VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

REDAKČNÍ SYSTÉM

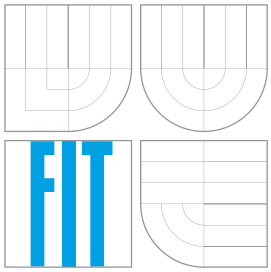
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

LUKÁŠ POHL

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

REDAKČNÍ SYSTÉM

CONTENT MANAGEMENT SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

LUKÁŠ POHL

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ŠÁRKA KVĚTOŇOVÁ

BRNO 2008

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Pohl Lukáš**
Obor: Informační technologie
Téma: **Redakční systém**
Kategorie: Web

Pokyny:

1. Seznamte se s problematikou redakčních systémů.
2. Seznamte se s operačním systémem Fedora Core 4 a technologiemi mySQL, HTML a PHP.
3. Navrhněte webovou aplikaci, která bude sloužit ke sdílení a předávání informací na firemním intranetu. Hlavní důraz kladťe na celkovou jednoduchost, přehlednost a podřízenost funkci.
4. Zrealizujte funkční prototyp redakčního systému.
5. Diskutujte další (neimplementované) možnosti použité technologie a zhodnoťte dosažený výsledek.

Literatura:

- Williams, H. E., Lane, D.: PHP a MySQL - Vytváříme webové databázové aplikace. Computer Press, 2002, 552 s. ISBN 8072267604
- Kosek, J.: HTML, tvorba dokonalých www stránek. Praha: Grada Publishing, 1998, 291 s. ISBN 80-7169-608-0
- DeLisle, M.: PHPMyAdmin - efektivní správa MySQL. Brno: Zoner Press, 270 s. ISBN 8086815099
- Ullman, L.: PHP a MySQL. Computer Press, 2004, 536 s. ISBN 8025100634

Při obhajobě semestrální části projektu je požadováno:

- Body 1 - 3.

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

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

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

Vedoucí: **Květoňová Šárka, Ing.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2007
Datum odevzdání: 14. května 2008

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

doc. Ing. Jaroslav Zendulka, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Lukáš Pohl**
Id studenta: 78794
Bytem: Nad Císařskou 170, 541 01 Trutnov
Narozen: 10. 03. 1986, Trutnov
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   bakalářská práce

Název VŠKP: Redakční systém
Vedoucí/školitel VŠKP: Květoňová Šárka, Ing.
Ústav: Ústav informačních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

.....	.....
Nabyvatel	Autor

Abstrakt

Cílem této práce je vytvořit webovou aplikaci (redakční systém), která slouží ke zlepšení firemní komunikace, sdílení a předávání informací prostřednictvím firemního intranetu. Největší důraz je kladen na samotné uživatelské rozhraní, které by mělo být jednoduché, logické a snadno pochopitelné pro běžného uživatele. Je také vyžadována funkčnost ve třech nejpoužívanějších prohlížečích (Internet Explorer, Firefox, Opera). Použité technologie jsou PHP(OOP), JavaScript, AJAX, HTML, MySQL a webový server Apache2, který běží pod operačním systémem Fedora Core.

Klíčová slova

Redakční systém, Javascript, AJAX, PHP, MySQL, Apache

Abstract

The aim of this work is to create a web application which is used for improving company communication, for sharing and transfer of information via the company intranet (network). Main focus is placed on the user interface itself, which should be simple, logical and user friendly for a regular user. Utility in the three most frequently used browsers (Internet Explorer, Firefox and Opera) is also required. The technologies used are PHP (OOP), JavaScript, AJAX, HTML, MySQL and web server Apache2, under the Fedora Core operation system.

Keywords

Content management systems, CMS, Javascript, AJAX, PHP, MySQL, Apache

Citace

Lukáš Pohl: Redakční systém, bakalářská práce, Brno, FIT VUT v Brně, 2008

Redakční systém

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením paní ing. Šárky Květoňové.

.....
Lukáš Pohl
5. května 2008

Poděkování

Touto cestou bych chtěl poděkovat zástupci z firmy DELONG INSTRUMENTS a.s. panu ing. Petru Mezuláníkovi za jeho profesionální a trpělivý přístup při vývoji celého systému.

Obsah

Kapitola 1

Úvod

Tato práce se skládá z několika základních kapitol, ve kterých se budu věnovat problematice redakčních systémů.

Hned po této úvodní kapitole se ve druhé kapitole pokusím nastínit základní funkce redakčních systémů. Vysvětlím zde také tento pojem, jak a z čeho vznikl a pokusím se popsat hlavní uživatelské role v těchto systémech z hlediská práv a ovládání. Zmíním se o architektuře, na které tyto systémy pracují a co je potřeba pro jejich provoz a údržbu (APACHE, IIS,...). Věnovat se také budu pojmům a technologiím se kterými tyto systémy nejčastěji spolupracují. Popíši nejpoužívanější serverové skriptovací technologie (PHP, ASP) a také klientské skriptovací technologie (JavaScript, AJAX) a pojmy se kterými tyto jazyky úzce souvisí (XML, DOM, DHTML ...). V závěru této kapitoly popíši nejpoužívanější databázové technologie (MySQL, PostgreSQL, Oracle).

Ve třetí kapitole se již budu věnovat návrhu, řešení a implementaci vlastního systému. Uvedu zde použité skriptovací technologie, databázovou technologii, operační systém a prostředí, která jsem zvolil pro vývoj. Uvedu ukázku analýzy a postupu při návrhu databáze včetně SQL skriptu. Velký důraz je také kladen na návrh uživatelského rozhraní, kterému se věnuji v následujících podkapitolách. Z praktického hlediska popíši postup při řešení přihlašování uživatelů, zabezpečení a uvedu příklady nastavení adresářů se zdrojovými soubory pomocí serveru Apache, použitím několika základních direktiv. Věnovat se také budu komponentám, ze kterých se můj systém skládá, jaký je jejich účel a funkčnost a také role uživatelů s jejich právy a možnostmi. V závěru této kapitoly jsem věnoval celou podkapitolu klientským skriptovacím technologiím AJAX a JavaScript a jejich využití v aplikaci.

V závěru celé práce zhodnotím dosažený výsledek a nastíním možnosti rozšíření. Zmíním se o zkušenostech, které jsem při vývoji celého projektu získal, a také o problémech, které mě při realizaci potkaly.

Kapitola 2

Teorie

V následujících podkapitolách se pokusím rozebrat některé pojmy z teoretického hlediska, v další kapitole pak popíši vlastní postup a řešení dané problematiky.

2.1 Redakční systémy

Redakční systémy, jinými slovy systémy pro správu obsahu (Content management system, CMS v angličtině), jsou programy zajišťující správu dokumentů. V dnešní době jsou to většinou webové aplikace a správa jejich obsahu. Takové systémy jsou vhodné, jestliže obsah spravuje více lidí a tito lidé nemusí mít hluboké, popřípadě žádné znalosti technologií, se kterými systém pracuje (HTML...). Jejich obsluhu by měl zvládnout běžný poučený uživatel.

2.2 Základní funkce redakčního systému

Mezi základní funkce, které se z pohledu uživatelských práv dělí na administrátorské a uživatelské, patří:

- vkládání, publikaci a často i modifikaci článků,
- řízení přístupu k informacím a dokumentům, většinou se správou uživatelů a přístupových práv,
- správa diskusí či komentářů, ať už k publikovaným dokumentům nebo obecných,
- vkládání a správa souborů,
- správa obrázků či galerií,
- plánovací kalendář, kalendář akcí,
- nástěnka, která sdílí některé informace,
- ankety, atd...

Jak už jsem naznačil výše, uživatelská práva hrají v systému velkou roli. Zpravidla se uživatel dělí na:

1. Administrátor.

2. Uživatel,

3. Host.

Administrátor má běžně práva na vysoké úrovni a narozdíl od běžného uživatele má neomezený pohyb v systému (např. s právy mazání). Uživatel má většinou práva definovaná administrátorem jen na určitou část systému. V diskuzích je to většinou mazání a editace svých příspěvků. Host má práva velmi omezená, například jen práva pro čtení.

2.2.1 WYSIWYG editor

Pro jednodušší manipulaci a formátování textu je pro běžného uživatele velmi vhodné používat WYSIWYG (What You See Is What You Get) editor. Jeho princip je takový, že když změníte cokoli v textu (Barvu, velikost, tučnost, podtržení ...), ihned se to tak zobrazí a vy tudíž nemusíte znát HTML nebo jiný značkovací systém, který používá například redakční systém PHPBB ([7]). Na tomto principu jsou založeny textové editory jako například Word od firmy Microsoft. I webové aplikace již takovéto editory mohou obsahovat. Na Internetu existuje spousta takových, již hotových editorů (některé jsou komerční), ale existují i kvalitní OpenSourcové, tedy pro volné použití. Většina z nich je napsána čistě v JavaScriptu.

2.3 Architektura klient-server

Redakční systémy i většina dalších systémů a aplikací, které pracují nejen s databází, jsou založené na architektuře klient–server.

Princip této architektury je založen na dotazu a odpovědi. Klientů je obvykle více a pracují v tzv. aktivním módu, protože posílají požadavky na server. Server je obvykle jen jeden, běží na vzdáleném počítači a čeká na požadavky klientů.

V případě dotazu klienta na server, server zpracuje požadavek a pošle výsledek. U webových aplikací je to například webový prohlížeč – klient, kliknutí na odkaz – dotaz na server, zobrazení nové stránky – odpověď od serveru, v tomto případě vygenerovaný obsah nové stránky. Na serveru je dynamicky generováno HTML na základě požadavku klienta. Ke klientovi (prohlížeč) je zasílán už jen čistý HTML kód, který lze zobrazit v libovolném prohlížeči [1].

2.4 Webové servery

Instalace webového serveru je nutným základem pro vývoj webových aplikací, pokud potřebujeme pracovat se serverovými skriptovacími jazyky (viz. 2.5). Na výběr jich máme hned několik, zde však uvedu jen dva nejpoužívanější.

2.4.1 Apache

Asi nejpoužívanější webový server (cca 60 %) je bezpochyby Apache (dnes ve verzi 2.0), původně napsán pro Unixové systémy. Na Internetu však existuje spousta již přednastavených “user friendly” balíčků, pro instalaci i na systému Windows, včetně PHP a MySQL. Právě s těmito technologiemi se tento server nejčastěji používá, ale jde nainstalovat například i s podporou skriptovacího jazyka ASP.

2.4.2 IIS

Asi 30 % webových stránek na Internetu poskytuje webový server IIS (Internet Information Services) od firmy Microsoft. Ihned po Apachi je to druhý nejpoužívanější HTTP server. Nyní ve verzi 7.0 je součástí operačního systém Windows Vista i Windows Server 2008. Předchozí verze využívaly pro ukládání dat metabázi, novější verze využívají konfigurační model známý z ASP.NET, hierarchicky organizované XML soubory [9]. Přestože se IIS používá spíše pro technologie od Microsoftu, podporuje například i PHP.

Například "Sun Java System Web Server" je navržen spíše pro velké nebo střední komerční systémy. Používá se sice většinou s JSP nebo Java Servlet technologiemi, ale má například i podporu PHP.

2.5 Serverové skriptovací jazyky

Serverových skriptovacích technologií pro tvorbu webových aplikací je hned několik. Všechny však mají několik znaků společných, například výše popsané odeslání jen čisté HTML stránky po dotazu klienta, takže k prohlížení webu těchto technologií stačí obyčejný prohlížeč s podporou HTML. V následujících odstavcích popíši dva nejpoužívanější.

2.5.1 PHP

PHP (PHP Hypertext Preprocessor) je serverový skriptovací jazyk speciálně navržený pro psaní dynamických webových aplikací. Velká výhoda toho jazyka je, že je tzv. Open Source. To znamená, že zdrojový kód je volně přístupný a může se dále používat, distribuovat a upravovat bez jakýchkoli poplatků. Nejnovější verzi můžete zdarma kdykoliv stáhnout z manuálových stránek http://www.php.net.

