlivými prohlížeči.

Nevýhody CSS

Hlavní nevýhodou CSS je zatím stále špatná podpora v majoritních prohlížečích. Různé prohlížeče interpretují stejný CSS kód jinak a je někdy velmi obtížné jej napsat tak, aby se na všech (resp. na několika vybraných) prohlížečích výsledek zobrazil stejně. Pravděpodobně největším hříšníkem v tomto směru je na poli moderních prohlížečů Internet Explorer, který si do současné doby drží majoritní podíl uživatelů.

Syntaxe CSS

Stylový předpis se skládá z posloupnosti pravidel. Každé pravidlo určuje vzhled některého elementu dokumentu, nebo skupiny elementů. Pravidlo začíná tzv. selektorem, který specifikuje („adresuje“) skupinu elementů. Selektor je následován seznamem deklarací, které určují vzhled vybrané skupiny elementů. Celý seznam je uzavřen ve složených závorkách a jednotlivé deklarace jsou odděleny středníkem (tj. za poslední deklarací středník už být nemusí). [10]

Příklad kaskádových stylů:

```
1 h1 {                  /* vzhled nadpisu první úrovně */
2       margin: 5px;    /* okraj šířky 5 pixelů        */
3     font-size: 12pt } /* velikost fontu 12 bodů      */
```

2.1.3 JavaScript

JavaScript je platformě nezávislý, objektově orientovaný skriptovací jazyk, který se v současné době používá jako interpretovaný programovací jazyk pro WWW stránky. Program v JavaScriptu se obvykle spouští až po stažení WWW stránky z Internetu (tzv. na straně klienta), na rozdíl od jiných interpretovaných programovacích jazyků (např. PHP a ASP), které se spouštějí na straně serveru ještě před stažením z Internetu. Z toho plynou jistá bezpečností omezení, JavaScript např. nemůže pracovat se soubory, aby tím neohrozil soukromí uživatele. Samotný kód se vkládá přímo do HTML kódu stránky a jsou jím zpravidla ovládány různé interaktivní prvky GUI (Graphical User Interface – Grafické Uživatelské rozhraní), jako např. tlačítka, textová políčka aj. nebo tvořeny animace a efekty obrázků.

Historie JavaScriptu

Autorem Javascriptu je Brendan Eich. JavaScript byl původně obchodní název implementace společnosti Netscape, kde byl vyvíjen nejprve pod názvem Mocha, později LiveScript, ohlášen byl společně se společností Sun Microsystems v prosinci 1995 jako doplněk k jazykům HTML a Java. Pro verzi firmy Microsoft je použit název Jscript.

Jeho syntaxe patří do rodiny jazyků C/C++/Java. Slovo Java je však součástí jeho názvu pouze s marketingových důvodů a s programovacím jazykem Java jej vedle názvu spojuje jen podobná syntaxe. JavaScript byl v červenci 1997 standardizován asociací ECMA (Europen Computer Manufacturers Association) a v srpnu 1998 ISO (International Standards Organization). Standardizovaná verze JavaScriptu je pojmenována jako ECMAScript a z ní byly odvozeny i další implementace, jako je například ActionScript.[9]

Princip Javascriptu

Spolu s HTML stránkou je prohlížeči odeslán i nějaký kus JavaScriptového kódu a ten je ve vhodnou chvíli na "cílovém" počítači spuštěn. Vhodná chvíle může nastat například při kliknutí na tlačítko, při najetí myší na odkaz, při otevření okna prohlížeče a podobně. O spuštění klientského kódu se stará prohlížeč. V prohlížeči tedy musí být JavaScript podporován.

V zásadě existují 3 způsoby jak začlenit JavaScript do stránky:

- **Pomocí tagu <script> normálně do proudu dokumentu** - skript se může objevit kdekoliv v HTML kódu -- jak v hlavičce, tak v těle dokumentu. Prohlížeč pak skript zpracovává okamžitě, jakmile na něj narazí.

  Příkladem může být například následující skript zobrazující hlášku „Hello world!“.

```
1 <script type="text/javascript"> alert("Hello world!"); </script>
```

- **Tagem <script> s odkazem na externí soubor** – stejné řešení jako minule, jen s tím rozdílem, že skript je uložen v externí souboru a je do dokumentu vkládán příkazem.

  V našem případě je skript uložen v externím souboru s názvem externi_skript.js.

```
1 <script src="externi_skript.js"></script>
```

- **In-line (řádkový) zápis jako atribut tagu** – tento způsob nevyužívá tag <script>, ale zapisuje se jako atribut jiného tagu. Většinou reaguje na nějakou uživatelovu událost. Seznam těchto událostí je možné nalézt na internetu. V následujícím příkladu je touto událostí najetí myši (kurzoru) na odkaz.

```
1 <a href="#" onmouseover="alert('Sem nechoď!!')">Odkaz</a>
```

2.1.4 PHP

PHP (PHP: Hypertext Preprocesor, původně Personal Home Page) je skriptovací programovací jazyk, určený především pro programování dynamických internetových stránek. PHP skripty jsou prováděny na straně serveru, k uživateli je přenášen až výsledek jejich činnosti. Syntaxe jazyka kombinuje hned několik programovacích jazyků (Perl, C, Pascal a Java). PHP je nezávislý na platformě - skripty fungují bez úprav na mnoha různých operačních systémech. Obsahuje rozsáhlé knihovny funkcí pro zpracování textu, grafiky, práci se soubory, přístup k většině databázových serverů (mj. MySQL, ODBC, Oracle, PostgreSQL, MSSQL) a podporu celé řady internetových protokolů (HTTP, SMTP, SNMP, FTP, IMAP, POP3, LDAP, …).

PHP se stalo velmi oblíbeným především díky jednoduchosti použití a tomu, že kombinuje vlastnosti více programovacích jazyků a nechává tak vývojáři částečnou svobodu v syntaxi. V kombinaci s databázovým serverem (především s MySQL nebo PostgreSQL) a webovým serverem Apache je často využíván k tvorbě webových aplikací. Díky velmi častému nasazení na serverech se vžila zkratka LAMP – tedy spojení Linux, Apache, MySQL a PHP (nebo Perl).[4]

Obrázek 8 - Logo PHP

Historie PHP

Počátky PHP spadají do roku 1994. Tehdy se Rasmus Lerdorf rozhodl vytvořit jednoduchý systém pro počítání přístupu ke svým stránkám. Ten byl napsán v jazyce PERL. Za nějakou dobu byl však systém přepsán do jazyka C, protože perlovský kód poměrně hodně zatěžoval server. Sada těchto skriptů byla ještě téhož roku vydána pod názvem "Personal Home Page Tools", zkráceně PHP. U toho však nezůstalo. V polovině roku 1995 se systém PHP spojil s jiným programem stejného autora, a to sice s nástrojem "Form Interpreter" neboli zkráceně FI. Tak vznikl systém PHP/FI 2.0, který si postupně získal celosvětovou proslulost a byl velmi rozšířen. V roce 1997 Zeev Suraski a Andi Gutmans přepsali parser a zformovali tak základ PHP 3. Současně byl název změněn na dnešní podobu. PHP 3.0 vyšlo koncem roku 1998. Oproti verzi 2.0 bylo mnohem rychlejší, obsahovalo více funkcí a také již běželo i pod operačním systémem Windows. V roce 2000 vychází PHP verze 4, o čtyři roky později pak verze 5 s plnou podporou objektově orientovaného přístupu, podobného jazyku Java.

Princip PHP

Jak již bylo řečeno, PHP je technologie běžící na straně serveru. Typický PHP skript obsahuje jednak kusy normálního HTML kódu, a jednak kusy programového kódu. Když webový server obdrží požadavek na zpracování takového skriptu, vezme kusy HTML kódu tak jak jsou, dále zpracuje části PHP programového kódu a nakonec tyto dvě části zkombinuje a odešle prohlížeči. Server takto provede jednu nebo více operací (podle rozsáhlosti a složitosti skriptu), ale internetový prohlížeč obdrží pouze prostou HTML stránku. Pro správnou funkci vašich PHP skriptů je, kromě správné syntaxe, třeba dodržet dvě základní věci.

První věcí je správná přípona souboru s PHP skriptem. Díky příponě server pozná, že má v souboru hledat příkazy PHP. Možností je hned několik, ale obvykle se používá přípona

.php. Dalšími příponami pak mohou být například .php3 nebo .phtml. Úplná katastrofa může nastat, pokud soubor se skriptem PHP má příponu .htm nebo .html. Nejenže jej server pošle do prohlížeče bez jakýchkoli úprav a tedy nedosáhneme chtěného efektu, ale uživatel může vidět zdrojový kód skriptu a tak se dostat k citlivým informacím jako jsou např. přihlašovací údaje k databázi apod. Na druhou stranu neexistuje žádné omezení proč nedat „obyčejným“ html souborům příponu .php. Pokud totiž server zjistí, že v nich není žádný php kód, pošle je prohlížeči tak, jak jsou a navíc pokud později bude potřeba do nich nějaký PHP kód přidat, už se nemusí přejmenovávat.

Druhou věcí je pak správné oddělení PHP kódu od HTML. V zásadě existují dva správné způsoby. Prvním je použití oddělovacích znaků <?php a ?>. Mezi těmito pak může být jakýkoli PHP skript. Příkladem pak může být následující kód, který na obrazovku vypisuje hlášku „Hello world!“.

```
1 <?php echo "Hello world!"; ?>
```

Druhý správný způsob je syntakticky čistší a je kompatibilní se standardem XML, ale vzhledem k jeho délce nebývá moc často používán.

```
1 <SCRIPT LANGUAGE="php"> echo "Hello world!"; </SCRIPT>
```

Existují i další způsoby, pomocí kterých je možno vkládat PHP skripty do těla dokumentu, ale ty se zpravidla nedoporučují, protože na různých webserverech nemusí být podporovány a může dojít k odeslání skriptů v nezpracované podobě stejně jako při nesprávné příponě souboru. Takovýmto způsobem může být i použití zkráceného oddělovače <? a ?>.[11]

Umístění PHP skriptu v těle dokumentu pak může být libovolné s ohledem na to, že celý dokument je zpracováván od prvního řádku k poslednímu (ostatně jako každý jiný).