PHP se zapisuje přímo do HTML mezi značky `<? ... ?>`, ale povoleny jsou i jiné zápisy, například i ty, které používá ASP (`<% ... %>`). Následující příkaz zobrazí uvedený text v prohlížeči:

```
<?php echo("Hello world!"); ?>
```

Syntaxe jazyka může být kombinovaná z různých programovacích jazyků (například Java, Perl, C), takže vývojáři dávají svobodně zvolit, který z těchto zápisů si vyberou.

Nejnovější verze PHP 5 již plně podporuje objektově orientované programování, čímž odpovídá na moderní trendy v programování. Obsahuje tedy funkce a syntaxi podobné čistě objektově orientovanému jazyku Java. Obsahuje například několikanásobné dědičnosti, soukromé a chráněné vlastnosti a metody, abstraktní třídy a metody, rozhraní, konstruktory a destruktory [3].

Tento jazyk se nejčastěji používá v kombinaci se serverem Apache, operačním systémem Linux a databázovým systémem MySQL. Díky tomu se stala oblíbená zkratka LAMP, která znamená kombinaci těchto technologií, tedy Linux, Apache, MySQL, PHP (Python, Perl). Na webu existuje spoustu kompletních softwarových balíků pro systém Windows, který je po instalaci možný začít ihned používat.

2.5.2 ASP

Active Server Pages je technologie vyvinutá společností Microsoft (narozdíl od PHP, která je opensource[1]) a její podpora je součástí každého IIS serveru. Přesto se tyto stránky dají také provozovat na webovém serveru Apache, ale tato kombinace je méně častá. ASP kód se podobně jako u PHP vkládá mezi značky `<%` ... `%>` a zapisují se přímo do HTML kódu. Pro vývoj stránek v ASP se používaly programovací jazyky jako JScript nebo VBscript.

Pro ukázku zde uvedu kód "Hello world", který tuto větu vypíše na stránce.

```
<% Response.Write "Hello World!" %>
```

Tyto dva jazyky se od sebe příliš neliší. Oba jsou založené na podobném principu. ASP se považuje spíše za objektově orientovaný jazyk vhodnější pro větší projekty. PHP je spíše jazyk s velkým množstvím funkcí, ale od verze 5 obsahuje také podporu objektově orientovaného programování. Obecně se říká, že přechod z jednoho jazyka na druhý nepředstavuje nijak zvlášť velký problém. Nelze jednoznačně určit, který z jazyků je lepší, proto záleží na preferencích každého vývojáře.

Mezi další serverové skriptovací jazyky patří například technologie JSP, CGI skripty, Python, ASP.NET a mnoho dalších, které tu jen zmíním. Rozebírat důkladněji je však nebudu, neboť to není hlavním předmětem této práce.

2.6 Klientské skriptovací jazyky

Klientské skriptovací jazyky jsou, jak již název napovídá, skripty prováděné na straně klienta. Ze serveru se po požadavku stáhne HTML stránka i s klientským skriptem, který se provádí podle potřeby.

Takových skriptovacích jazyků je hned několik, avšak nejznámější a nejpoužívanější je JavaScript (Jscript pod Microsoftem). Proto v následujících kapitolách tuto technologii popíši podrobněji i s pojmy, se kterými tato technologie úzce souvisí. Ostatní skripty, jako VBscript (zjednodušený Visual Basic), se ujaly jen minimálně, proto se jimi nebudu více zaobírat.

2.6.1 JavaScript

JavaScript je interpretovaný objektově orientovaný klientský skriptovací jazyk, který vychází z programovacích jazyků Java a C. Dále však s jazykem Java nemá kromě syntaxe více společného. Jelikož se jedná o klientskou technologii, je tento jazyk součástí všech moderních prohlížečů jako Internet Explorer, Mozilla Firefox, Opera či Netscape. JavaScript má potlačenou typovou kontrolu, takže typ proměnné rozlišuje až poté, co se do ní vloží nějaká hodnota.

JavaScript se může psát přímo do HTML kódu mezi značky `<script>...</script>` nebo se na něj lze odkázat přímo ze samostatného souboru. Nejčastěji se JavaScript používá jako reakce na nějakou akci v zobrazené stránce prohlížeče. Jeho výhoda je, že může "rozhýbat", či jinak změnit obsah HTML stránky, bez nutnosti znovunačtení stránky. Můžou se například kontrolovat korektně vyplněná data z formulářů, například pomocí regulárních výrazů, ještě před odesláním na server. Díky moderní technologii AJAX, kterou podrobně rozeberu v kapitole 2.6.2, se data mohou poslat na server dokonce bez znovunačtení stránky.

[1] Opensource – zdrojové kódy jsou každému k dispozici a každý programátor si jej může dle vlastních potřeb upravit.

XML

O této technologii se zde ještě mnohokrát zmíním, a to i přesto, že nesouvisí jen s JavaScriptem, pokládám za důležité o tomto jazyku napsat pár řádků.

XML (Extensible Markup Language) je jednoduchý značkovací jazyk, který je odvozen od univerzálního jazyka SGML. Původně byl navržen pro popis a publikování dokumentů, ale hraje také důležitou roli při výměně dat mezi aplikacemi nejen na Internetu. XML dokument může mít libovolnou strukturu a lze v nich používat libovolně pojmenované a strukturované prvky. Dokument začíná standardní deklarací, která definuje XML dokument s kódováním unicode (UTF-8):

```
<?xml version="1.0" encoding="UTF-8"?>
```

K definici metadat XML slouží jazyky pro popis schématu XML dokumentu, které určují, jaké prvky a atributy můžeme v XML používat, jak je lze navzájem kombinovat a co mohou obsahovat. Protože je XML schema napsáno v jazyce XML, můžeme s ním manipulovat například pomocí DOM (viz. [8]).

Ukázku XML schematu můžete vidět na obrázku 3.10. Klient toto XML schema používa ke čtení odpovědi ze strany serveru za pomocí objektu `XMLHttpRequest`.

V souvislosti s webem se často hovoří o jazyku XHTML, který vyvinulo konsorcium W3C[2]. Tento jazyk vychází právě z XML, proto, na rozdíl od klasického HTML, musí splňovat podmínky pro zápis tohoto jazyka. Původně měl tento jazyk nahradit HTML. Jeho hlavní výhodou je, že se snaží více oddělit strukturu od formátování, proto používá striktní syntaxi. Pro formátování se užívají kaskádové styly. Hlavní výhodou XHTML, oproti HTML, je jednodušší implementace a jednodušší zpracování.

XML je poměrně nová technologie, která v dnešní době získává stále více na oblibě. Čím dál více aplikací využívá tento jazyk k ukládání a předávání různých druhů informací, které si s sebou nenesou pouze samotný text, ale i informaci o jeho významu. Další zpracování nebo konverze do jiného formátu pak může být jednoduchá a rychlá. Ještě na závěr této podkapitoly uvedu pár příkladů použití jazyka XML:

- XHTML – jazyk HTML, který vychází z XML,
- RSS – čtení novinek z internetových stránek,
- DocBook – dokumentace a publikace,
- MathML – popis matematických vzorců a symbolů na webu,
- Jabber – Instant Messaging protocol,
- SVG – dvourozměrná vektorová grafika,
- MusicXML – zápis notové osnovy.

a mnoho dalších ...

[2] W3C – je mezinárodní konsorcium, které vyvíjí standardy pro Internet.

DOM

DOM (Document Object Model) je oblíbenou formou reprezentace dokumentů XML. Nejedná se o nejrychlejší nebo nejjednodušší způsob, ale je nejběžnější. DOM je implementován ve spoustě programovacích jazyků (Java, Perl, PHP, JavaScript ...) a byl vytvořen, aby poskytl intuitivní způsob, jak procházet hierarchii XML.

Protože validní HTML je podmnožinou XML, existence efektivního způsobu pro analýzu a prohlížení DOM je nezbytná, aby bylo programování v JavaScriptu jednodušší. Velká část událostí v JavaScriptu probíhá mezi JavaScriptem a různými prvky jazyka HTML, které obsahuje nějaká webová stránka, proto je DOM vhodný nástroj, jenž tento proces zjednodušuje [6].

Jedním z častých příkazů využívající DOM je například `document.getElementById` nebo `document.getElementByTagName`. Uvedu zde malou ukázku použitím těchto příkazů:

```
<html>
  <head>
    <title>Titulek stránky</title>
    <script type="text/JavaScript">
      ...
      clock = document.getElementById("mainClock");
      clock.innerHTML=TimeValue;
      ...
      nadpis = document.getElementByTagName("h3");
      nadpis.style.fontSize=14+"px";
    </script>
  </head>
  <body>
    <h3>Příklad na DOM</h3>
    <div id="mainClock"></div>
  </body>
</html>
```

V HTML je element DIV, který má dané ID a v tomto případě JavaScript pomocí příkazu `document.getElementById("mainClock")` a innerHTML vloží mezi tyto značky nějakou hodnotu. V druhém případě příkaz `document.getElementByTagName("h3")` nastaví pomocí stylu určitou velikost písma všech nadpisů třetí úrovně.

CSS a DHTML

Nyní rozeberu, jak spolu úzce souvisí pojmy kaskádové styly a JavaScript. Nejdříve trochu teorie, co to vlastně kaskádové styly jsou.

Kaskádové styly (CSS, Cascading Style Sheets) jsou standardem pro vytváření stylů a zobrazování webových stránek napsaných v HTML, XML nebo XHTML. Rozšiřuje značkovací jazyk HTML o nové vlastnosti, které v tomto jazyce chybí. Pomocí kaskádových stylů se dá částečně oddělit obsah od jeho struktury, což je v podstatě hlavní účel kaskádových stylů.

Spolupráce mezi JavaScriptem a CSS je základem moderního programování v JavaScriptu a velmi úzce souvisí s pojmem DHTML. Jedná se tedy o dynamický HTML, který umožňuje dynamickou změnu jinak statického obsahu stránek. DHTML není žádný standard či něco podobného, ale je to kombinace technologií JavaScript, CSS, HTML a často i DOM. Zde

uvedu ukázku mého zdrojového kódu, který zobrazí formulář vypadající jako nové vyskakovací (pop-up) okno, ale ve skutečnosti je to obyčejné HTML, vytvořené JavaScriptem.

```
function display_send_messageform(...)
{
  ...
  var newelement = document.createElement("div");
  newelement.style.visibility = "hidden";
  newelement.setAttribute("id","UserMessageDiv");
  newelement.style.top=topOffset;
  newelement.style.left=leftOffset;
  var HTML=
  // zde vložím kód pro formulář, který chci zobrazit v "pop up" okně

  if (newelement != null)
  {
    newelement.innerHTML=HTML;
    newelement.style.visibility = "visible";
  }

  var ParentElement = document.getElementById("ParentID");
  ParentElement.appendChild(newelement); // pripojeni html
  ...
}
```

Zde je vidět použití DOM funkcí `ParentElement.appendChild(newelement)` pro připojení nově vytvořeného elementu ke svému nadřezenému.