2.1.5 FPDF

FPDF je PHP třída pro vytváření PDF dokumentů v rámci webových aplikací jejímž autorem je Olivier Plathey. Tato třída je šířená pod freeware licencí a je ji tedy možné použít jak v komerčních, tak nekomerčních projektech a to zcela zdarma. FPDF funguje nezávisle na jiných rozšířeních PHP (kromě zlib pro kompresi) a je tedy možné ji využít pro generování PDF souborů i na serverech, kde není nainstalována knihovna PDFlib, pomocí které by se soubory PDF vytvářely za „normálních“ okolností. Knihovna PDFlib je

navíc zpoplatněna a tak je její použití na jiné než nekomerční projekty podmíněno koupí licence. Díky nezávislosti na ostatních knihovnách je tedy možné vytvořit webovou aplikaci s možností vytvářet PDF soubory bez nutnosti spoléhat se na nastavení serveru.

Ačkoliv má třída FPDF menší výkon než knihovna PDFlib, která je v podstatě volitelnou součástí PHP, ostatní její vlastnosti jí dávají neskutečný potenciál a o její oblíbenosti hovoří i velký počet rozšíření a nadstaveb nebo to, že je použita i v jednom z nejlepších nástrojů pro správu MySQL databází s názvem phpMyAdmin.

Obrázek 9 - Logo FPDF

2.1.6 MySQL

MySQL je databázový systém, vytvořený švédskou firmou MySQL AB. Jeho hlavními autory jsou Michael „Monty“ Widenius a David Axmark. MySQL je považováno za úspěšného průkopníka dvojího licencování, protože je k dispozici jak pod bezplatnou licencí GPL (General Public License), tak pod komerční placenou licencí. Jak již název napovídá, komunikace s MySQL probíhá pomocí jazyka SQL (Structured Query Language) a podobně jako u jiných SQL databází se jedná o dialekt tohoto jazyka s některými rozšířeními.[5]

MySQL je multiplatformní databáze a lze ji tedy nainstalovat na většinu operačních systémů. Z těch největších je možné uvést např. MS Windows, GNU/Linux nebo Mac OS X. Kompletní seznam operačních systémů je uveden na oficiálních internetových stránkách.

MySQL bylo od počátku optimalizováno především pro rychlost, a to i za cenu některých zjednodušení. MySQL má jen jednoduché způsoby zálohování, a až donedávna nepodporovalo některé vyspělé funkce jakými jsou pohledy (předpisy toho, jakým způsobem mají být získána data z tabulek), triggery (definují činnosti, které se mají provést v případě definované události nad databázovou tabulkou) nebo uložené procedury

(databázové objekty, které obsahují část programu, který se nad daty v databázi má vykonávat).

Pro svou snadnou implementovatelnost, výkon a především díky tomu, že se jedná o volně šiřitelný software, má vysoký podíl na trhu současně používaných databází. Velmi oblíbená a často nasazovaná je pak již dříve zmíněná kombinace MySQL, PHP a Apache jako základní software webového serveru.

Obrázek 10 - Logo MySQL

Pro komunikaci s databází se používají PHP funkce, které zprostředkovávají přístup na MySQL databázový server. Bližší informace je možné nalézt na oficiálních stránkách MySQL nebo PHP.

2.2 Ostatní technologie

Tyto technologie doplňují nebo zlepšují funkci celé aplikace. Ačkoliv by se na první pohled mohlo zdát, že jsou podružné nebo nedůležité, opak je pravdou. Hašovací funkce je např. velmi silným nástrojem a i při průniku do databáze zamezí ve většině případů prozrazení uživatelských hesel, která by jinak mohla být zneužita.

2.2.1 Hašovací funkce MD5

Než se dostaneme k samotnému algoritmu MD5. Je nejprve důležité vysvětlit pojem hašovací funkce.

Hašovací funkce

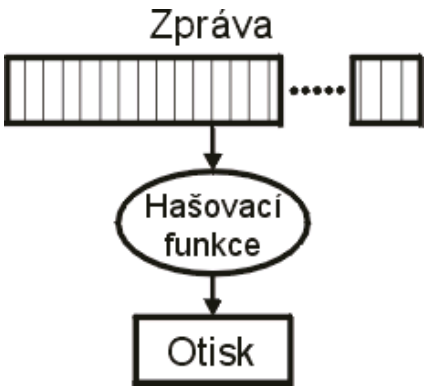
Hašovací funkce (z angl. hash function) je silným nástrojem moderní kryptologie. Je jednou z klíčových myšlenek druhé poloviny dvacátého století a přinesla řadu nových použití.

Hašovací funkce spadá do oblasti šifrování, kde potřebujeme informaci pouze zašifrovat, ale už nikdy nedešifrovat. Nejčastěji se používá k ukládání hesel, ale může sloužit také ke kontrole integrity dat, k rychlému porovnání dvojice zpráv, indexování, vyhledávání a je také důležitou součástí kryptografických systémů pro digitální podpisy.

Pro lepší pochopení si uveďme jednu z možných definic:

Hašovací funkce je funkce h, která má přinejmenším tyto vlastnosti: je kompresní - provádí mapování argumentu /vstupu/ x libovolné bitové délky na hodnotu h(x) /výstup/, která má pevně určenou bitovou délku, je snadno vypočitatelná – pro dané h a argument x je snadné vypočítat h(x).

Tuto vlastnost ilustruje následující obrázek, na němž je argument označen jako „zpráva“ a hodnota funkce slovem „otisk“.



Obrázek 11 - Kódování zprávy hašovací funkcí

Smyslem použití hašovacích funkcí obecně bývá možnost reprezentovat potenciálně i velmi dlouhou zprávu jen jejím krátkým otiskem a vytvořit pomyslnou relaci 1:1 mezi zprávou a otiskem. Důležitou vlastností hašovacích funkcí je pak to, že na tentýž vstup odpovídá tímtéž výstupem. Naopak při sebemenší změně vstupu je výstup zcela odlišný.

Příklad:

MD5("The quick brown fox jumps over the lazy dog") =
9e107d9d372bb6826bd81d3542a419d6

MD5("The quick brown fox jumps over the lazy cog") =
1055d3e698d289f2af8663725127bd4b

Praktické použití hašovacích algoritmů potom vypadá následovně. Představme si, že naší zprávou je heslo, např. k emailové schránce, diskuznímu fóru apod. Toto heslo je při přihlašování zašifrováno hašovací funkcí a takto vytvořený otisk je potom porovnáván s otiskem uloženým v databázi. Dojde-li ke shodě těchto dvou otisků, pokračujeme dále a jsme přihlášeni do systému.

MD5

MD5 (Message-Digest algorithm 5) je hašovací funkce s otiskem (nebo chcete-li hašovým kódem) o velikosti 128 bitů. Tato hašovací funkce je popsána v internetovém standardu RFC 1321 a prosadila se do mnoha aplikací, např. pro kontrolu integrity souborů nebo ukládání hesel.

Algoritmus MD5 byl vytvořen v roce 1991 (Ronaldem Rivestem), aby nahradil dřívější hašovací funkci MD4. V roce 1996 pak byla objevena vada v návrhu MD5 a ačkoli nebyla zásadní, kryptologové začali raději doporučovat jiné algoritmy, jako je například SHA-1 se 160-ti bitovým hašovým kódem. Ten se však také ukázal nebýt bezchybným.[3]

I když v dnešní době již existují metody, které dokáží generovat kolize a tedy prolomit tuto hašovací funkci dokonce i na obyčejném notebooku, je hašovací funkce MD5 stále hojně využívána. V kritických bezpečnostních aplikacích se od této metody však upouští.

II. PRAKTICKÁ ČÁST

3 REALIZACE INTERNETOVÉ APLIKACE

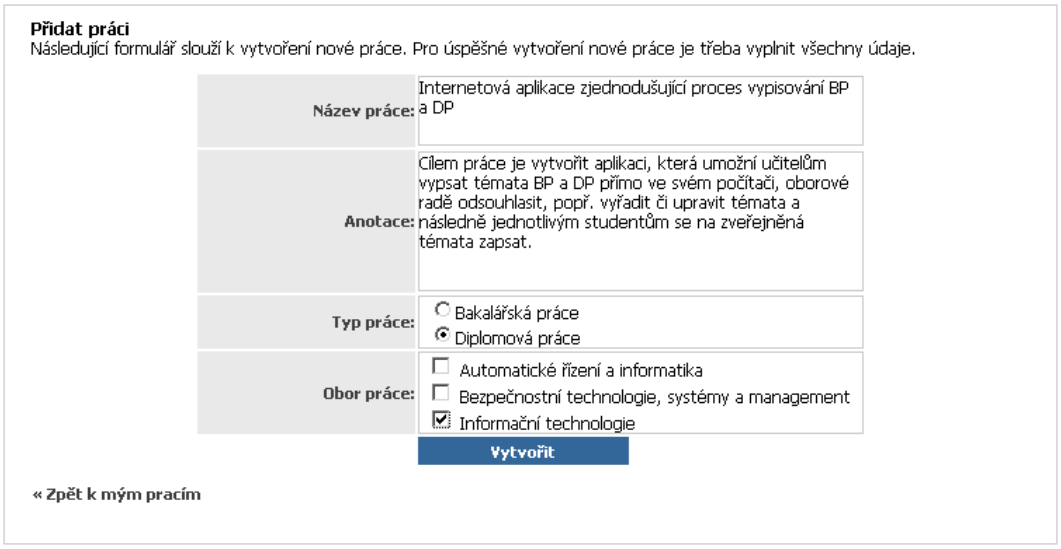
Hlavní částí této diplomové práce bylo vytvoření internetové aplikace respektive portálu pro správu bakalářských a diplomových prací. Tato část se tedy bude zabývat právě jí. Nebude zde však vysvětleno uživatelské ovládání, protože to je podrobně popsáno v manuálu, ale bude zde naopak uvedena funkce a propojení databázových tabulek, vysvětleny naprogramované třídy a rozebrány některé složitější operace, ke kterým dochází při běžném provozu systému.

3.1 Agenda aplikace

Přestože jsou všechny funkce systému podrobně popsány v manuálu, pro úplné porozumění tomu jak aplikace funguje je třeba ještě uvést agendu aplikace. Tato agenda vystihuje pracovní náplň aplikace tak, jak by měla vypadat po dobu celého jednoho roku. Chronologicky vypsané události budou pro lepší představu doplněny o screenshoty jednotlivých kroků.

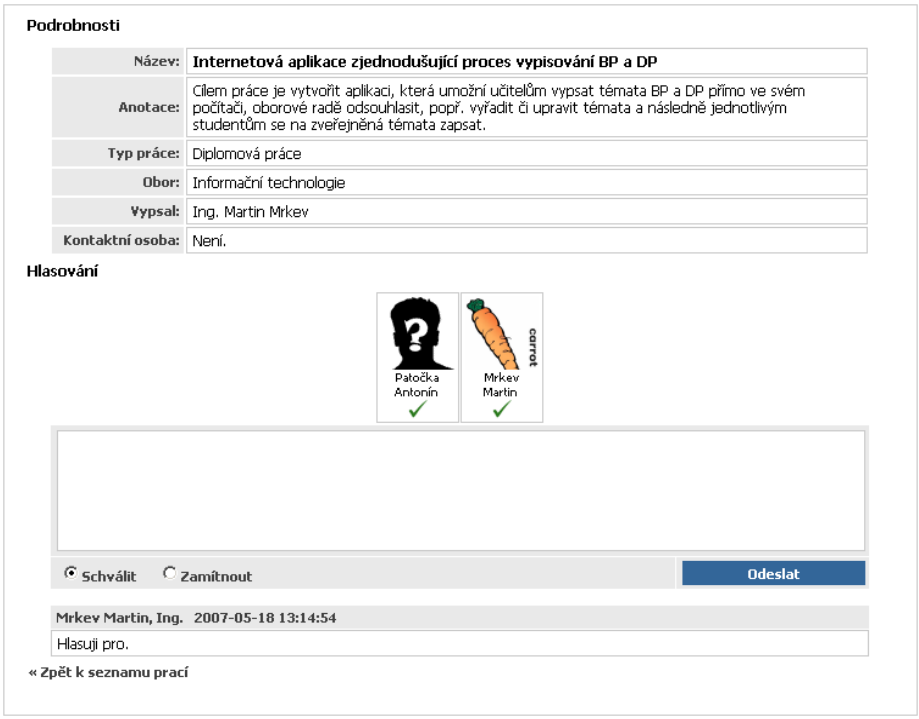
Výpis událostí:

- učitel vypíše práci



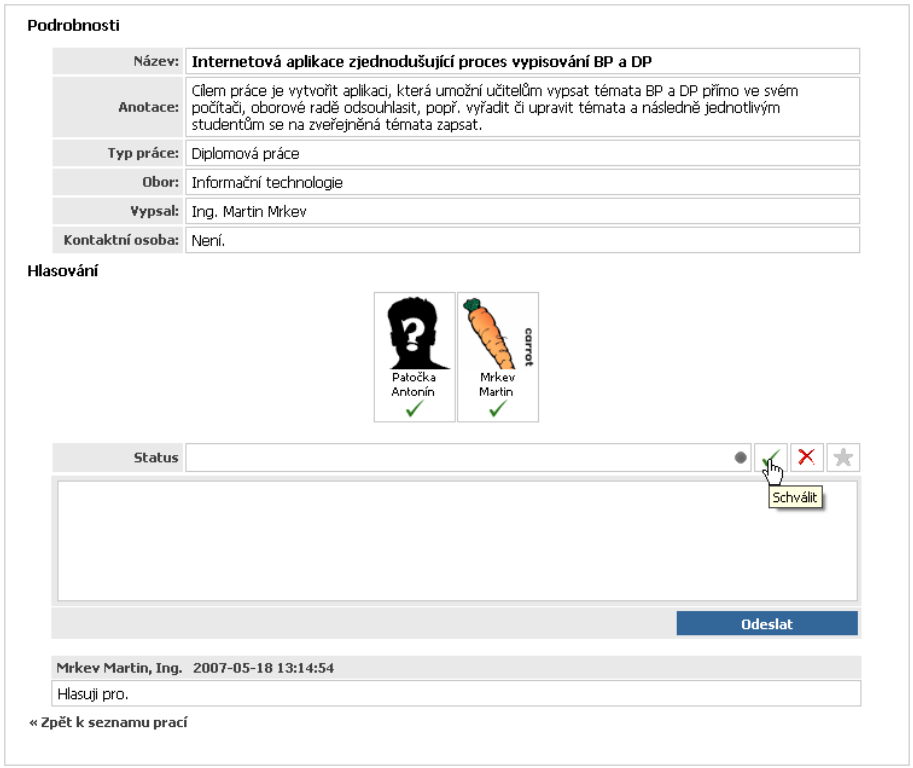
Obrázek 12 - Formulář pro vytvoření nové práce

- členové oborové rady hlasují pro schválení nebo zamítnutí práce



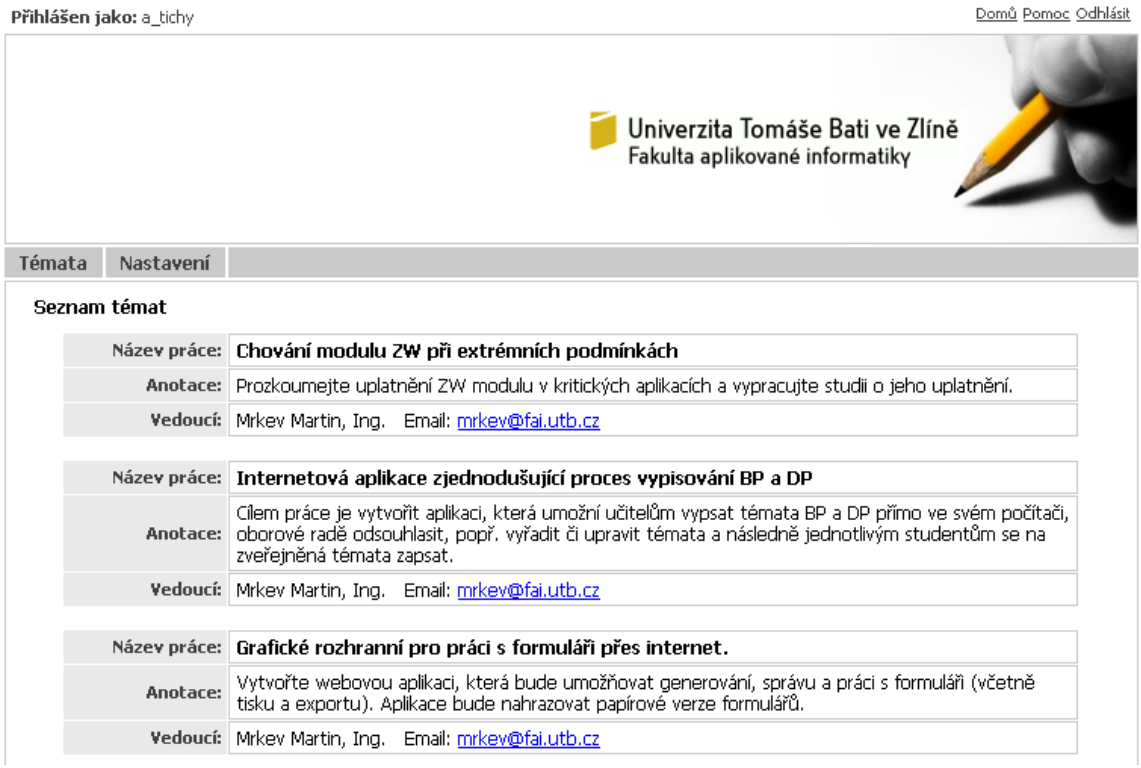
Obrázek 13 - Rozhraní pro hlasování a diskuzi k tématům

- předseda oborové rady práci na základě připomínek buď schválí nebo neschválí



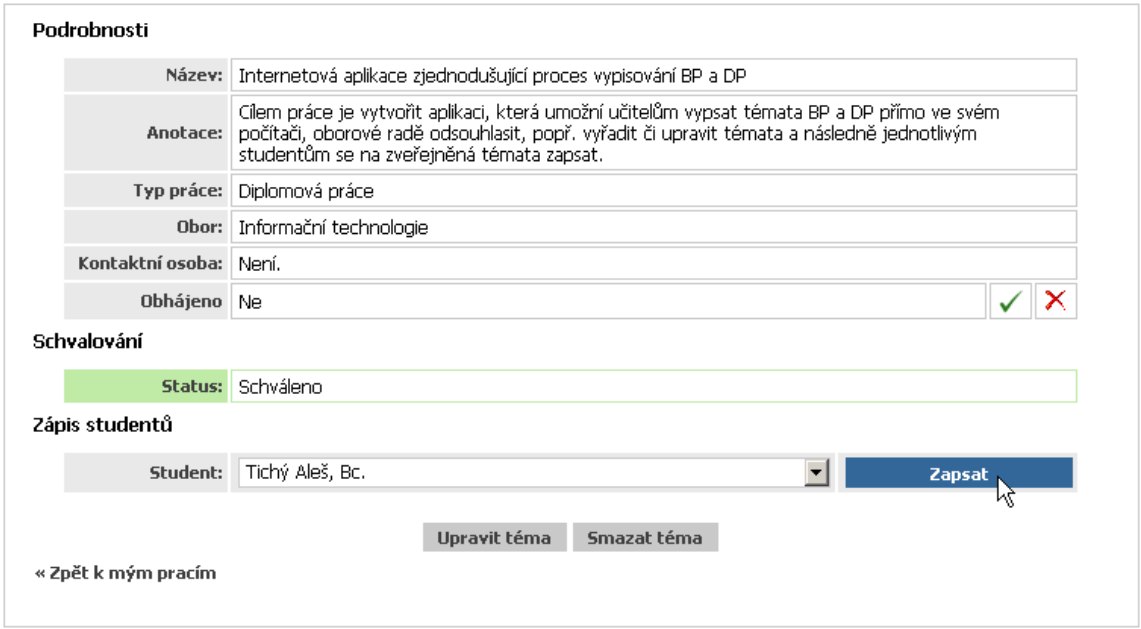
Obrázek 14 - Schválení práce předsedou OR