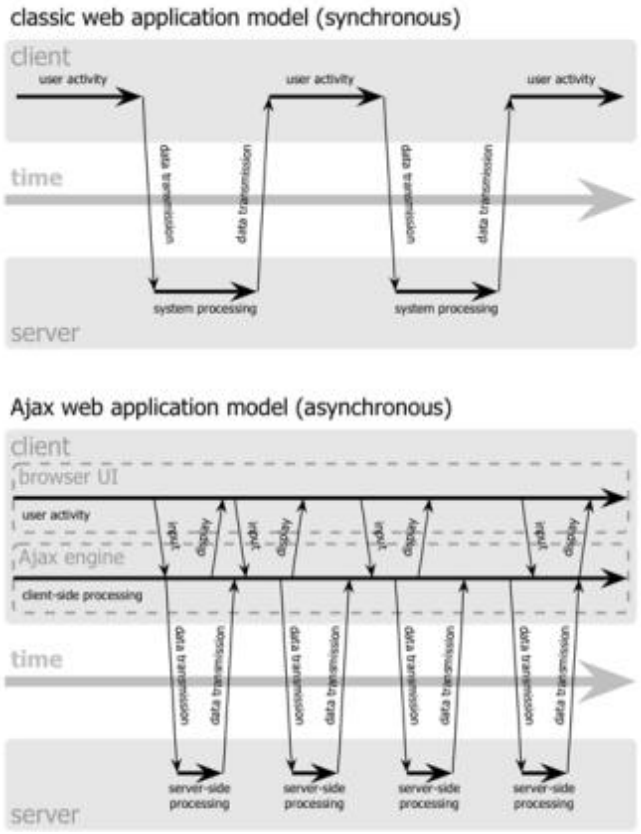
Uvádím také ukázku se styly pomocí příkazu `newelement.style.top=topOffset`, kde používám aktuálně vypočítanou hodnotu, v tomto případě relativní od pozice kurzoru myši. Zobrazil jsem tak malý formulář jen pomocí klientských technologií bez dotazu na server. Data z formuláře se poté odesílají pomocí technologie AJAX, takže za celou dobu nedojde ani k jedinému znovunačtení stránky. Takto vytvořený formulář můžete vidět na obrázku 3.9.

2.6.2 AJAX

Technologie AJAX je zkrácený název pro Asynchronní JavaScript a XML a umožňuje, jak již název napovídá, asynchronní komunikaci mezi klientem a serverem. K tomuto druhu komunikace byl navržen objekt `XMLHttpRequest`, který pro koumunikaci využívá HTTP protokol. Rozdíl mezi klasickou synchronní a asynchronní komunikací, kterou využívá technologie AJAX, je na obrázku 2.1. Data jsou většinou posílána ve formátu XML, ale mohou být použity i jiné formáty, například HTML nebo jen obyčejný text. Komunikace se serverem se děje bez nutnosti načtení celého obsahu stránky, výměna dat a změna obsahu stránek mohou probíhat na pozadí, často bez vědomí uživatele. To je výhodné například při odeslání či přijetí malého množství dat, odeslání formulářů bez nutnosti znovunačtení stránky, atd. To jistě patří mezi hlavní výhody této technologie.

Mezi hlavní nevýhody patří změny v paradigmatu používání webu. Webové stránky se chovají jako plnohodnotná aplikace se složitou vnitřní logikou, nikoli jako posloupnost stránek, mezi kterými se lze navigovat i pomocí tlačítek Zpět a Další. Moderní AJAXové

aplikace jsou schopny funkce těchto tlačítek (přinejmenším částečně) obnovit za použití různých technik (např. využití části adresy za znakem # či pomocí neviditelných IFRAMEs) (Viz [6]). Další nevýhoda je například nutnost používání moderních prohlížečů, které tuto technologii podporují. Nejpoužívanější prohlížeče jako Internet Explorer, Firefox nebo Opera tuto možnost většinou mají.



Obrázek 2.1: Rozdíl mezi synchronní a asynchronní komunikací.[2].

2.6.3 Nevýhody

Nevýhoda těchto skriptů je, že je lze volně zobrazit v prohlížeči po načtení stránky. Může to být například problém u aplikací, které jsou napsané převážně na těchto technologiích, neboť si jejich zdrojový kód může kdokoliv volně stáhnout bez vědomí autora.

Ale existují metody, které takové čtení kódu mohou minimálně znepříjemnit. Jednou z nich je tzv. obfuskace zdrojového kódu. Příklad uvedu na následujícím zdrojovém kódu:

```
function get_online_users1()
{
  var xmlHttp = createXmlHttpRequestObject();
  if (xmlHttp)
  {
    try
    {
```

```
        xmlHttp.open("POST", "get_online_users.php", true);
        xmlHttp.setRequestHeader("Content-Type", "application/...");
        xmlHttp.onreadystatechange = handleRequestStateChangeOnline;
        xmlHttp.send(null);
      }
      catch (e)
      {
        display_warning("Error:\n" + e.toString());
      }
    }
  }
```

Záměnou názvu proměnných a odstranění odřádkování a odsazení dostáváme kód, který je téměř nečitelný:

```
function aa(){var b=a();if(b){try{b.open("POST","aa.php",true);
b.setRequestHeader("Content-Type","application/...");
b.onreadystatechange =d; b.send(null);}catch(e){ c("Error:\n" +
e.toString());}}}
```

Toto byl jen malý příklad. Osobně jsem už na pár takových zdrojových kódů narazil, a i když autor nechá původní názvy proměnných, po odstranění odřádkování a bílých znaků se kód stává absolutně nečitelným.

2.7 Psaní kódu a phpDocumentator

Systém by měl být dobře srozumitelný a pochopitelný nejen pro uživatele, kteří se systémem pracují, ale i pro vývojáře, kteří budou chtít systém rozšířit nebo například jen najít chybu. Tím mám teď na mysli čitelnost zdrojového kódu. Pravidel pro psaní zdrojových kódů je hned několik. Pěkně rozepsané a popsané je najdete například v [4].

Psaní komentářů je nesmírně důležité a někdy i hodně podceňované. Z vlastní zkušenosti jsem poznal, jak je obtížné modifikovat vlastní program, který není řádně okomentován. Od té doby se snažím komentovat co možná nejvhodněji. Komentovat by se měly nejen jednotlivé funkce, ale i cykly, podmínky a jiné důležité části kódu.

Velmi výhodné je používat nástroj pro automatickou tvorbu dokumentace phpDocumentator. Tento nástroj vychází z JavaDoc, který byl určen pro Javu. Jde tedy o způsob psaní komentářů, kterým tento nástroj rozumí a vygeneruje s popisem funkcí například HTML stránku, nebo novější verze i PDF nebo XML formát. Vygenerovat se může buď přes webové rozhraní nebo pomocí příkazové řádky. Ukázka takto okomentovaného kódu může vypadat následovně (viz. [10]):

```
<?php
  /**
  * POPIS OBSAHU SOUBORU
  *
  * nadpis souboru
  * @author Jmeno Autora
  * @version 1.0
  * @package Trida
```

```
  */

  /**
  * Nazev a popis tridy
  */
  class get_rows
  {

  /**
  * Popis privatni promenne
  * @access private
  * @var typ promenne
  */
  private $count;

  /**
  * popis meotdy
  * @param $nazev_parametru typ_parametru popis parametru
  * @return typ navratova hodnota
  */
  public function do_query($query)
  {
    ...
    ...
  }
}
?>
```

2.8 Databázové technologie

Pokud budeme chtít vyvíjet nějakou aplikaci, musíme najít způsob, jak pracovat s různými infromacemi. Nabízí se ukládání a načítání dat ze souboru. Pro práci se souborem nabízí skriptovací jazyky velké množství funkcí, ale toto řešení je v mnoha ohledech zcela nevyhovující. Spousta problémů vzniká, pokud by například chtěly najednou dva skripty i přes použití zámků zapisovat. Další problém by byl při vyhledávání komplexnějších informací, protože by se musel prohledávat každý řádek zvlášť. Zapisování do souboru se v tomto případě hodí nejvýše na logování některých událostí, například chyb.

Daleko vhodnějším řešením je použití relačního databázového systému, kde jsou data uspořádána do tabulek (relací). Ty umožňují mnohem rychlejší a efektivnější přístup k datům než soubory, navíc mají zabudované mechanismy paralelního přístupu k datům a spoustu dalších operací jako vkládání, zobrazování, mazání a další, takže jako vývojáři se již nemusíme starat o implementaci těchto operací. Mohou například vyhledávat množství dat podle předem zadaných kritérií a také se používá velmi vyspělý systém práv a zabezpečení.

Pro práci právě s těmito daty v relační databázi byl navržen jazyk SQL (Structured Query Language). Strukturovaný dotazovací jazyk, jak již název připomíná, pracuje na principu dotaz–odpověď. Disponuje celkem velkým množstvím příkazů nejen pro práci s daty jako vkládání, zobrazování, mazání a aktualizace, ale i pro vytváření nových tabulek, mazání a změnění existujících, nebo například příkazy pro změnu a nastavení práv.

V následujících podkapitolách rozepíši některé z často používaných relačních databázových systémů.

2.8.1 MySQL

MySQL je databázový systém vytvořený švédskou firmou MySQL AB. Je považován za úspěšného průkopníka dvojího licencování, protože je k dispozici jak pod bezplatnou licencí GPL, tak pod komerční placenou licencí.

MySQL je multiplatformní databáze. Komunikace s ní probíhá, jak už název napovídá, pomocí jazyka SQL. Podobně jako u ostatních SQL databází se jedná o dialekt tohoto jazyka s některými rozšířeními.

Díky své snadné implementovatelnosti (lze jej instalovat na Linux, MS Windows, ale i další operační systémy), výkon a především díky tomu, že se jedná o volně šiřitelný software, má vysoký podíl na v současné době používaných databázích. Velmi oblíbená a často nasazovaná je kombinace MySQL, PHP a Apache jako základní software webového serveru.

MySQL bylo od počátku optimalizováno především na rychlost, a to i za cenu některých zjednodušení: má jen jednoduché způsoby zálohování a až donedávna nepodporovalo pohledy, triggery a uložené procedury. Tyto vlastnosti jsou doplňovány teprve v posledních letech, kdy začaly nejčastějším uživatelům produktu, programátorům webových stránek, již poněkud scházet, viz [11]. Přihlásit k databázovému serveru se můžeme například z konsole po zadání příkazu:
`mysql -h` *hostitel* `-u` *uživatel* `-p`

Po úspěšném přihlášení můžeme pomocí příkazů pracovat libovolně v MySQL a provádět příkazy, které tento jazyk dovoluje. Asi uznáte, že pamatovat si všechny možné příkazy je pro méně zkušeného správce databáze téměř nemožné. Z těchto důvodů existuje aplikace PhpMyAdmin.

PhpMyAdmin

PhpMyAdmin je nástroj napsaný v jazyce PHP a umožňuje správu databázového serveru MySQL pomocí webového rozhraní. Tato aplikace se stala velice oblíbenou, protože práce s ní je opravdu jednoduchá a poskytuje opravdu velké množství funkcí. Kromě klasických operací s databázemi a tabulkami obsahuje například statistiku aktuálního stavu serveru, počet přenesených dat, počet provedených dotazů, průměr dotazů za hodinu, minutu i sekundu a spousta dalších.

Další velice užitečnou funkcí této aplikace je možnost exportu dat do různých formátů. Exportovat data se můžou například do těchto formátů: CSV, MS Excel, MS Word, LaTeX, PDF, SQL ale například i moderní XML. K dispozici je další velká spousta nastavení, například export do souboru i za použití komprese zip či gzip, smazání databáze po exportu a spousty dalšího. Formát importovaného souboru je pouze SQL, ale můžeme importovat například i ze zazipovaného formátu.

Ještě zmíním, že existuje jednodušší verze PhpMinAdmin, která má podobu jednoho 150kb PHP skriptu.

2.8.2 PostgreSQL

PostgreSQL je plnohodnotným relačním databázovým systémem s otevřeným zdrojovým kódem. Má za sebou více než patnáct let aktivního vývoje a má vynikající pověst pro svou

spolehlivost a bezpečnost. Běží na všech rozšířených operačních systémech včetně Linuxu, UNIXů a Windows. Plně podporuje cizí klíče, operace JOIN, pohledy, spouště a uložené procedury. Obsahuje většinu SQL92 a SQL99 datových typů, např. INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL a TIMESTAMP. K systému existuje kvalitní volně dostupná dokumentace včetně českých překladů FAQ a FAQ pro operační systémy firmy Microsoft.

PostgreSQL je šířen pod BSD licencí, která je nejliberálnější ze všech open source licencí. Tato licence umožňuje neomezené používání, modifikaci a distribuci PostgreSQL. PostgreSQL je možno šířit se zdrojovými kódy nebo bez nich, zdarma nebo komerčně.

Předností systému PostgreSQL je rozšiřitelnost. Systém může být bezproblémově rozšiřován o nové datové typy, funkce operátory, agregační funkce, procedurální jazyky. Díky tomu mohly vzniknout následující rozšíření: PostGIS – podpora pro geografické informační systémy, TSearch2 – podpora fulltextového vyhledávání, Slony-I – master to multiple slaves replikace. Na serveru pgfoundry je k dispozici několik desítek doplňků včetně doplňků rozšiřující o funkcionalitu MySQL, SQL Serveru a Oraclu. Viz [11]. Podobně jako MySQL i PostgreSQL obsahuje webovou aplikaci PhpPgAdmin napsanou v PHP, která slouží pro správu databázového serveru.

2.8.3 Oracle

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

Aktuální verze podporuje nejen standardní relační dotazovací jazyk SQL podle normy SQL92, ale také proprietární firemní rozšíření Oracle (např. pro hierarchické dotazy), imperativní programovací jazyk PL/SQL rozšiřující možnosti vlastního SQL (v tomto jazyce je možné tvořit uložené procedury, uživatelské funkce, programové balíky a triggery), dále podporuje objektové databáze a databáze uložené v hierarchickém modelu dat (XML databáze, jazyk XSQL). Viz [11].

Kapitola 3

Analýza, návrh řešení a implementace

V této kapitole se zaměřím na analýzu a řešení daného problému. Pokusím se popsat použité technologie, vývojové prostředí, použitý operační systém a jiné postupy, které mě dovedly k řešení a realizaci celého projektu. Většinu problému jsem konzultoval přímo se zákazníkem, tedy administrátorem firmy, pro kterou systém vyvíjím.

3.1 Výběr technologií

Pro samotný vývoj aplikace jsem zvolil serverový skriptovací jazyk PHP z důvodu, že jsem s tímto jazykem neměl žádné předchozí zkušenosti a chtěl jsem se tento jazyk naučit a do budoucna tímto získat zkušenosti. Další technologie, bez kterých jsem se při řešení problémů nemohl obejít, byly: JavaScript (XML, AJAX), HTML, Kaskádové styly a databázová technologie MySQL.

3.2 Výběr software a skriptovacích jazyků

Výběr prostředí pro vývoj systému, je jednou se základních věcí při vývoji Softwarového produktu. Existuje celá spousta WYSIWYG editorů pro tvorbu HTML, CSS nebo JavaScriptu, ale již od střední školy se nás od takových editorů snaží odradit. Pro tvorbu PHP skriptů, Javascripů a HTML jsem tedy zvolil linuxový textový editor Kate a pro nápovědu k HTML atributům a CSS jsem používal editor Quanta Plus, který se nachází v běžném balíčkovacím systému distribuce Kubuntu.

3.2.1 Operační systém

Výsledný systém bude spuštěn na firemním intranetu pod linuxovou distribucí Fedora Core s kombinací LAMP[1].

Pro vlastní vývoj jsem tedy zvolil také operační systém linux, distribuci Kubuntu. Také z důvodu, že tento systém používám denně již několik let, a nejen s konfigurací serveru mám četné zkušenosti.

[1] Zkratka LAMP znamená kombinaci Linux Apache MySql a Php (Perl nebo Python). Protože se jedná o svobodný software a technologie, tato kombinace se stala velmi oblíbenou pro provoz a vývoj dynamického webu.

3.2.2 Požadavky na prohlížeče

Vzhledem k tomu, že systém budou používat běžní uživatelé, je naprostou samozřejmostí zajistit co největší kompatibilitu mezi nejpoužívanějšími prohlížeči. Ve firmě se používají 3 prohlížeče, a to Internet Explorer, Firefox a Opera, většinou nejnovější verze. Jiné prohlížeče jako Netscape nebo Safari uvažovat nemusím, neboť se ve firmě nepoužívají. HTML kód, JavaScript a CSS tudíž musím psát tak, aby výsledný systém běžel v uvedených prohlížečích. Žádná jiná podmínka není potřeba, a proto nemusím řešit například úplnou validitu kódu podle standardu W3C.

3.3 Návrh databáze

Vývoj celé aplikace se dále vyvíjí podle celkové struktury databáze, takže je potřeba věnovat návrhu a analýze zvýšenou pozornost. Při návrhu relačního modelu jsem vycházel z požadavků zákazníka a vždy se pokoušel získat co nejvíce informací o budoucím systému, abych strukturu nemusel časem příliš měnit, neboť to přináší zbytečné komplikace při implementaci a servisu.

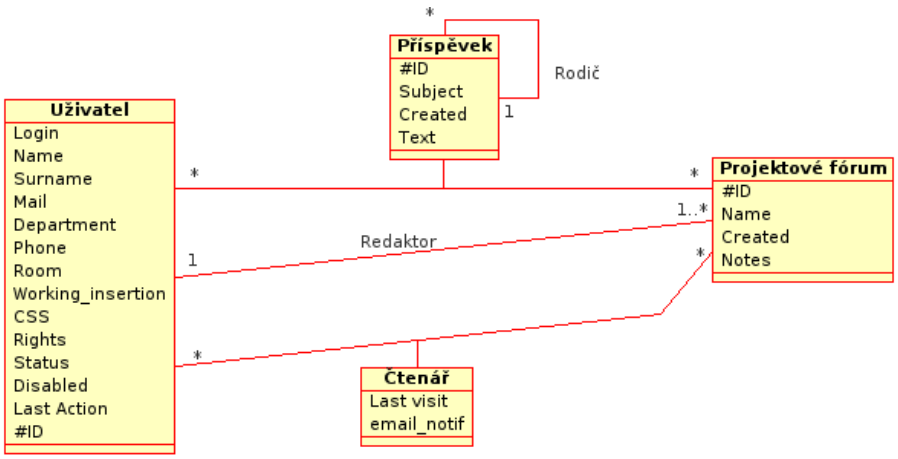
Zadání a získané informace jsem si rozdělil do tzv. entit, nesoucí další informace, které je potřeba ukládat do databáze. Vytvořil jsem ER–Diagram (poslední verze diagramu viz Příloha 1), který reprezentuje danou strukturu. Mezi entitami jsem vytvořil vztahy `1:N` a `N:N`. Přidal jsem cizí klíče, vytvořil vazební tabulky mezi vztahy `N:N` spolu s dalšími atributy tohoto vztahu, a tím jsem připravil návrh tabulek, které jsem vytvářel v aplikaci PhpMyAdmin.

Pokusím se tu rozebrat návrh a analýzu několika tabulek, které jsem v aplikaci použil. Takto vypadala část zadání od zákazníka:

Ke každému projektu bude existovat diskusní fórum, které by mělo nahradit e–mailovou i IM komunikaci. Správou každého fóra bude administrátorem pověřen redaktor, diskuse nebude moderovaná. Přístup pro čtení a odesílání zpráv a souborů bude možný jen pro určené čtenáře. Veškerá komunikace ve fóru musí být archivována a nesmí být odstranitelná ani redaktorem, ani čtenáři. Příspěvky budou mít stromovou strukturu a na vyžádání bude čtenářům zasíláno avizo o nových zprávách či souborech.

Toto zadání jsem si rozdělil na tyto základní entity: uživatel, projektové fórum, příspěvek a čtenář. Poté jsem si ověřil další informace, například jestli může být redaktorů víc nebo jen jeden, a získal všechny potřebné údaje, které by se měly v databázi ukládat. Dle zadání jsem vytvořil vztahy mezi entitami a určil jejich kardinalitu. Mezi entitami uživatel a projektové diskuze jsem vytvořil dva vztahy s kardinalitou `N:N`. Jedna vztahová množina představuje příspěvek a druhá informace o čtenářích. Poté jsem přidal vztahy s kardinalitou `1:N` pro entitu Příspěvek, protože by příspěvky měly mít stromovou strukturu, je potřeba uchovávat informaci o rodiči jednotlivého příspěvku. Další a poslední vztah `1:N` jsem přidal mezi entitní množiny uživatel a projektové diskuze, který obsahuje informaci o redaktorovi, kterého by měla mít každá diskuze. Výsledný ER–diagram by pak mohl vypadat podle obrázku 3.1. Poté jsem dle kardinalit vytvořil cizí klíče a diagram transformoval na tabulky relační databáze. Výsledný SQL skript pro vytvoření tabulek pak vypadá takto:

```
CREATE TABLE user (
  id INT( 8 ) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY ,
  login VARCHAR( 20 ) NOT NULL ,
  name VARCHAR( 100 ) NULL ,
```



Obrázek 3.1: ER diagram příkladu v kapitole 3.3

```
  surname VARCHAR( 100 ) NULL ,
  mail VARCHAR( 200 ) NULL DEFAULT '@',
  department VARCHAR( 250 ) NULL,
  phone VARCHAR( 100 ) NULL,
  room VARCHAR( 200 ) COMMENT,
  working_insertion VARCHAR( 200 ) NULL,
  css VARCHAR( 30 ) NOT NULL DEFAULT 'main.css',
  rights VARCHAR( 15 ) NOT NULL DEFAULT 'user',
  status VARCHAR( 8 ) NOT NULL DEFAULT 'offline',
  disabled BOOL NOT NULL DEFAULT '0',
  last_action INT NOT NULL DEFAULT '0'
)CHARACTER SET utf8 COLLATE utf8_czech_ci;

CREATE TABLE project_discussion (
  id INT( 5 ) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY ,
  name VARCHAR( 100 ) NOT NULL ,
  fk_user_login INT NOT NULL ,
  created DATETIME NOT NULL,
  note TEXT NULL
)CHARACTER SET utf8 COLLATE utf8_czech_ci;

CREATE TABLE allowed_reader (
  id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY ,
  fk_discussion INT NOT NULL ,
  fk_user INT NOT NULL ,
  last_user_visit INT NOT NULL,
  email_notification TINYINT UNSIGNED NOT NULL DEFAULT '1'
)CHARACTER SET utf8 COLLATE utf8_czech_ci;

CREATE TABLE message (
  id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY ,
```

```
  fk_user VARCHAR( 10 ) NOT NULL ,
  subject VARCHAR( 100 ) NOT NULL ,
  children INT NOT NULL DEFAULT '0',
  fk_discussion INT NOT NULL ,
  created INT NOT NULL ,
  text TEXT NULL ,
  parent INT NOT NULL
)CHARACTER SET utf8 COLLATE utf8_czech_ci;
```

Protože databáze nyní obsahuje 11 tabulek a v nejbližší době se při rozšiřování aplikace toto číslo bude zvětšovat, další postupy při návrhu těchto tabulek nebudu podrobně rozebírat, neboť byly velmi podobné příkladu, který jsem tu nyní rozebral. Zmíním zde však tabulky se kterými redakční systém pracuje.

Podobně jako projektová fóra existuje tabulka dokumentace, která také obsahuje příspěvky stromové struktury a soubory jako přílohy. Neobsahuje však tabulku čtenářů, neboť se řídí jinými pravidly (práva a více je popsáno v kapitole 3.6.2 – Systém dokumentace). Tabulka users messages, která je ve vztahu s kardinalitou `N:N` k uživateli, obsahuje informace o osobních zprávách mezi uživateli. Poslední tabulka user log obsahuje informace o přihlášení a odhlášení uživatelů. Viz Příloha 1 – ER diagram.

Nyní jsem rozebral návrh a analýzu databáze a v následujících podkapitolách se více zaměřím na nejdůležitější datové struktury a typy se kterými jsem se při vývoji systému setkal, a také popíši a zdůvodním zvolené řešení.