- student si vybere ze seznamu schválených prací



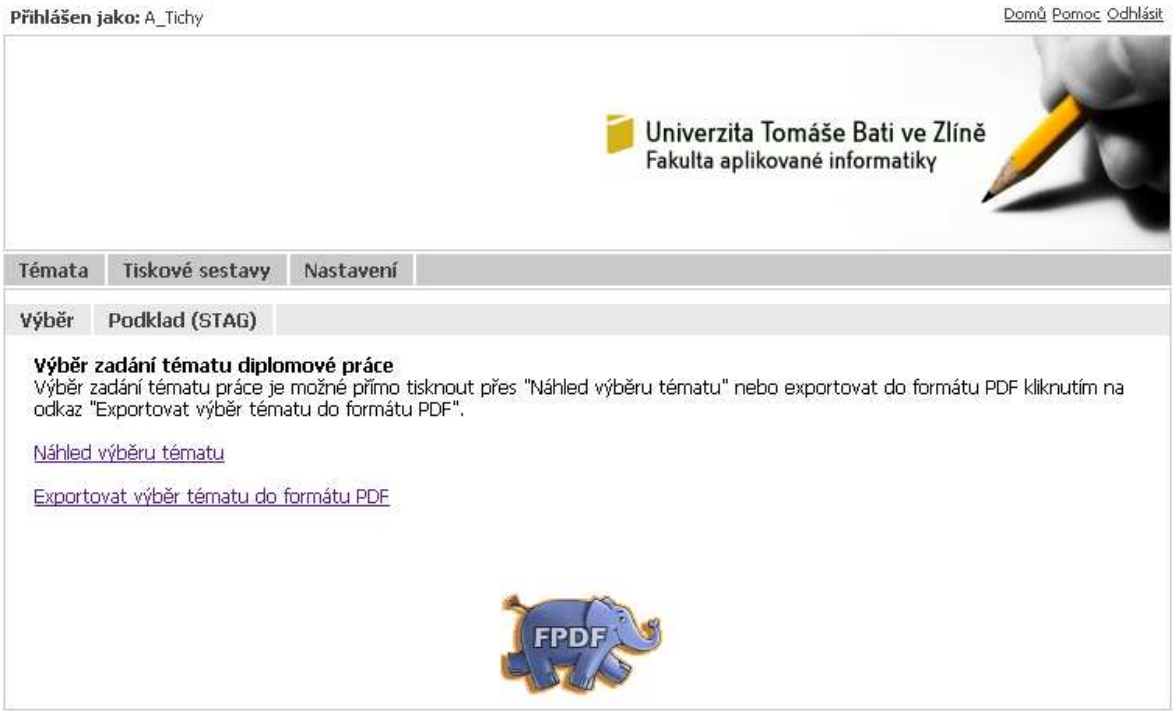
Obrázek 15 - Seznam prací v rozhraní pro studenty

- učitel zapíše studenta na práci



Obrázek 16 - Zapsání studenta na práci

- student tiskne výběr zadání tématu a zadává exportovaný podklad do systému STAG



Obrázek 17 - Rozhraní pro tisk a export výběru zadání práce

- pokud student úspěšně práci obhájil, učitel označí práci za obhájenou

Obrázek 18 - Označení práce jako obhájené

- nakonec jsou generovány statistiky uplynulého roku

Počet prací na učitele

Jméno	Celkem	BP		DP		Zapsaných celkem
		Počet	Zapsaných	Počet	Zapsaných	
Kalich Karel	1	1	0	0	0	0
Mrkev Martin, Ing.	5	0	0	5	2	2
Patočka Antonín, Ing.	0	0	0	0	0	0
Sysala Tomáš	0	0	0	0	0	0

Počet prací na ústav

Ústav	Celkem	BP		DP		Zapsaných celkem
		Počet	Zapsaných	Počet	Zapsaných	
Ústav aplikované informatiky	6	1	0	5	2	2
Ústav automatizace a řídící techniky	0	0	0	0	0	0
Ústav elektrotechniky a měření	0	0	0	0	0	0
Ústav matematiky	0	0	0	0	0	0
Ústav řízení procesů	0	0	0	0	0	0

Počet prací na obor

Obor	Jednoob.					Viceob.			Zapsaných celkem
	Celkem	BP		DP		Celkem*	Zapsané		
		Počet	Zapsaných	Počet	Zapsaných		BP	DP	
Automatické řízení a informatika	0	0	0	0	0	1	0	0	0
Bezpečnostní technologie, systémy a management	1	0	0	1	0	1	0	0	0
Informační technologie	4	1	0	3	2	1	0	0	2

Obrázek 19 - Ukázka generovaných statistik

3.2 Znakové sady

V tomto projektu bylo použito dvou znakových sad. Pro grafické uživatelské rozhraní bylo zvoleno kódování windows-1250. Naopak pro tabulky v databázi bylo zvoleno kódování UTF-8 a collation utf8_czech_ci. Bezproblémové společné fungování těchto dvou znakových sad dohromady zajišťuje třída cMysql, která obsahuje volání funkce pro konverzi znakových sad mezi sebou.

3.3 Deskripce databázových tabulek

Celá aplikace je tvořena sadou několika tabulek. Základní počet tabulek pro plnohodnotnou funkci systému je 11, avšak skutečný počet tabulek bude narůstat s dobou používání. Tento nárůst tabulek pak bude výsledkem funkce $6+(x*5)$, kde neznámou x je počet vytvoření databázových tabulek obsahujících informace, které se s každým rokem mění.

Následující podkapitoly obsahují popis jednotlivých tabulek a jejich funkce v systému, a to nejprve „statické“ šestice a poté zbylé pětice (řetězec na místě sufixu je nahrazen hvězdičkou). Jednotlivé skupiny tabulek jsou seřazeny abecedně. U každé tabulky je vždy uveden MySQL příkaz pro vytvoření dané tabulky, a to z důvodu rozlišení datových typů jednotlivých řádků tabulky a tím pádem zamezení zbytečného natahování textu diplomové práce jejich vypisováním. MySQL příkazy jsou navíc velmi dobře čitelné a tak by s tímto zápisem neměl mít problém ani člověk, který má o MySQL databázích jen základní povědomí.

Tabulka 1 - Funkce jednotlivých tabulek

Název tabulky	**Funkce**
activetable	Tabulka obsahuje sufix právě používané pětice tabulek
adminlogin	Tabulka administrátorských účtů
kontexty	Tabulka kontextů sítě Novell
obor	Tabulka obsahuje informace o oborech
ucitele	Tabulky obsahuje učitelské profily
ustav	Tabulka obsahuje seznam ústavů
diskuse_*	Tabulka s příspěvky ke schvalovaným pracím
hlasovani_*	Tabulka údajů o hlasování při schvalovaní prací
pocty_*	Tabulka obsahuje počty studentů a
práce_*	Tabulka obsahuje informace o vypsaných pracích
studenti_*	Tabulka obsahuje informace o studentech

3.3.1 Tabulka „activetable“

Tato tabulka obsahuje jediný parametr *suffix*, který je příponou právě těch tabulek, které se budou každý rok měnit. Konkrétně se to týká tabulek s informacemi o studentech, počtu studentů, pracích, diskuzích a hlasování.

```
1 CREATE TABLE `activetable` (
2   `suffix` text collate utf8_czech_ci NOT NULL
3 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;
```

3.3.2 Tabulka „adminlogin“

Tabulka adminlogin je tabulkou administrátorských účtů. Sloupec *login* obsahuje uživatelské jméno administrátora, sloupec *password* heslo kódované hašovacím algoritmem MD5 a sloupec *priznak* obsahuje příznak o typu administrátorského účtu. Je-li tento příznak nastaven na hodnotu 0 (tedy defaultní hodnotu), jedná se o běžný

administrátorský účet. Je-li naopak nastavena na hodnotu 1, jedná se o nadstandardní administrátorský účet s možností vytvářet další účty.

```
1 CREATE TABLE `adminlogin` (
2   `id_admin` int(10) unsigned NOT NULL auto_increment,
3   `login` char(30) collate utf8_czech_ci NOT NULL,
4   `password` text collate utf8_czech_ci NOT NULL,
5   `priznak` tinyint(4) default '0',
6   PRIMARY KEY  (`id_admin`)
7 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;
```

3.3.3 Tabulka „kontexty“

Tato tabulka obsahuje kontexty ze školní sítě Novell pro uživatelské přihlášení do systému. Parametr *kontext* obsahuje název kontextu (např. fai-st.utb) a parametr *priznak* obsahuje příznak o typu kontextu. Jedná-li se o kontext pro učitele je v tomto sloupci uloženo slovo „pedagog“. Pokud jde o kontext pro studenty je zde uloženo slovo „student“. Podle zvoleného kontextu se pak při přihlašování rozezná jde-li o studenta či učitele a podle toho se mu zpřístupní jedna nebo druhá část systému.

```
1 CREATE TABLE `kontexty` (
2   `id_kontextu` int(10) unsigned NOT NULL auto_increment,
3   `kontext` char(30) collate utf8_czech_ci NOT NULL,
4   `priznak` char(10) collate utf8_czech_ci NOT NULL,
5   PRIMARY KEY  (`id_kontextu`)
6 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;
```

3.3.4 Tabulka „obor“

Tabulka obor obsahuje seznam oborů a informace o nich. V tabulce je tedy název oboru, jeho zkratka, název studijního programu, pod který tento obor spadá a typ studia. Typ studia pak může nabývat tří různých hodnot. Jedná-li se o bakalářské studium je zde vepsána zkratka BS. Jedná-li se o magisterské studium je zde vepsána zkratka MS. Pokud je obor určen pro oba typy studia je zde vepsána kombinace předchozích zkratek a to ve tvaru BS|MS. Tento tvar zkratky je zvolen záměrně, protože může být jednoduše rozdělen do pole pomocí PHP funkce explode.

```
1 CREATE TABLE `obor` (
2   `id_obor` int(10) unsigned NOT NULL auto_increment,
3   `nazev` text collate utf8_czech_ci NOT NULL,
4   `zkratka` char(30) collate utf8_czech_ci NOT NULL,
```

```
5   `program` text collate utf8_czech_ci NOT NULL,
6   `typ` char(30) collate utf8_czech_ci NOT NULL,
7   PRIMARY KEY  (`id_obor`)
8 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;
```

3.3.5 Tabulka „ucitele“

Tato tabulka obsahuje seznam informací o učitelích. Účel většiny sloupců je dobře vystižen jejich názvy. Problém pak může nastat u sloupce kontakt, který obsahuje číselnou hodnotu udávající ID učitele, kterého si daný učitel zvolil za zástupce (např. v době svojí nepřítomnosti). Tento způsob pak umožňuje to, aby si jednoho učitele zvolilo za zástupce více lidí. Poslední dva sloupce tabulky nazvané *clen_OR* a *predseda_OR* obsahují zkratky oborů jejichž oborových rad je daný učitel členem oddělených znakem |.

```
1 CREATE TABLE `ucitele` (
2   `id_ucitel` int(10) unsigned NOT NULL auto_increment,
3   `novell_login` char(30) collate utf8_czech_ci NOT NULL,
4   `titul_pred` char(30) collate utf8_czech_ci default NULL,
5   `jmeno` char(30) collate utf8_czech_ci NOT NULL,
6   `prijmeni` char(30) collate utf8_czech_ci NOT NULL,
7   `titul_za` char(30) collate utf8_czech_ci default NULL,
8   `ustav` char(30) collate utf8_czech_ci NOT NULL,
9   `mistnost` text collate utf8_czech_ci,
10  `email` text collate utf8_czech_ci NOT NULL,
11  `telefon` char(30) collate utf8_czech_ci NOT NULL,
12  `kontakt` int(10) unsigned NOT NULL,
13  `clen_OR` text collate utf8_czech_ci,
14  `predseda_OR` text collate utf8_czech_ci,
15  PRIMARY KEY  (`id_ucitel`)
16 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;
```