3.3.1 Datum a čas

V databázi existuje spousta záznamů, kde je potřeba uložit datum a čas. Je několik způsobů, jak toho docílit. Například vytvořit sloupec s datovým typem TIMESTAMP([m]), kde na hodnotě `m` závisí zobrazovací formát [3]. Například TIMESTAMP(12) zobrazí čas ve formátu RRMMDDhhmmsss. Poté je vhodné vkládat datum a čas pomocí funkce MySQL `NOW()`. Tedy po provedení následujícího příkazu:

```
INSERT INTO test (time)
VALUES (NOW());
```

se do databáze uloží následující hodnota `2008-04-05 16:27:25`, což je výhodné, pokud s datem nepotřebujeme dále pracovat a spokojíme s jeho zobrazení práve v tomto tvaru. Pro úpravu a jiný formát data se může použít buď na úrovni PHP různé funkce pro přeformátování řetězce nebo funkce `MySql DATE_FORMAT`. Například tento příkaz

```
SELECT DATE_FORMAT(time,'%e.%c.%Y') as time FROM test
```

zobrazí datum ve formátu "5.4.2008". Když chceme však seřadit tyto záznamy podle data vložení (`order by time`), seřadí se podle prvních čísel.

Zvolil jsem ukládání data a času v MySQL databázi datový typ sloupce INT a hodnoty do databáze ukládám ve tvaru, který vrací funkce PHP `time();` [2]. Dnešní datum pomocí PHP funkce `time()` (tedy 5.4.2008 17:46:21) se do databáze uloží ve tvaru `1207410381`. Toto číslo pak zpracuji funkcí `format()`, která je implementována jako metoda třídy `time`.

[2] Funkce vrací 32bitové číslo obsahující počet sekund od půlnoci roku `January 1 1970 00:00:00 GMT`, datu kterému se říká počátek *unixové epochy* [3].

Jestli je vrácené datum dnešní, vrátí se pouze čas, nikoli celé datum. Uživatelé tak ihned vidí, že například příspěvěk diskuze byl napsán dnes. Jestliže není datum dnešní, vrátí se v celém formátu, tedy i s datem.

```
/**
 * reformat date from function time()
 * 435776343 -> 3.5.2004 13:32:21
 *
 * if it is today it reformat like:
 * 435776343 -> 13:32:21
 *
 * @param int $time input time
 * @return string $date reformated date
 * @access public
 * @author Lukas Pohl
**/
public function format($time)
{
  $OldDay=date("Ymd",$time);
  $todayDay=date("Ymd",time());

  //message was sent today, we will not display today date
  if($OldDay==$todayDay)
  {
    return date("H:i:s",$time);
  } else
  {
    return date("d.m.Y H:i:s",$time);
  }
}
```

Tento způsob ukládání data a času v databázi má několik dalších výhod. Například porovnání dat, které je větší či menší (tedy. starší či novější) nepotřebuje zpracování dalšími funkcemi. Další výhodou je jednodušší provádění aritmetických operací s tímto formátem – když potřebuji toto datum zmenšit o 10 minut, jen odečtu 600 sekund (10*60s). To využívám například při odstranění uživatelů ze seznamu připojených (popsané v kapitole 3.7.2).

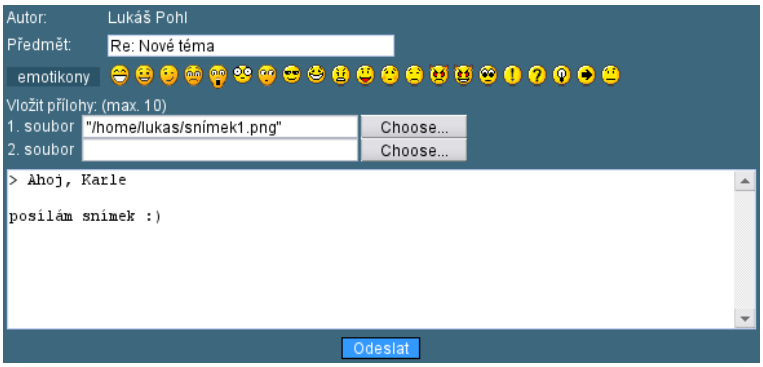
Má to samozřejmě také nevýhody. Asi největší nevýhodou funkce `time();` je, že číslo je 32 bitové, takže vrací datum jen v rozsahu let 1970 až 2038, což někdy může představovat problém. Další nevýhoda je, když zobrazím údaje přímo z databáze bez dalšího zpracování, tak nejsem schopen z těchto čísel vyčíst datum, ale maximálně rozeznat starší a novější.

3.3.2 Upload a ukládání souborů

Uživatelé mají možnost ke každému příspěvku jak v projektových fórech, tak v systému dokumentace, přiložit maximálně 10 souborů. Mám na výběr minimálně ze dvou možností, jak data ukládat v databázi a každá z nich má pochopitelně výhody a nevýhody.

Jednou z možností je ukládat v databázi pouze názvy souborů a pomocí nich se odkazovat na soubory uložené přímo na serveru.

Další možností je, a tu jsem zvolil i já, vytvořit sloupec datového typu BLOB v MySql, který slouží pro ukládání binárních dat přímo v databázi. Nevýhodou je, že se zvětšuje velikost databáze. Výhodou je například při zálohování celé databáze, kdy se zálohují s daty i všechny soubory, což je v tomto případě velice důležité. Záloha souborů ze serveru nebude tak častá, jako záloha dat (poměr zhruba 30:1).



Obrázek 3.2: Formulář pro vložení příspěvku

Když bude chtít uživatel odpovědět na nějaký příspěvěk nebo vložit nový, objeví se mu formulář podobný obrázku 3.2. Jak jsem již psal, je možné uložit až 10 souborů. Z hlediska uživatelského rozhraní by nebylo příliš vhodné, aby se zobrazilo 10 prázdných formulářů pro uložení souboru, protože příspěvek také žádný soubor obsahovat nemusí. To řeší JavaScriptová funkce, která uživateli zobrazí jen jeden formulář pro vložení souboru a v případě vybrání prvního souboru se mu zpřístupní další a takhle dál až po zobrazení všech deseti formulářů. Uživatel může uložit libovolný formát souborů, ale je omezen velikostí na 16 MiB.

Tabulka, kterou ukládám soubory do databáze, má následující strukturu:

```
CREATE TABLE doc_files (
  id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY ,
  fk_doc_message INT UNSIGNED NOT NULL ,
  name VARCHAR( 30 ) NOT NULL ,
  type VARCHAR( 30 ) NOT NULL ,
  size INT NOT NULL ,
  content MEDIUMBLOB NOT NULL
)CHARACTER SET utf8 COLLATE utf8_czech_ci;
```

V tabulce ukládám název souboru, MIME type[3], velikost a samotný obsah souboru. Tyto informace používám při zobrazení souboru pomocí PHP skriptu:

```
  ...
  header("Content-length: $size");
  header("Content-type: $type");
  header("Content-Disposition: attachment; filename=$name");
  echo $content;
  ...
```

[3]MIME (Multipurpose Internet Mail Extensions) je internetový standard, který rozšiřuje původní standard RFC822 o možnost posílání obrázků, zvuků, zpráv s diakritikou a jiném formátu. Informace o obsahu zprávy se skládá z typu a podtypu (více viz [12]).

3.4 Návrh uživatelského rozhraní

Jelikož produkt budou používat běžní uživatelé, je celkem důležité věnovat návrhu uživatelského rozhraní patřičnou pozornost. Takové uživatelské rozhraní je potřeba navrhnout tak, aby uživatel již na první pohled zhruba pochopil jeho funkčnost a snadno a rychle se jej naučil ovládat. Příliš složité, nekompletní nebo příliš pomalé ovládání by uživatele časem odradilo a náš systém by byl k ničemu [4]. Všechny principy ovládání, zobrazení a chování systému jsem pečlivě konzultoval se zákazníkem. Změnu vzhledu, jako vlastnost uživatelského účtu, jsem popsal v kapitole 3.4.2.

3.4.1 Layout a třída page

Na obrázku 3.3 je návrh základního rozhraní systému a jeho obsahu [4]. Jak je vidět, layout se skládá zhruba z šesti základních částí, které se teď pokusím více rozebrat.

Obrázek 3.3: Základní layout stránky

1. Horní panel slouží jako banner a obsahuje logo firmy vlevo nahoře a vpravo je umístěno datum a běžící čas. Po kliknutí na celou tuto část se vrátíte vždy na úvodní stránku.

2. Úplně vlevo se nachází hlavní menu, které slouží pro hlavní navigaci na stránce. Většinou (až na výjimky jako superadministrátor) je toto menu pro všechny uživatele stejné.

3. Tato část obsahuje jméno, příjmení a login přihlášeného uživatele a je zobrazené vždy. Hned pod tím je místo, kde se zobrazují varovné zprávy v případě potřeby, jako v tomto případě *Projekt byl úspěšně vytvořen.*

4. V této části stránky se zobrazuje hlavní obsah systému. Tento obsah se samozřejmě mění dynamicky a je zobrazen jinak uživatelům a administrátorům či redaktorům.

[4] Pozn. červené čáry se samozřejmě nezobrazují, slouží zde pouze pro zvýraznění hlavních částí stránky.

V tomto případě je seznam projektů zobrazen administrátorem, a jak vidíme, je zobrazena možnost editace projektu, mazání a v dolní části je formulář pro vložení nového. To se samozřejmě uživateli nezobrazí.

5. V pravém sloupci je zobrazen seznam připojených uživatelů a možnost zobrazení uživatelů odpojených. Ti tam nejsou obvykle vidět, protože celý seznam obsahuje asi 50 uživatelů, a byl by zbytečne velmi dlouhý, kdyby se měl zobrazovat vždy. Po kliknutí na jednoho uživatele v seznamu (nezáleží jestli připojeného nebo odpojeného) se zobrazí malé okno, které slouží pro posílání krátkých zpráv mezi uživateli. Nad celým seznamem je zobrazena ikonka obálky, která se rozbliká při přijetí nové zprávy od jiného uživatele. Princip posílání zpráv, přechod uživatelů online/offline a další je podrobněji popsáno v kapitole 3.7.2.

6. Poslední část, pruh úplně dole, obsahuje příchozí zprávy od ostatních uživatlů. Zprávy nejde mazat a může být zobrazeno maximálně 50 zpráv, více zpráv se nearchivuje. Při kliknutí na řádek příchozí zprávy se ihned zobrazí okno pro odpověď odesílateli.

V několika předchozích bodech jsem rozebral hlavní rozložení stránky. V programu mám vytvořenou hlavní třídu `page`, která se skládá z několika metod a atributů. Metody `do_header()`, `do_menu()`, `do_content()` nebo `do_footer()` složí jednotlivé komponenty do výsledné podoby a metoda `display()` zobrazí celou stránku. Při vytvoření nového obsahu stránky vytvářím nové třídy, které zdědí metody a atributy po třídě `page` a přepíšu metodu `do_content()`, která jen změní obsah stránky. Atributy `title` nebo `CSS` obsahují titulek a název souboru kaskádového stylu. Za zmínku stojí ještě atribut `warning` typu string, který může obsahovat varovnou zprávu, třeba takovou, která je vidět na obrázku 3.3.

3.4.2 Volba skinu

Jedna z vlasností uživatelského účtu je změna vzhledu stránky (skinu). Každý je zvyklý pracovat například v jiném barevném rozložení. Někomu vyhovuje více tmavý podklad a někomu světlý. Skiny jsou tvořené pomocí kaskádových souborů v adresáři `styles`, které jsou použity podle toho, který si konkrétní uživatel uloží jako výchozí. Zatím má uživatel k dispozici tři výchozí styly, ale je možné kdykoli přidat další.

Soubory s kaskádovými styly se neukládájí v databázi, ale stačí je pouze uložit do složky `styles` a patřičně pojmenovat. PHP skript automaticky prohledá tento adresář a soubory, jejichž název vyhovuje regulárnímu výrazu `^main[a-z0-9_.]*[.css]$` zobrazí i s popiskem na výběr na stránce. Popisek si soubor nese s sebou a je uložen zakomentovaný na prvním řádku v souboru.

V době odevzdání byly hotové tři grafické návrhy, které jsou součástí práce jako příloha.

3.4.3 Hlídání špatného vstupu formulářů

Uživatel v systému nepotřebuje mnoho formulářů pro ukládání dat. Do příspěvků v diskuzích může vkládat libovolné znaky, jsou povoleny i HTML značky (zobrazí se tak, jak se napíšou). Snad jen u formulářů pro změnu osobních údajů nebo formulář pro první uložení jména, příjmení a dalších údajů při prvním přihlášení uživatele by se mělo hlídat nevhodné vložení znaků.

Z hlediska uživatelského rozhraní jsem dle požadavků napsal JavaScriptovou funkci, která se aktivuje při každé události `onkeyup="check_symbols(this);"` a ihned odstraní

nepotřebné znaky. To je výhodné v tom, že uživatele "neobtěžuji" s varovnými hláškami po odeslání formuláře. Zdrojový kód funkce (zkrácený pro prohlížeče Firefox a Opera):

```
function check_symbols(field)
{
  if (field.setSelectionRange)
  { /* FF, OPERA */
    selStart = field.selectionStart;
    selEnd = field.selectionEnd;
    //zde je regulární výraz povolených znaků
    field.value=field.value.replace(/[^a-záäčçd'éëěíňóöřšť'úůüýž -]/gi, "");
    field.setSelectionRange(selStart, selEnd);
  }
  else if (field.createTextRange)
  { /* IE */
    ...
  }
}
```

3.5 Uživatelský profil a zabezpečení

Jak už jsem psal v kapitole 3.2.1, redakční systém běží na operačním systému Fedora Core a server Apache. Bylo potřeba nějakým způsobem vyřešit správu a přihlašování uživatelů. Když do firmy přijde nový zaměstnanec, je mu automaticky vytvořen účet v Active Directory, účet na přístup k mailu atd... Bylo by tudíž nevhodné nutit uživatele k zapamatování dalšího hesla, jak mi bylo řečeno ve firmě, čím míň pro uživatele k zapamatování, tím líp.

Se zákazníkem jsme se tedy dohodli na autentizaci uživatele na úrovni Apache, kterým jde zaheslovat přímo daný adresář. Zabezpečení může probíhat pomocí direktiv konfiguračního souboru `httpd.conf`, nebo nemáme-li přístup k tomuto souboru (například hosting na veřejném serveru), můžeme vytvořit soubor `.htaccess` v daném zabezpečeném adresáři. Soubor s hesly na firmě obsahuje seznam uživatelů a jejich hesel, která používají pro přihlašování k mailu, takže nemusíme zatěžovat uživatele dalšími hesly. Oba soubory jsou textové a k jejich editaci stačí obyčejný textový editor. Konfiguraci Apache, direktiv a souborů rozepíši v následujících kapitolách.

3.5.1 Konfigurace Apache serveru pro zabezpečení

Nemám sice přístup ke konfiguračním souborům Apache na firemním intranetu, ale vzhledem k tomu, že systém vyvíjím většinou na svém PC (na firemní intranet nemá nikdo z venku přístup), potřeboval jsem nastavit Apache podobným způsobem.

Nejdřív je potřeba textový soubor `.passwd_file` s hesly vytvořit následujícím příkazem:

`htpasswd -c .passwd_file` *uživatel*

Soubor se vytvoří v aktuálním adresáři a nedoporučuje se umísťovat jej do adresáře se soubory webu. Na konci příkazu musíme uvést jméno prvního uživatele, kterého chceme do souboru vložit. Bez toho se soubor vytvořit nepodaří. Po spuštění příkazu jsme požádáni zadat 2× heslo, které se v zašifrované podobě uloží do souboru. Pro další přidání uživatele (soubor je tedy už vytvořen) zadáme podobný příkaz, jen bez parametru `-c` , tedy:

`htpasswd .passwd_file` *uživatel*

Opět jsme nuceni zadat 2× heslo pro ověření a uživatel je do souboru přidán. Pro ukázku takových záznamů zde vložím výpis takového souboru s hesly (Příkaz `cat .passwd_file`). Ve firmě je záznam uživatele přidán do souboru pravděpodobně automaticky pomocí nějakého skriptu, který se používa pro přihlašování k mailu již při vytvoření uživatelského účtu do Active Directory.

```
lukas@localhost:/var/www/hesla/rsdi$ cat .passwd_file
user:ixDdE3RNu2.1E
root:54kFkn0Hib5PQ
user1:du4sjlVzYSbjk
webman:RJ8A9fng7W5Ys
```

Teď když máme vytvořené soubory s uživateli a jejich hesly, potřebujeme zabezpečit samotný adresář a donutit uživatele zadat správný `login` a `heslo`. To se dělá vždy na konkrétní adresář serveru a jeho konfigurace zasahuje rekurzivně všechny podadresáře a soubory, pokud to v konfiguraci nenastavíme jinak. Samotná konfigurace je v souboru `httpd.conf` v adresáři s konfiguračními soubory Apache serveru, většinou ve složce `/etc/apache2/`. Zobrazení mého souboru (Příkaz `cat httpd.conf`) vypadá následovně:

```
lukas@localhost:/etc/apache2$ cat httpd.conf

  <Directory "/var/www/bproject/rsdi">

      AddDefaultCharset UTF-8

      order allow,deny
      allow from all

      AuthType Basic
      AuthName "Prihlaste se prosim"
      AuthUserFile /var/www/hesla/rsdi/.passwd_file
      require valid-user

  </Directory>
```

Pro nás je nejdůležitější atribut direktivy `<Directory> "/var/www/bproject/rsdi"`, což je cesta daného adresáře, který chceme zabezpečit. V případě konfigurace pomocí souboru `.htaccess`, o jehož použití jsem se zmiňoval v kapitole 3.5, musíme direktivu `<Directory>` vynechat. Další direktivy značí například nastavení výchozího kódování, omezení přístupu na IP adresy (v tomto případě žádné) nebo nastavení způsobu autentizace. Důležitá je direktiva `AuthUserFile /var/www/hesla/rsdi/.passwd_file`, kde je odkaz na soubor s uživateli a jejich hesly.

Další a podrobnější konfigurace, jako například omezení na jednotlivého uživatele, IP adresu, členění uživatelů na skupiny atd., je nad rámec této práce a najdeme je například v [5].

3.5.2 Přihlášení do systému

V předchozích kapitolách jsem rozebral konfiguraci a zabezpečení adresáře na serveru. Teď se pokusím popsat samotné přihlašování z hlediska uživatele a další zpracování na serveru.

Jestliže uživatel zadá adresu stránky, která leží v zabezpečeném adresáří, zobrazí se mu přihlašovací formulář, který je na obrázku 3.4. Zobrazený formulář je z prohlížeče Firefox.

Jestliže uživatel zadá správný login a heslo, zobrazí se mu úvodní stránka systému, v opačném případě Apache nepovolí žádný přístup. Nyní mám vyřešené přihlašování na vyšší úrovni a nemusím tedy řešit žádný přihlašovací formulář.

V PHP skriptu nám proměnná `$_SERVER['PHP_AUTH_USER']` vrátí login přihlášeného uživatele. Jak už jsem rozebral v kapitole 3.3, uživatel používá pro komunikaci v systému jméno a příjmení a jiné atributy, které je nutné mít uložené v databázi. Jestliže se uživatel přihlašuje do systému poprvé, přinutíme ho zadat jméno a příjmení, popřípadě vzhled či jiné atributy, které při dalším přihlašování zadávat již znovu nemusí. Dané údaje se načtou z databáze do superglobálních proměnných `$_SESSION` a používají se po celou dobu sezení.

Obrázek 3.4: Přihlašovací formulář

3.5.3 Odhlášení

Odhlášení uživatele neproběhne úplně, ale uživatel se jen označí za odhlášeného. Jelikož uživatelé ve firmě jsou povinni zamknout stanici na úrovni operačního systému, když stanici opustí, nemusím řešit zneužití systému neoprávněnou osobou. Tedy úplné odhlášení, aby prohlížeč žádal znovu heslo a login, proběhne až po zavření prohlížeče. Při nezavření se objeví okno, s úspěšném odhlášení a nabídne zavření prohlížeče (mimo Firefox) nebo opětovné přihlášení, které se provede bez nutnosti znovuzadání hesla a loginu.

Odhlášení je zde tedy z jediného důvodu, aby login zmizel okamžitě ze seznamu online uživatelů. Když uživatel zavře prohlížeč bez kliknutí na tlačítko odhlášení, ze seznamu online uživatelů nezmizí ihned, ale až po nastavené době (cca 15 minut) neaktivity (viz kapitola 3.7.1).

3.5.4 Mazání a obnova uživatelů

Jestliže nějaký zaměstnanec opustí firmu a má v systému účet, administrátor může jeho účet deaktivovat. Fyzicky však v databázi zůstává stále, jen není možné se pod daným loginem přihlásit a běžným uživatelům se deaktivovaný uživatel nezobrazí ani v seznamu offline uživatelů.

Jestliže je uživatel přihlášen a administrátor deaktivuje jeho účet, uživateli je ihned znemožněno v práci jakkoli pokračovat. Administrátor samozřejmě v položce menu `Uživatelé`

vidí i deaktivované účty a může je kdykoliv aktivovat zpět. Fyzicky se z databáze záznamy uživatelů nemažou nikdy.

3.6 Komponenty systému a jejich řešení

V této kapitole tu podrobněji rozepíši jednotlivé komponenty, ze kterých se systém skládá. Popíši funkčnost těchto částí, ale i příklady implementace, které mě dovedli k realizaci celého řešení.

3.6.1 Projektová fóra

Projektová fóra slouží jako diskuze pro řešení jednotlivých projektů. Každé projektové fórum má jednoho redaktora, který může vybrat nové čtenáře nebo odstranit stávající. Pouze čtenáři, redaktor a administrátor mají přístup do fóra. Pokud není uživatel čtenářem některého projektového fóra, je příslušný řádek seznamu neklikací a šedý.

Práva čtenáře v projektových diskuzích:

- Číst jednotlivé příspěvky a reagovat na ně.
- Může připojit maximálně 10 souborů (libovolného typu) k příspěvku jako přílohu.
- Skrývat/odkrývat jednotlivé podstromy v příspěvcích.

Práva redaktora [5] v projektových diskuzích:

- Přidávat a odebírat čtenáře své diskuze.

Práva administrátora[6] v projektových diskuzích:

- Měnit název projektové diskuze.
- Změnit redaktora.
- Smazat prázdnou diskuzi.

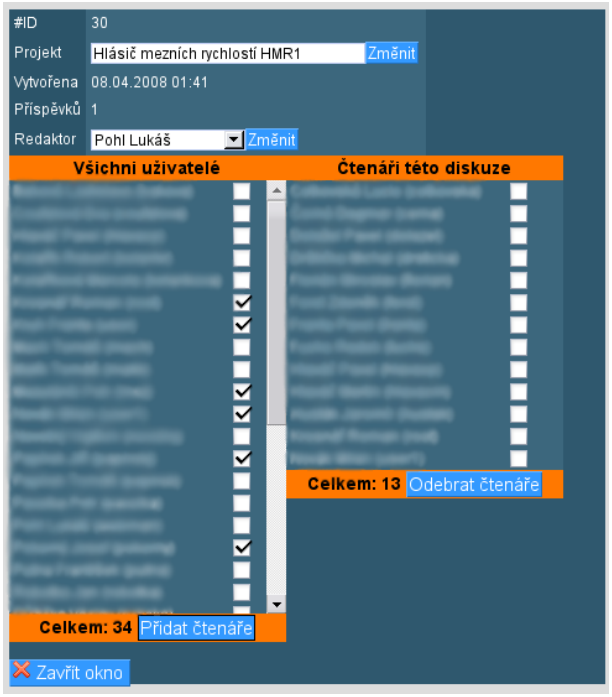
Příspěvky ani diskuze obsahující příspěvky se nedají mazat ani administrátorem. Veškeré vložené příspěvky jsou tedy nevratné a jsou archivovány.