3.3.6 Tabulka „ustav“

Tato tabulka obsahuje seznam ústavů. Každý ústav je vždy definován plným názvem ústavu a jeho zkratkou. Tabulka ústavů plní spíše podružnou funkci, ale její ID je zapsáno v tabulce učitelů a také se využívá při tvorbě ročníkových statistik.

```
1 CREATE TABLE `ustav` (
2   `id_ustav` int(10) unsigned NOT NULL auto_increment,
3   `nazev` text collate utf8_czech_ci NOT NULL,
4   `zkratka` char(30) collate utf8_czech_ci NOT NULL,
```

```
5   PRIMARY KEY  (`id_ustav`)
6 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;
```

3.3.7 Tabulka „diskuse_*“

Tato tabulka obsahuje příspěvky do diskuzí k jednotlivým pracím. Aby se dali jednotlivé příspěvky dobře rozlišit obsahuje tato tabulka kromě textu samotného příspěvku (sloupec *text*) i identifikační číslo učitele (sloupec *id_ucitel*), který je jeho autorem a identifikační číslo práce (sloupec *id_prace*), ke kterému byl příspěvek napsán. Posledním sloupcem je pak *datum*, který obsahuje informaci o času a datu zaslání příspěvku.

```
1 CREATE TABLE `diskuse_*` (
2   `id_diskuse` int(10) unsigned NOT NULL auto_increment,
3   `id_prace` int(10) NOT NULL,
4   `id_ucitel` int(10) NOT NULL,
5   `text` text collate utf8_czech_ci,
6   `datum` text collate utf8_czech_ci,
7   PRIMARY KEY  (`id_diskuse`)
8 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;
```

3.3.8 Tabulka „hlasování_*“

Tato tabulka obsahuje informace k hlasování o schválení či zamítnutí práce. V tabulce je vždy zapsáno identifikační číslo práce (sloupec *id_prace*), které se hlasování týká, identifikační číslo člena oborové rady (sloupec *id_ucitel*), který hlasoval a hlas samotný. Pokud člen OR navrhl práci ke schválení, je ve sloupci hlas vepsáno číslo 1. Pokud naopak navrhl práci k zamítnutí je zde číslo -1. Pokud učitel ještě nehlasoval, ale přispěl do diskuze je pro něj vytvořen řádek s nulovou hodnotou hlasu. Řádky v tabulce jsou vytvářeny až v případě aktivity člena OR a nejsou tedy generovány automaticky dle aktuálního nastavení prací a členů OR. Tímto způsobem je zajištěno, že tabulky se zbytečně nebude plnit v podstatě informačně bezcennými řádky s hodnotu hlasu 0.

```
1 CREATE TABLE `hlasovani_*` (
2   `id_hlas` int(10) unsigned NOT NULL auto_increment,
3   `id_prace` int(10) unsigned NOT NULL,
4   `id_ucitel` int(10) unsigned NOT NULL,
5   `hlas` tinyint(4) default '0',
6   PRIMARY KEY  (`id_hlas`)
7 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;
```

3.3.9 Tabulka „pocty_*“

Tato tabulka obsahuje informace o počtech studentů jednotlivých oborů a zákazech zápisu víceoborových prací. V každém řádku tabulky je vždy zapsáno identifikační číslo oboru (sloupec *id_obor*), samotný počet studentů (sloupec *pocet_studentu*) a informace o zákazu víceoborových prací (sloupec *zakaz*). Počet studentů je vždy ve tvaru „číslo|číslo“ (např. 20|20), kde levé číslo je počet studentů bakalářského studia a pravé počet studentů magisterského studia. Pokud je obor určen např. pouze pro bakalářské studium je na místě parametru určujícího počet studentů magisterského studia nulový a naopak. Sloupec *zakaz* obsahuje hodnotu ve stejném tvaru jako sloupec s počtem studentů Rozdíl je v tom, že číselná hodnota může nabývat pouze hodnot 0 a 1 (tedy např. 0|1), kde nula znamená dostatečný počet víceoborových prací a tedy povolení zápisu na tyto práce studenty oboru jehož se řádek týká a jednička zákaz zápisu na tyto práce. Pomocí tohoto zákazu se lze vyvarovat těch situací, kdy by na studenty jednoho z oborů už nevyšly práce. Fungování algoritmu pro nastavení těchto zákazů je popsán dále v textu.

```
1 CREATE TABLE `pocty_*` (
2   `id_pocty` int(10) unsigned NOT NULL auto_increment,
3   `id_obor` int(10) unsigned NOT NULL,
4   `pocet_studentu` char(30) collate utf8_czech_ci NOT NULL,
5   `zakaz` char(30) collate utf8_czech_ci NOT NULL default '0|0',
6   PRIMARY KEY  (`id_pocty`)
7 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;
```

3.3.10 Tabulka „prace_*“

Tato tabulka obsahuje informace vztahující se k pracím. V každém řádku je kromě identifikačního čísla, které ostatně obsahuje každá tabulka, uveden také plný název práce (sloupec *nazev*), anotace práce (sloupec *anotace*), typ práce (sloupec *typ_prace*), kde zkratka BP znamená bakalářskou práci a DP práci diplomovou, dále zkratka oboru nebo oborů práce (sloupec *obor*), identifikační číslo učitele, který práci vypsal (sloupec *id_ucitel*), identifikační číslo studenta (sloupec *id_student*), který je na práci zapsán a informace o schválení předsedou oborové rady (sloupec *schvaleno*) a obhájení práce studentem (sloupec *obhajeno*).

Pokud je práce vypsána pro více oborů jsou jednotlivé zkratky těchto oborů odděleny znakem |. Sloupec *schvaleno* obsahuje číselnou hodnotu vyjadřující status práce v procesu

schválení. Tato hodnota může být buď 0, je-li práce neschválená ani nezamítnutá, nebo -1, je-li práce zamítnutá, případně 1, je-li práce schválena. Je-li práce určena pro více oborů zaráz jsou statusy těchto prací odděleny znakem | a pořadí hodnot odpovídá pořadí zkratek oborů ve sloupci *obor*.

```
1 CREATE TABLE `prace_*` (
2   `id_prace` int(10) unsigned NOT NULL auto_increment,
3   `nazev` text collate utf8_czech_ci NOT NULL,
4   `anotace` text collate utf8_czech_ci NOT NULL,
5   `typ_prace` char(30) collate utf8_czech_ci NOT NULL,
6   `obor` text collate utf8_czech_ci NOT NULL,
7   `id_ucitel` int(10) unsigned NOT NULL,
8   `id_student` int(10) unsigned NOT NULL default '0',
9   `schvaleno` text collate utf8_czech_ci NOT NULL,
10  `obhajeno` tinyint(4) default '0',
11  PRIMARY KEY  (`id_prace`)
12 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;
```

3.3.11 Tabulka „studenti_*“

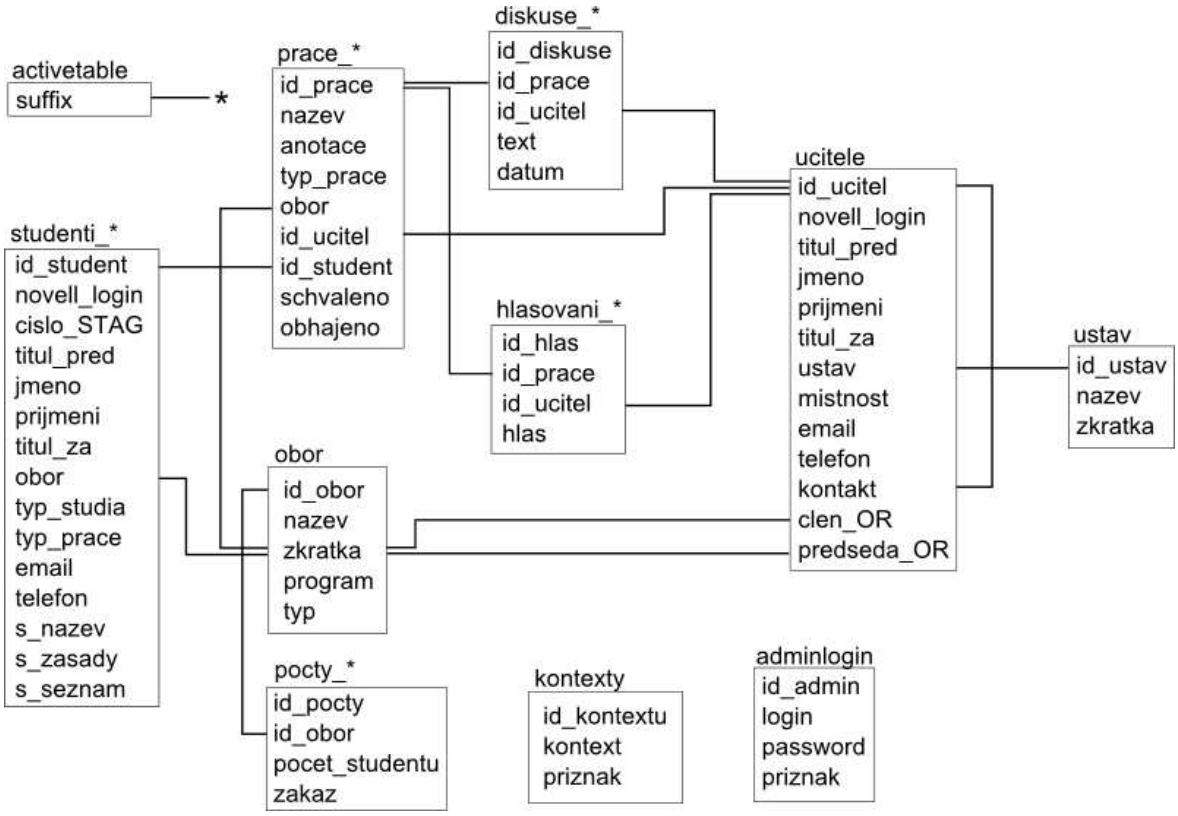
Tato tabulka obsahuje informace o studentech pro rok daný sufixem názvu tabulky. Každý řádek pak obsahuje losovací jméno studenta do školní sítě Novell (sloupec *novell_login*), studentovo číslo ze systému STAG (sloupec *cislo_STAG*), celé jméno studenta včetně titulů (sloupce *titul_pred, jmeno, prijmeni a titul_za*), zkratku oboru studia (sloupec *obor*), typ studia (sloupec *typ_studia*), který může nabývat hodnoty PS pro prezenční studiu a KS pro studium kombinované, dále typ práce (sloupec *typ_prace*), nabývající hodnot BP nebo DP – tedy bakalářská práce nebo diplomová práce, a informace s kontaktními údaji na studenta (sloupce *email* a *telefon*). Poslední tři sloupce se týkají podkladu pro zadání bakalářské/diplomové práce a obsahují název práce (sloupec *s_nazev*), zásady pro vypracování (sloupec *s_zasady*) a seznam literatury (sloupec *s_seznam*).

```
1 CREATE TABLE `studenti_*` (
2   `id_student` int(10) unsigned NOT NULL auto_increment,
3   `novell_login` char(30) collate utf8_czech_ci NOT NULL,
4   `cislo_STAG` char(30) collate utf8_czech_ci NOT NULL,
5   `titul_pred` char(30) collate utf8_czech_ci default NULL,
6   `jmeno` char(30) collate utf8_czech_ci NOT NULL,
7   `prijmeni` char(30) collate utf8_czech_ci NOT NULL,
8   `titul_za` char(30) collate utf8_czech_ci default NULL,
```

```
 9   `obor` text collate utf8_czech_ci NOT NULL,
10   `typ_studia` text collate utf8_czech_ci NOT NULL,
11   `typ_prace` text collate utf8_czech_ci NOT NULL,
12   `email` text collate utf8_czech_ci NOT NULL,
13   `telefon` char(30) collate utf8_czech_ci NOT NULL,
14   `s_nazev` text collate utf8_czech_ci,
15   `s_zasady` text collate utf8_czech_ci,
16   `s_seznam` text collate utf8_czech_ci,
17   PRIMARY KEY  (`id_student`)
18 ) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;
```