Na obrázku 3.5 můžeme vidět ukázku editačního okna konkrétního projektového fóra, které využívá redaktor pro přidání a odstranění čtenáře. Toto okno může zobrazit pouze administrátor, protože obsahuje komponenty, které umožňují změnu názvu projektového fóra a jeho redaktora. Tyto operace nemůže provádět nikdo jiný než administrátor, tedy ani redaktor.

Příspěvky mají stromovou strukturu, takže jednotlivý příspěvek může obsahovat `N` odpovědí. Seznam příspěvků stromové struktury můžete vidět na obrázku 3.6. Řádek obsahuje pouze předmět, autora a datum uložení, popřípadě sponku, obsahuje-li příspěvek přílohu. Jednotivé podstromy se můžou rozbalovat a sbalovat pomocí tlačítek + a −. Údaj o tom, které příspěvky se mají rozbalit, nese pole `$_SESSION['expanded'][$_GET['expand']]`. Toto pole `$_SESSION['expanded']` je superglobální proměnná, takže zobrazený strom zachová svoji podobu do ukončení celé relace. Viz [3].

V záhlaví podstromu jsou tlačítka pro rozbalení a sbalení všech podstromů, vytvoření nového téma a možnost zaškrtnutí zasílájí upozornění na e-mail při novém příspěvku v daném projektovém fóru.

[5] Vychází z práv čtenáře
[6] Vychází z práv čtenáře a redaktora



Obrázek 3.5: Ukázka editačního okna projektové diskuze

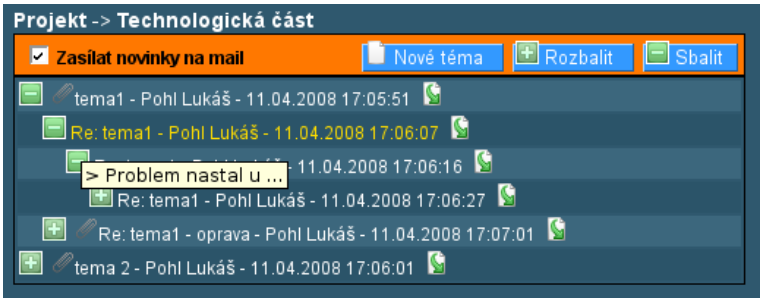
Zobrazovaný podstrom se nejdříve vytvoří pomocí rekurze, kde se v jednotlivých uzlech vytvoří potřebné informace:

```
public function __construct($text, $postid, ... , $id_discussion,$id_message)
 {
   $this->t_postid = $postid; // ID příspěvku
   $this->t_title = $title; // predmet prispevku
   $this->t_poster = $poster; // autor prispevku
   $this->t_posted = $posted; // cas prispevku
   $this->t_children =$children; // obsahuje/neobsahuje potomka
   $this->t_childlist = array(); // pole dalsich potomku
   $this->t_depth = $depth; // hloubka zanoreni
   $this->t_id_discussion = $id_discussion; // ID diskuze
   $this->t_id_message = $id_message; // zprava, ktera je vybrana
   $this->t_text = $text; // samotna zprava
   ...
 }
```

Tyto informace jsou pak dále použity při následném zobrazení stromu, přidají se ikony (+, − . . .), CSS styly, tlačítko odpovědi atd. . .

Okno se zprávou se po vybrání řádku stromu zobrazí nad stromem s textem příspěvku. V textu se zobrazují emotikony, které se nahrazují z textové podoby ” :) ” do podoby obrázku.

Funkce `public function format_links($text,$maxlength)` pak ještě upraví odkazy v textu. Funkce nejprve rozdělí celý text podle mezer na pole slov a poté jednotlivá slova zpracuje. Pokud narazí při zpracování pomocí regulárních výrazů (viz ukázka zdrojového



Obrázek 3.6: Stromová struktura příspěvků

kódu dále) na odkaz, vloží HTML kód tak, aby byl odkaz klikací a následně pole slov složí zpět. Jestli je odkaz delší než nastavený počet znaků, například odkaz na Google maps, jeho délka se zkrátí.

```
  ...
//is the word link ?
if(eregi("http(s)?://[a-zA-Z0-9_\.\-]+",$ilet[$i]))
{
  //we check if display link is more than 15 chars longer
  $display_link=$ilet[$i];
  if(strlen($display_link)>$maxlength)
  $display_link=$this->short($display_link,$maxlength);

  //we add HTML code
  $ilet[$i]= str_replace($ilet[$i],"<a class='odkaz_zvyrazneny'
    href='".$ilet[$i]."' target='_blank'>".$display_link."</a>",$ilet[$i]);
}
  ...
return $text;
```

3.6.2 Systém dokumentace

Systém dokumentace je souborem řídicích dokumentů, vnitrofiremních předpisů, číselníků, předloh provozní a projektové dokumentace a popisů technologických postupů, jehož aplikace má vést ke zlepšení komunikace uvnitř společnosti, zefektivnění výroby a zvýšení jakosti.

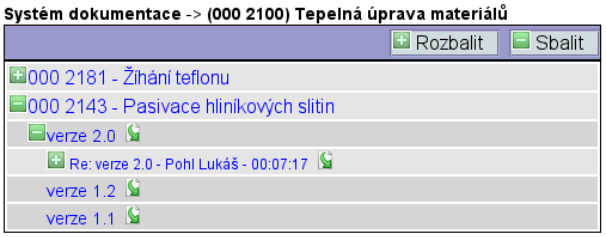
Až na pár rozdílů je struktura systému dokumentace velmi podobná jako projektové diskuze. Dělí se na několik oborů dokumentace a může obsahovat příspěvky ve stromové struktuře. Rozdíl je v tom, že příspěvky první úrovně označují předpis dokumentace. Mají větší velikost, neobsahují žádný text příspěvku a ve stromu se zobrazí pouze předmět. Po kliknutí na předpis (příspěvek první úrovně) se vždy rozbalí jeho podstrom.

Příspěvky druhé úrovně představují verze jednotlivých předpisů, které obsahují minimálně jeden soubor jako přílohu. Po kliknutí na řádek s verzí ve stromě příspěvků, se zobrazí pouze okno s textem příspěvku, nikoliv strom s příspěvky pod ním. K tomu využívám rekurzivní funkci pro výpočet hloubky zanoření:

```
private function get_depth($message_id,$n=1)
```

```
{
  global $dbase;
  $result=$dbase->do_query("select * from doc_message where id='$message_id'");
  $row=$result->fetch_object();
  $result->free();
  if(!$row->parent) return $n;
  else return $this->get_depth($row->parent,++$n);
}
```

Příspěvky od 3. do `N` té hloubky zanoření jsou komentáře uživatelů na jednotlivé verze. Z hlediska práv systém dokumentace neobsahuje žádné vybrané čtenáře. Uživatelé mají přístup pro čtení všech dokumentací, ale odpovídat na příspěvky můžou až od 3. hloubky zanoření příspěvků, bez možnosti připojení souborů. Pouze administrátoři mohou vytvářet nové dokumentace, předpisy a jejich verze, ke kterým můžou vkládat soubory (příspěvkům 2. úrovně).



Obrázek 3.7: Stromová struktura příspěvků v systému dokumentace z pohledu uživatele

3.7 Použití AJAXu

V kapitole 2.6.2 jsem se zaměřil na princip technologie AJAX z teoretického hlediska. Nyní popíšu, jak jsem AJAX použil při vývoji tohoto systému.

3.7.1 Seznam uživatelů

Při návrhu uživatelského rozhraní jsem zvolil pravý sloupec stránky pro zobrazení seznamu připojených uživatelů (viz. kapitola 3.4.1, Návrh uživatelského rozhraní). Odpojení uživatelé nejsou implicitně zobrazení a zobrazí se v případě potřeby až po kliknutí na zaškrtávací tlačítko. Tento seznam je příliš dlouhý, aby byl zbytečně zobrazen neustále. Připojení uživatelé jsou zobrazeni vždy.

Při odpojení/připojení uživatele se jeho jméno v seznamu mění automaticky, bez jediného obnovení stránky. K tomu slouží JavaScriptová funkce `get_online_users()`. Klient posílá pomocí objektu AJAX `XMLHttpRequest()` každých 5 sekund dotaz na server o aktuální seznam uživatelů. Na serveru skript `get_online_users.php` zpracuje požadavek a pomocí objektu `$dom = new DOMDocument('1.0','UTF-8');` sestaví odpověď ve formátu XML (viz obrázek 3.8). Klient tato XML data přijme a pomocí DOM vytvoří HTML kód s uživateli a výsledek zobrazí na stránce v případě, že došlo ke změně. Změnu skript zjišťuje podle aktuálních hodnot proměnných v `COOKIES`.

```
<users>
  <user name="Pohl Lukáš" id="55" status="online" login="webman"/>
  <user name="[illegible]" id="4" status="offline" login="[illegible]"/>
  <user name="[illegible]" id="5" status="offline" login="[illegible]"/>
  <user name="[illegible]" id="6" status="offline" login="[illegible]"/>
  <user name="[illegible]" id="7" status="offline" login="[illegible]"/>
  <user name="[illegible]" id="8" status="offline" login="[illegible]"/>
```

Obrázek 3.8: XML odpověď pro seznam uživatelů

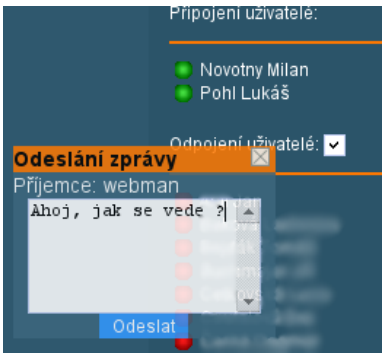
3.7.2 Odeslání zprávy uživateli

Pro odeslání zprávy jednotlivým uživatelům slouží malé okno, které je na obrázku 3.9. Ono se to jen jako okno tváří, ale ve skutečnosti je to HTML element `<div>`, který se pozicuje relativně od aktuální pozice myši. Tato varianta je daleko rychlejší, než varianta s vyskakovacím (Pop-up) oknem, která může uživatele jen odradit.

Po kliknutí na odeslání se data pomocí AJAX objektu `XMLHttpRequest()` odešlou na server, kde se uloží do databáze. Od serveru ještě přijde odpověď ve formátu XML (`<response>`*odpověď*`</response>`). Odpověď od serveru se zobrazí na 3 vteřiny jako varování pro uživatele.

Možné druhy odpovědi:

- *Zpráva odeslána.*
- *Nelze poslat zprávu sám sobě.*
- *Nelze poslat prázdnou zprávu.*
- *Uživatel neexistuje.*
- *Chyba při odeslání.*
- *CHYBA*



Obrázek 3.9: Okno pro odeslání zprávy uživateli

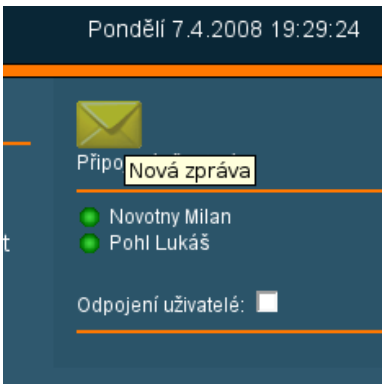
```
−<messages>
 −<message id="8">
    <sender id="3">Novák Franta (user1)</sender>
    <text>Ahoooj jsi tam ? dlouho jsem te nevidel ...</text>
    <time>14:43:27</time>
  </message>
 −<message id="7">
    <sender id="2">Novotny Milan (user)</sender>
   −<text>
      Mrkni se sem, třeba ti to pomůže http://cz.php.net/strings
    </text>
    <time>14:41:03</time>
  </message>
 −<message id="6">
```

Obrázek 3.10: Formát XML odpovědi pro získání příchozích zpráv

3.7.3 Přijetí zprávy od uživatelů

Příchozí zprávy od uživatelů se na stránce opět objevují bez jediného načtení nebo obnovení stránky téměř ihned po odeslání odesílatelem. K tomu slouží JavaScriptová funkce `get_messages()`, která pomocí AJAX objektu `XMLHttpRequest()` posílá každých 5 vteřin požadavek pro stažení aktuálních zpráv. Na serveru skript `get_messages.php` zpracuje požadavek a sestaví odpověď.