3.3.12 Relační model

Z předchozího popisu tabulek je zřejmé, že většina tabulek je spolu jistým způsobem propojena. Pro lepší představu tohoto propojení slouží následující model zobrazující všechny tabulky a propojení mezi nimi. Propojení jednotlivých sloupců tabulek je zde vyobrazeno pomocí čar. Výjimkou je spojení tabulky *activetable*, které vede k symbolu hvězdičky. Toto spojení představuje propojení na sufix tabulek, které se každý rok obměňují.



Obrázek 20 - Relační model databáze

3.4 Použité třídy

Jelikož v PHP5 je již plná podpora objektově orientovaného programování, byl by škoda této možnosti nevyužít. Pro to byly speciálně pro tuto aplikaci byly vytvořeny dvě třídy. První s názvem cMySQL obsahuje několik základních metod pro práci s databází. Druhá třída nazvaná cAvatar se stará o upload učitelských ikon.

3.4.1 Třída cMySQL

Tato třída obsahuje metody pro vytvoření spojení s databází, provedení dotazu, vrácení počtu řádků tabulky a dvě metody pro export databáze.

Konstruktor třídy

Konstruktor této třídy obsahuje pouze inicializaci proměnných nezbytných pro připojení k MySQL serveru.

```
1 public function __construct($MYSQL_SERVER, $MYSQL_USER,
2 $MYSQL_PASSWORD, $MYSQL_DATABASE) {
3         $this->MYSQL_SERVER = $MYSQL_SERVER;
4         $this->MYSQL_USER = $MYSQL_USER;
5         $this->MYSQL_PASSWORD = $MYSQL_PASSWORD;
6         $this->MYSQL_DATABASE = $MYSQL_DATABASE;
7 }
```

Metoda connect

Tato metoda slouží k připojení k databázovému serveru. Při volání této metody se nejprve vytvoří spojení s MySQL serverem (řádky 3 až 6) a poté je volána funkce pro nastavení databáze (řádky 7 a 8). Za povšimnutí také stojí řádky 9 a 10, které se starají o konverzi znakové sady. Absence těchto dvou řádků by způsobila nesprávné zobrazovaní některých znaků v okně prohlížeče. Metoda nemá žádné vstupní parametry, protože využívá parametrů nastavených v konstruktoru, ale vrací identifikátor spojení s databází a booleovskou proměnnou informující o úspěchu či neúspěchu vytvoření spojení s databází.

```
1 public function connect() {
2   ($this->link = mysql_connect( $this->MYSQL_SERVER,
3                                 $this->MYSQL_USER,
4                                 $this->MYSQL_PASSWORD)
5                               ) or $this->printError();
```

```
6    ($db = mysql_select_db( $this->MYSQL_DATABASE, $this->link)
7                           ) or $this->printError();
8    mysql_query("SET NAMES 'cp1250';");
9    mysql_query("SET CHARACTER_SET 'cp1250';");
10   return ($this->link and $db);
11 }
```

Metody query a numRows

Tyto dvě metody se starají o provádění dvou nejčastějších příkazů. První metoda nazvaná *query* provádí prosté volání PHP funkce mysql_query(), která posílá MySQL dotaz. Metoda má pouze jeden vstupní parametr a tím je již zmiňovaný dotaz. Návratovou hodnotou je identifikátor výsledku.

```
1 public function query($query) {
2   $sr = mysql_query($query, $this->link);
3   return($sr);
4 }
```

Druhá funkce nazvaná *numRows* navíc volá ještě PHP funkci mysql_num_rows(), která vrací počet řádků ve výsledku metody *query*, který je zároveň výstupem metody. I zde je jediným parametrem prováděný dotaz.

```
1 public function numRows($query) {
2   $i = $this->query($query);
3   $num_rows = mysql_num_rows($i);
4   return ($num_rows);
5 }
```

Metody GetTableDef a GetTableContent

Tyto dvě metody se používají při zálohování databáze. Obě metody mají dva vstupní parametry, kde první je název tabulky a druhý je znak zalomení řádku. Návratovou hodnotou metody *GetTableDef* je příkaz pro smazání a znovuvytvoření zadané tabulky. Metoda *GetTableContent* zase vrací obsah zadané tabulky. Společně tak tyto dvě metody vytvoří sadu příkazu, která se dá použít pro obnovu dat po kritickém výpadku databázového serveru nebo při přesunu aplikace jinam.

Tyto dvě funkce byly převzaty a upraveny z nástroje pro správu databází phpmyadmin, který je šířen pod svobodnou licencí GPL (GNU General Public License).

Příklad použití třídy cMySQL

Pro demonstraci praktického použití této třídy byla vybrána část skriptu pro zálohu databázových tabulek, protože využívá většinu metod této třídy.

Jelikož je soubor s tímto skriptem volán po kliknutí na odkaz a je tedy pouze jakýmsi typem rozšíření je nejprve nutné vložit soubor obsahující deklaraci třídy (cMySQL.php) a soubor s údaji pro připojení k databázi (dtbData.php).

```
1 include "../classes/cMysql.php";
2 include "../includes/dtbData.php";
```

Následuje vytvoření nového objektu třídy s názvem *$trida*. Proměnné uvedené v konstruktoru třídy byly inicializovány vložením souboru dtbData.php.

```
3 $trida = new cMysql($MYSQL_SERVER, $MYSQL_USER,
4                     $MYSQL_PASSWORD, $MYSQL_DATABASE);
```

Na řádku 5 je vidět použití členské funkce *connect*, která vytvoří spojení s databázovým serverem. Na dalším řádku se pak pomocí metody *query* provede dotaz na vypsání všech tabulek z databáze.

```
5 $trida->connect();
6 $result = $trida->query("SHOW TABLES FROM $MYSQL_DATABASE");
```

Zbytek skriptu už je holý export. Nejprve dojde k inicializaci proměnné *$oneBigString* (řádek 7), poté se začne provádět smyčka while, která pro každou jednu tabulku v databázi provede metody *GetTableDef* a *GetTableContent*. Po skončení této smyčky je pak v proměnné *$oneBigString* nachází celý obsah databáze.

```
 7 $oneBigString = "";
 8
 9 While($line = mysql_fetch_array($result))
10 {
11    $table = $line[0];
12    $crlf = "\n";
13
14    $oneBigString .= $trida->GetTableDef($table, $crlf);
15    $oneBigString .= $trida->GetTableContent($table,$crlf);
16    $oneBigString .= "\n\n";
17 }
```

Skript dále pokračuje zápisem do souboru. Pro ukázku funkce třídy je však předchozí část skriptu plně dostačující.

3.4.2 Třída cAvatar

Tato třída obsahuje členské funkce pro hledání souboru a samotný upload souboru ikony na server. Do konstruktoru třídy se nevkládají žádné parametry je tedy možné přejít rovnou k vysvětlení jednotlivých metod.

Metoda FindAvatar

Tato metoda byla vytvořena pro hledání ikon. Metoda má dva vstupní parametry: cestu k adresáři a jméno souboru. Jméno souboru se zadává bez koncovky, protože koncovka souboru může být různá v závislosti od formátu obrázku. Metoda vrací celý název souboru je-li nalezen. V případě kdy soubor nalezen není vrací metoda hodnotu FALSE.

```
1  public function FindAvatar($path, $nameOfFile) {
2     $directoryArray = scandir($path);
3     $numEl = count($directoryArray);
4     for ($i = 0; $i < $numEl; $i++) {
5        if ( ($len = strlen($directoryArray[$i])) > 4) {
6           $name = substr($directoryArray[$i], 0, ($len-4));
7
8           if ($name == $nameOfFile) {
9              $result = $directoryArray[$i];
10             break;
11          }
12          else {
13             $result = FALSE;
14          }
15       }
16    }
17    return $result;
18 }
```

Metoda UploadImage

Tato metoda slouží k uploadu obrázku. Její vstupní parametry jsou jméno souboru, typ souboru, velikost souboru a jméno souboru v adresáři pro dočasné soubory (nejčastěji TMP

nebo TEMP), dále cesta k adresáři kam bude soubor nahrán, nové jméno souboru a příznak pro vypisování hlášek pro ladění skriptu při nefunkčnosti. Metoda nevrací žádnou hodnotu.

```
1  public function UploadImage($IMG_NAME, $IMG_TYPE, $IMG_SIZE,
2                              $IMG_TMP_NAME, $path, $newName,
3                              $beVerbose) {
4     $this->IMG_NAME = $IMG_NAME;
5     $this->IMG_TYPE = $IMG_TYPE;
6     $this->IMG_SIZE = $IMG_SIZE;
7     $this->IMG_TMP_NAME = $IMG_TMP_NAME;
8     //upload do docasneho adresare
9     if ($beVerbose == true) {
10       if (is_uploaded_file($this->IMG_TMP_NAME) == TRUE) {
11          echo "<b>Soubor byl úspěšně nahrán do tmp.</b><br>";
12       }
13       else {
14          echo "<b>Chyba pri nahrávání souboru.</b><br>";
15       }
16    }
17
18    //vytvoreni noveho jmena
19    $delka = strlen($this->IMG_NAME);
20    $koncovka = substr($this->IMG_NAME, ($delka-4), $delka);
21    $fullName = $newName.$koncovka;
22    $destination = $path.$fullName;
23
24    //presunuti a prejmenovani souboru
25    $vysl = move_uploaded_file( $this->IMG_TMP_NAME, $destination);
26
27    if ($beVerbose == TRUE) {
28       if ($vysl == TRUE) {
29          echo "<b>Přesunutí souboru proběhlo v pořádku.</b><br>";
30       }
31    }
32 }
```

Příklad použití třídy cAvatar

Dobrým příkladem použití této metody je část skriptu která se vykoná vždy rozhodne-li se učitel přiřadit ke svému profilu ikonu. Vstupní pole pro upload souboru je součástí formuláře s ostatními údaji o daném učiteli. Při odesílání formuláře se tak společně se

souborem odesílají i jiné údaje. Jedním z těchto údajů je i login do sítě Novell, který slouží jako základ pro nový název souboru.

Nejprve je nutné vložit soubor s třídou a vytvořit nový objekt této třídy. V příkladu se tento objekt jmenuje *$ava*.

```
1 include "../classes/cAvatar.php";
2 $ava = new cAvatar();
```

Následně jsou ze superglobální proměnné $_FILES vyčteny údaje o velikosti, typu a názvech souboru.

```
3 $name = $_FILES["linkIkona"]["name"];
4 $type = $_FILES["linkIkona"]["type"];
5 $size = $_FILES["linkIkona"]["size"];
6 $tmp_name = $_FILES["linkIkona"]["tmp_name"];
```

Nakonec se zavolá členská funkce pro samotný upload.

```
7 $ava->UploadImage($name, $type, $size, $tmp_name, "../avatars/",
8                   $_POST[novell_login], FALSE);
```

3.4.3 Třída FPDF

Jedinou převzatou třídou je třída FPDF pomocí níž je generován výběr zadání tématu bakalářské/diplomové práce ve formátu PDF. Aby se tato třída dala v tomto projektu a v českých podmínkách vůbec použít, bylo nutné vytvořit pro ni sadu fontů s českým kódováním. Samotná knihovna totiž nepodporuje české znaky ani české kódování. Nicméně existuje postup, kterým lze dokumentům češtinu „vnutit“. Principiálně jde o to, že z TrueType fontů se vyberou znaky, které jsou pro dané kódování (v tomto případě cp1250) potřebné a ty se potom přidají k výslednému dokumentu. Nevýhodou je, že dokument se zvětšuje s množstvím použitých fontů. Jedno písmo si pak vezme cca 30kB.

Podrobný postup pro konverzi fontů do různých znakových sad a vysvětlení členských funkcí je popsáno na oficiálních stránkách projektu.

Příklad použití třídy FPDF

Následující příklad je zkrácenou verzí PHP skriptu pro vytvoření dokumentu s výběrem tématu bakalářské/diplomové práce. Uvedení celého skriptu není nutné z toho důvodu, že

funkce, které vkládají jednotlivé řádky se opakují stále ve stejném tvaru a tak by jejich uvedení nemělo téměř žádnou informační hodnotu.

Nejprve je nezbytné definovat cestu k použitým fontům (řádek 1) a vložit soubor s touto třídou (řádek 2). Poté je vytvořen nový objekt této třídy s názvem *$pdf* (řádek 3). Konstruktor třídy vyžaduje definici tří argumentů. Prvním argumentem je orientace stránky, kde P určuje, že stránka bude na výšku. Druhým argumentem je jednotka délky, která se má používat (tedy milimetr), a posledním je formát stránky.

```
1  define('FPDF_FONTPATH','font/');
2  require('../../../classes/fpdf.php');
3  $pdf = new FPDF("P","mm","A4");
```

Nyní je třeba zaregistrovat fonty, které se budou používat. Definice fontu vždy obsahuje rodinu písem (Font family), řez písma a název souboru s definicí. Pokud je řez písma prázdný volí se přednastavená hodnota regular.

```
4  //registrace fontu
5  $pdf->AddFont('times','','times.php');
6  $pdf->AddFont('timesi','','timesi.php');
7  $pdf->AddFont('timesbd','','timesbd.php');
```

Následuje vložení první a v tomto případě jediné stránky.

```
8  //Nova stranka
9  $pdf->AddPage();
```

Nyní je potřeba vložit logo univerzity. Kromě cesty k obrázku je argumentem ještě umístění vzhledem k levému hornímu rohu stránky a šířka obrázku. Délka obrázku se dopočítá automaticky.

```
10  //vlozeni loga univerzity
11  $pdf->Image('../../../img/logo.png',27,27,66);
```

Dalším krokem je tisk samotného textu. Nejdříve se stanoví typ písma řez a velikost pomocí členské funkce SetFont. Nastavený font se pak bude používat dokud nebude nastaven jiný. Poté je třeba zadat umístění řádku s písmem vzhledem k levému hornímu okraji. Pro vložení bloku textu se používají metody Cell (pro jednořádkový text) a MultiCell (pro více řádků). Jejichž argumenty jsou šířka a výška řádku, vypisovaný text,

šířka ohraničení a zarovnání textu. U metody Cell je předposledním argumentem mezera před vypsáním řádku.

```
12  $pdf->SetFont('timesbd','',18);
13  $pdf->SetXY(28, 45);
14  $pdf->Cell(156, 0, $nadpis, 0, 1, 'C');
15
16  $pdf->SetXY(60, 59.5);
17  $pdf->MultiCell(123, 5, $line[nazev], 0, 'L');
```

Posledním krokem je výstup vytvořeného PDF souboru do prohlížeče. Pokud prohlížeč obsahuje plugin pro zobrazování souborů je vygenerovaný soubor zobrazen. V opačném případě je nabídnut ke stažení do počítače.

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

3.5 Vybrané skripty

Vzhledem k rozsáhlosti této aplikace zde není možné uvést popis každého PHP skriptu, a bylo by to i zbytečné. Většinou totiž obsahuje běžné funkce, které se dají dohledat v manuálu a účel posloupnosti těchto funkcí a celého skriptu je na první pohled zřejmý. Následující podkapitoly se tak budou zabývat pouze těmi skripty, které jsou jistým způsobem neobvyklé nebo zajímavé.

3.5.1 Autorizace pomocí LDAP a její využití při přihlášení

Běžný uživatel při přihlašování do systému vyplňuje trojici informací: uživatelské jméno, heslo a kontext (vše stejné jako při běžném přihlášení do sítě Novell). Tyto údaje slouží k autorizaci uživatele a to pomocí informace získané po připojení a bindnutí k LDAP (Lightweight Directory Access Protocol) školní sítě Novell. Pokud bind proběhne v pořádku vrátí se hodnota TRUE a přihlašovací proces normálně pokračuje. Pokud se vrátí hodnota FALSE, znamená to, že uživatel s těmito údaji v síti Novell neexistuje a systém ho nepustí dál nepustí. Funkce pro připojení a bind k LDAP vypadá následovně.

```
1 function LDAP_auth($LDAP,$username,$password) {
2    $ds = LDAP_connect($LDAP["server"], $LDAP["port"]);
3
4    if ($ds) {
5       if (@LDAP_bind($ds, $username, $password)) {
6          return(true);
```

```
7        }
8        else {
9           LDAP_close($ds);
10          return(false);
11       }
12    }
13    else {
14       echo("LDAP's down.");
15       return(false);
16    }
17 }
```

Aby tato funkce správně fungovala, je ještě před jejím voláním nezbytné nastavit kromě adresy Novell serveru (např. nw-central.utb.cz) a portu (389) také správný tvar přihlašovacího jména. Toto přihlašovací jméno musí obsahovat kromě samotného uživatelského jména také organizaci a organizační jednotku. Tyto dva údaje se získají rozdělením zadaného kontextu a plní funkci jakési cesty k uživatelskému účtu.

Například pokud uživatel zadá jméno *m_mrkev*, heslo *heslo* a kontext *fai-st.utb*, přihlašovací jméno se zasílá ve tvaru *cn=m_mrkev,ou=fai-st,o=utb*. Samotné volání funkce pak může vypadat například takto:

```
1 $LDAP["server"] = "nw-central.utb.cz";
2 $LDAP["port"] = "389";
3 LDAP_auth($LDAP, "cn=m_mrkev,ou=fai-st,o=utb", "heslo");
```

3.5.2 Konverze podkladu pro zadání práce do formátu pro sázecí systém TeX

Protože formulář pro vytváření podkladu pro zadání bakalářské/diplomové práce v systému STAG používá značky sázecího systému TeX, bylo nutné vytvořit funkci, která do obyčejného textu tyto značky přidá. Konkrétně se jedná o znak zalomení řádku a znaky pro číslování arabskými číslicemi. Funkce má jako vstupní parametr konvertovaný text. Funkce nevrací žádnou hodnotu, ale v průběhu skriptu postupně vypisuje konvertované řádky. Konkrétní skript je pak popsán v následujících řádcích.

Funkce nejdříve celý text rozkouskuje na řádky a tyto pak přiřadí do pole.

```
1 function convertTeX ($text) {
2    $zacatekCislovani = true;
3    $pole = explode("\n", $text);
```

Následuje dvojice for cyklů. První (vnější – řádek 4) cyklus postupně prochází celé pole. Druhý (vnitřní – řádek 5) cyklus prochází vždy první čtyři znaky řetězce a podmínka v něm testuje, zda-li jsou tyto znaky číslo nebo mezera (řádek 10). Nastane-li případ kdy podmínka není splněna je vnitřní cyklus přerušen pomocí příkazu break. Pokud podmínka splněna je, přičte se jednička k proměnné *$index* (řádek 14). Proměnná *$index* obsahuje informaci o počtu znaků, které se odstraní ze začátku řádku (řádek 17). Vzhledem k tomu jak je postavená podmínka (řádek 10), jedná se o znaky mezer nebo čísel.

```
4    for ($i = 0; $i < count($pole); $i++ ) {
5       for ($j = 0, $index = 0; $j < 4; $j++ ) {
6          // vrati j-té pismeno z retezce
7          $znak = $pole[$i]{$j};
8
9          //pokud není znak cislo nebo mezera tak break
10         if (is_numeric($znak) != true && ord($znak) != 32) {
11            break;
12         }
13         else {
14            $index++;
15         }
16      }
17      $text = substr($pole[$i], $index);
```

Je-li nyní na začátku řádku znak tečky, znamená to, že se bude číslovat. Pokud je navíc příznak začátku číslování nastaven na TRUE je ještě vložen příkaz pro začátek číslování (řádky 19-22). Potom se tato tečka odebere ze začátku řádku a místo ní se zde dá příkaz pro položku číslování „\item{}“.

```
18      if ( $text{0} == "." ) {
19         if ( $zacatekCislovani == true ) {
20            echo "\begin{arab} <br>";
21            $zacatekCislovani = false;
22         }
23
24         //odebrani tecky ze zacatku a vlozeni TeX znacky
25         $text = substr($text, 1);
26         echo $text = "\item{}".$text."<br>";
27      }
```

Pokud na začátku řádku není znak tečky znamená to, že jde o normální řádek a tomuto se pouze na konec přidá příkaz pro zalomení řádku (řádek 33). Nejprve je ovšem zjištěno jestli se zrovna neprovádí číslování. V tom případě je nejprve nutné vložit příkaz pro konec číslování (řádky 29-32) .

```
28        else {
29           if ( $zacatekCislovani == false ) {
30              echo "\end{arab} <br>";
31              $zacatekCislovani = true;
32           }
33           echo $text."\\nl{} <br>";
34        }
35     }
```

Aby nedošlo k situaci kdy po skončení vnějšího cyklu for (řádku 35) není ukončeno číslování je ještě jednou otestován příznak vložení příkazu pro začátek číslování a pokud číslování běží je vložen příkaz pro konec.

```
36     if ( $zacatekCislovani == false ) {
37        echo "\end{arab} <br>";
38        $zacatekCislovani = true;
39     }
40  }
```

3.5.3 Kontrola dostatečného počtu prací pro jednotlivé obory

Asi největším oříškem celé této aplikace byl skript, který hlídá dostatečný počet prací určených pro více oborů a při nedostatečném počtu prací nastaví zákaz. Pro lepší pochopení smyslu tohoto skriptu je vhodné uvést příklad.

Na fiktivní škole existují dva obory X a Y. Obor X studuje 30 studentů a má 30 jednooborových prací. Obor Y studuje také 30 studentů, ale počet jednooborových prací je 25. Navíc existuje 10 prací, které si může zapsat jak student oboru X, tak student oboru Y. Shrnutí viz tabulka 2.

Tabulka 2 - Shrnutí příkladu

Obor	Počet studentů	Počet jednooborových prací	Počet společných prací
X	30	30	10
Y	30	25	

Pokud si více jak 5 studentů z oboru X zvolí práce, které jsou společné pro oba obory, nastane situace, kdy pro studenty oboru Y nebude již dostatečný počet prací a pár z nich tak jednoduše „ostrouhá“. Aby k těmto situacím nemohlo dojít je zde právě tento skript.

Následující akce se provádí pokaždé, když je na práci zapisován student. Samotný skript zde nebude kvůli své délce uveden, ale je k nahlédnutí na přiloženém optickém médiu. V tomto případě si student oboru Informační technologie bude chtít zapsat víceoborovou diplomovou práci vypsanou pro jeho obor a navíc ještě pro obor Automatické řízení a informatika (dále jen AŘI). Celý proces pak vypadá následovně:

- nejprve se zkontroluje zda již není student někde zapsán
- poté se ověří zda obor práce odpovídá oboru studenta
- následuje zjištění jestli je práce jednooborová. Pokud by byla skript se ukončí a student se normálně zapíše na práci. Pro tento příklad to však neplatí.
- nyní se z tabulky počtu studentů zjistí jestli je pro obor studenta a typ práce nastaven zákaz. Pokud by byl zákaz nastaven skript se ukončí a vyskočí upozornění na tento fakt. V tomto případě však zákaz nastaven není.
- Nyní je třeba zjistit jestli je pro obor Automatické řízení a informatika ještě dost víceoborových diplomových prací. Takže se načte počet studentů AŘI a od něj se odečte počet jednooborových diplomových prací pro AŘI. Dále se načte počet víceoborových diplomových prací pro tento obor, kde už je nějaký student tohoto oboru zapsán. Ten se potom odečte od výsledku minulého rozdílu. Výsledkem je číslo určující nutný počet víceoborových diplomových prací.
- Nyní je třeba zjistit kolik víceoborových diplomových prací pro AŘI ještě zbývá.
- Pokud je počet volných víceoborových diplomových prací větší nebo roven nutnému počtu víceoborových diplomových prací může se student informačních technologií na vybranou práci zapsat. V opačném případě je vypsána hláška upozorňující na tento fakt a do sloupce zákaz se k oboru Informační technologie zapíše příznak zákazu zápisu víceobororových diplomových prací.

3.6 Manuál

Poslední částí této diplomové práce bylo zpracování manuálu k nově vytvořené aplikaci. Tento manuál je logicky rozdělen na tři části a to na:

- administrátorskou část
- část určenou pro učitele
- část určenou studentům

Každá část obsahuje podrobný popis funkcí typických pro jednotlivá grafická rozhraní. Administrátorská část navíc obsahuje návod k instalaci systému při nutnosti přechodu na jiný webhosting.

ZÁVĚR

Účelem této diplomové práce bylo vytvoření aplikace pro správu bakalářských a diplomových prací. Uživatelské rozhraní bylo vytvořeno pomocí technologií HTML a CSS, funkčnost systému pak zajištěna pomocí PHP a JavaScriptu. Data zpracovaná systémem jsou uložena v MySQL databázi, která je dostupná na školním serveru.

Hlavní funkcí této aplikace je zpříjemnit a zjednodušit celou administraci kolem bakalářských a diplomových prací a to jak učitelům, tak studentům. Nově vytvořený systém obsahuje celou řadu funkcí. Zjednodušeně lze však říci, že systém umožňuje učitelům práce vypisovat, členům oborových rad o nich hlasovat, předsedům tyto práce schvalovat či zamítat a následně učitelům studenty na schválené práce zapisovat.

Tyto čtyři základní funkce jsou však doplněny o mnoho dalších. Jako příklad lze uvést funkci generující statistické údaje o vypsaných pracích, funkci pro export obhájených prací do formátu CSV (formát MS Excel) anebo funkci pro zálohování databáze a její obnovu ze zálohy. Praktické je také využití freeware knihovny FPDF, která umožňuje aplikaci generovat PDF soubory s výběrem tématu práce a to nezávisle na nainstalovaných PHP knihovnách. Dalším užitečným nástrojem je funkce pro export podkladu pro zadání práce do formátu pro sázecí systém TeX. Až do teď bylo totiž nutné doplňovat do systému STAG speciální znaky ručně až po vyplnění podkladu studentem. Takováto práce byla zdlouhavá a únavná.

Vzhledem k tomu, že aplikaci nebylo možné vyzkoušet za ostrého provozu je možné, že obsahuje některé slabiny, které se projeví teprve při jejím nasazení na začátku příštího školního roku. Při zkušebním provozu se však tato aplikace jevila jako funkční.

ZÁVĚR V ANGLIČTINĚ

The aim of this diploma thesis was creating an application for administration of bachelor and diploma theses. The interface was created by means of the HTML and CSS technologies. The system utility was ensured by PHP and JavaScript. The data processed by the system is saved in MySQL database, which is available on the school server.

The main function of this application is to simplify the whole administration of bachelor and diploma theses, for both: teachers and the students. The newly created system includes a variety of functions. In short, the system enables teachers to print the theses, members of the branch councils to vote on the theses, chairmen to approve or deny them and, consequently, teachers to announce their approval to students.

The four basic functions described above were enhanced by many others. For example, the program offers a function of generating statistical data of theses, a function of exporting the approved theses into CSV format (an MS Excel format), or a database backup function and its restoration from the backup. A good and practical aspect is the utilization of the freeware FPDF library, which enables the application to generate PDF files with selection of theses topics independently on installed PHP libraries. Another useful tool is function for exporting the basis for submission of the thesis into the format for the typesetting system TeX. So far, it has been necessary to input atypical symbols into the STAG system manually after students had submitted them. Such work was longsome and impractical.

Considering the fact that the application couldn’t be widely tested, it is possible, that it contains minor weaknesses, which will be discovered after its implementation at the beginning of the next school year. In an experiment, however, the application proved to be functional.

SEZNAM POUŽITÉ LITERATURY

[1] SCHLOSSNAGLE, George. *Pokročilé programování v PHP5*. Zoner Press: Brno, 2004.

[2] KMENT, Vojtěch. *Hašovací funkce: Jak se odolává hackerům* [online]. [cit. 2007-05-20]. Dostupný z WWW: <http://www.lupa.cz/clanky/hasovaci-funkce-jak-se-odolava-hackerum/>.

[3] WIKIPEDIE. *MD5* [online]. [cit. 2007-05-20]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/MD5>.

[4] WIKIPEDIE. *PHP* [online]. [cit. 2007-05-20]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/PHP>.

[5] WIKIPEDIE. *MySQL* [online]. [cit. 2007-05-21]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/MySQL>.

[6] WIKIPEDIE. *HTML* [online]. [cit. 2007-05-19]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/HTML>.

[7] WIKIPEDIE. *Secure Sockets Layer* [online]. [cit. 2007-05-03]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/SSL>.

[8] WIKIPEDIE. *HTTPS* [online]. [cit. 2007-05-04]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/HTTPS>.

[9] WIKIPEDIE. *JavaScript* [online]. [cit. 2007-05-12]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/JavaScript>.

[10] WIKIPEDIE. *Cascading Style Sheets* [online]. [cit. 2007-05-12]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Cascading_Style_Sheets>.

[11] ZAJÍC, Petr. *PHP(1) – Historie a budoucnost* [online]. [cit. 2007-05-20]. Dostupný z WWW: <http://www.linuxsoft.cz/php/>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

HTML	Hypertext Markup Language
PHP	Hypertext Preprocessor
CSS	Cascading Style Sheets
PDF	Portable Document Format
FTP	File Transfer Protocol
MD5	Message Digest 5
LDAP	Lightweight Directory Access Protocol
SSL	Secure Socket Layer
SGML	Standard Generalized Markup Language
XML	Extensible Markup Language
UTF	Universal Transformation Format
GPL	General Public Licence
OR	Oborová rada

SEZNAM OBRÁZKŮ

SEZNAM TABULEK