Tu sestaví pomocí objektu `$dom = new DOMDocument('1.0','UTF-8');` a výsledek ve formátu XML pošle zpět. Jak vypadají sestavená data ve formátu XML, vidíme na obrázku 3.10).



Obrázek 3.11: Rozblikaná ikona při nově příchozí zprávě

Tato data jsou seřazená podle času odeslání jednotlivých příspěvků, takže klient po přijetí odpovědi porovná první záznam v XML s datem uloženým v COOKIES z předchozí odpovědi. Jestliže je datum rozdílné, v novém požadavku přišla nová zpráva. Uživatele na to upozorní blikajicím titulkem stránky asi na 10 sekund, takže si může nově příchozí zprávy všimnout, i když má prohlížeč minimalizován. Uživatel by měl na první pohled vidět, že mu přišla nová zpráva, i když se vrátí k PC po delší době. K tomu slouží neustále blikající ikonka obálky nad seznamem uživatelů (viz obrázek 3.11). Ikona bliká, dokud uživatel někam

neklikne. K blikání byla použita vlastnost CSS `opacity`, tedy průhlednost. Příchozí zprávy jsou zobrazené ve spodní části rozhraní. Pro ukázku zde uvedu Javascriptovou funkci pro neustálé rozblikání obrázku. Blikání se zastaví příkazem `clearTimeout(TimeOpacity);`.

```
// objImage - obrazek ktery rozblikavam
// inc - jestli je true tak zvetsuju pruhlednost
//       jinak ji zmensuju
  function opacity(objImage,inc)
  {
    // průhlednost pro ne-IE prohlížeče
    if (objImage.style.opacity != null)
      objImage.style.opacity = ActOpacity/100;

    // průhlednost pro IE
    else if (objImage.style.filter != null)
      objImage.style.filter = "alpha(opacity="+ActOpacity+")";

    // konstantou je možné ovlivnit v kolika stupních se obrázek objeví
    if (inc) ActOpacity += 5;
      else ActOpacity -= 5;

    ImgOpacity = objImage;
    if (inc)//zvetsuju
    {
      if(ActOpacity<=80)
        TimeOpacity=setTimeout("opacity(ImgOpacity,true)",20);
      else
        TimeOpacity=setTimeout("opacity(ImgOpacity,false)",20);
    } else
    { //zmensuju
      if(ActOpacity>=20)
        TimeOpacity=setTimeout("opacity(ImgOpacity,false)",20);
      else
        TimeOpacity=setTimeout("opacity(ImgOpacity,true)",20);
    }
  }
```

Shrnutí

V této kapitole jsem se snažil popsat to nejdůležitější a nejpodstatnější, s čím jsem se při řešení celého systému potkal. Popsal jsem a zdůvodnil zvolené řešení. Realizace celého systému probíhala přesně podle požadavků zadavatele, které zasahovaly až do těch nejmenších detailů. Z tohoto důvodu jsem také musel velmi často některé části měnit a ladit již za ostrého provozu.

Kapitola 4

Závěr

V době odevzdání této práce již systém zkušebně běží na firemním intranetu a denně s ním pracuje několik lidí. Nějaký čas však bude trvat, než si všichni uživatelé (zaměstnanci firmy) zvyknou systém každodenně používat.

Nyní jsou dokončené nejdůležitější části, které jsou důležité pro zlepšení komunikace, nejen při vývojích projektů. Tím však vývoj systému nekončí. Zadávány jsou další úkoly, o které by se měl systém v nejbližší době rozšířit. Systém bude obsahovat rozšířenější nástěnku, která nyní obsahuje pouze telefonní seznam a veřejné fórum. Přidat se mohou například fotogalerie uživatelů a operace s nimi spojené, úřední deska, která obsahuje důležitá sdělení od vedení firmy, aktuální jídelní lístky, práce s rozvrhy atd. ...

Během řešení celé práce jsem získal mnoho zkušeností, nejen s použitými technologiemi a postupy. Největší zkušeností však bylo, oproti běžným školním projektům, řešit a vyvíjet software na zakázku. Samozřejmostí byla častá komunikace se zákazníkem, kdy pro mne bylo velmi důležité vyvíjet program přesně podle daných specifikací. Několikrát se mně také stalo, že jsem zadání nesprávně vyhodnotil a musel jsem výsledek, mnohdy od základů, pracně předělávat.

I po splnění zadání však nebyla práce úplně dokončena, protože jsem musel upravovat drobnější problémy, chyby a nedostatky, na které se přišlo až během zkušebního provozu.

Pevně doufám a věřím, že aplikace splnila účel, pro který byla navržená a najde své uplnatnění.

Literatura

[1] Jiří Kosek. Serverové skriptovací technologie. [online], 2007. http://www.kosek.cz.

[2] Filip Koval. Ajax 1.dil. [online]. http://www.owebu.cz/obecne/vypis.php?clanek=1009.

[3] Laura Thomson Luke Welling. *PHP a MySQL, Rozvoj webových aplikací.* Computer Press, 2003. ISBN 80-86497-83-6.

[4] Petr Paleta. *Co programátory ve škole neučí, aneb Softwarové inženýrství v reálné praxi.* Computer Press, 2003. ISBN 80-251-0073-1.

[5] Vlastimil Pošmura. *Apache, Příručka správce WWW serveru.* Computer Press, 2002. ISBN 80-7226-696-9.

[6] John Resig. *JavaScript a Ajax, Moderní programování webových aplikací.* Computer Press, 2007. ISBN 978-80-251-1824-5.

[7] WWW stránky. Phpbb - redakční systém. [online]. http://www.phpbb.com/.

[8] Jiří Techet Tomáš Hruška, Jan Krouhlík. *Internetové aplikace (WAP) 3., Studijní opora.* FIT VUT Brno, Únor 2007.

[9] Michal Altair Valášek. Představení internet information services (iis) 7.0. [online]. http://www.aspnet.cz/.

[10] O Webu. Automatická dokumentace v php. [online]. http://www.owebu.cz/php/vypis.php?clanek=1306.

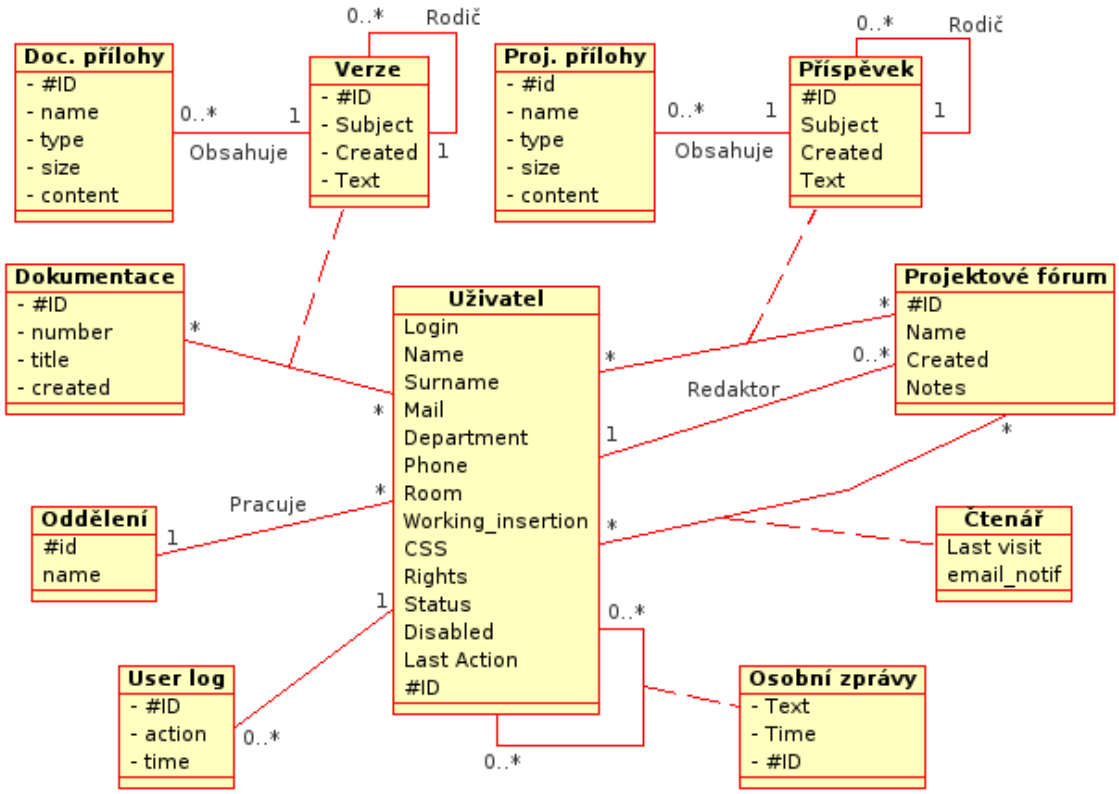
[11] Wikipedie. Databáze. [online]. http://cs.wikipedia.org/wiki/Databáze.

[12] Wikipedie. Multipurpose internet mail extensions. [online]. http://cs.wikipedia.org/wiki/MIME.

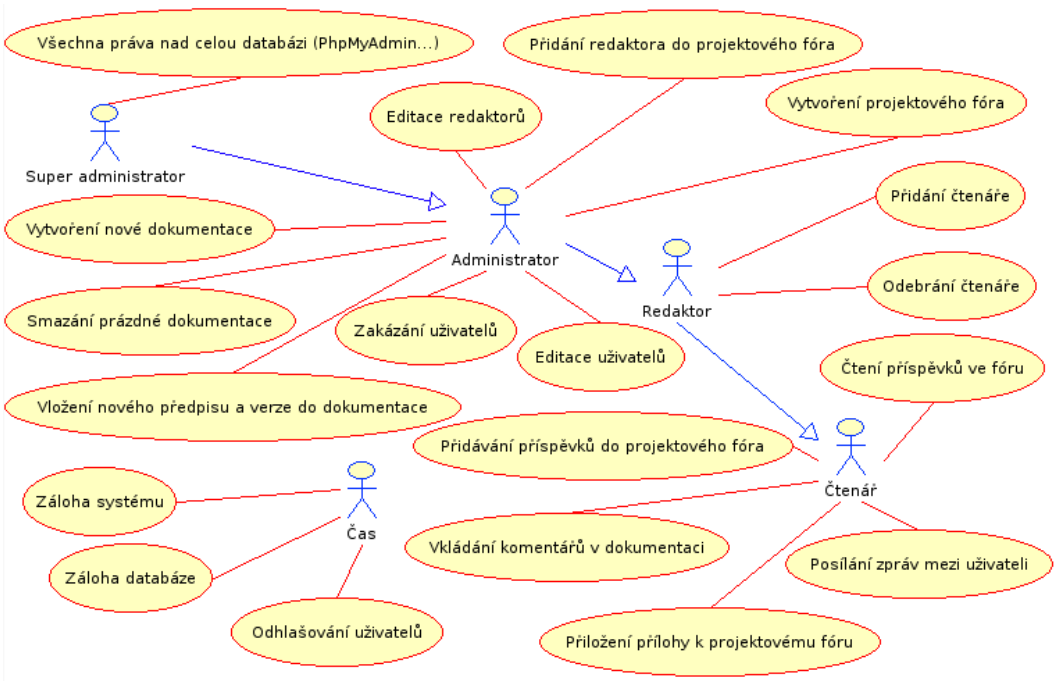
Seznam příloh

1. ER-diagram
2. Use Case diagram
3. Ukázky vzhledů – tmavý vzhled (výchozí)
4. Ukázky vzhledů – světlý vzhled
5. Ukázky vzhledů – Sahara vzhled

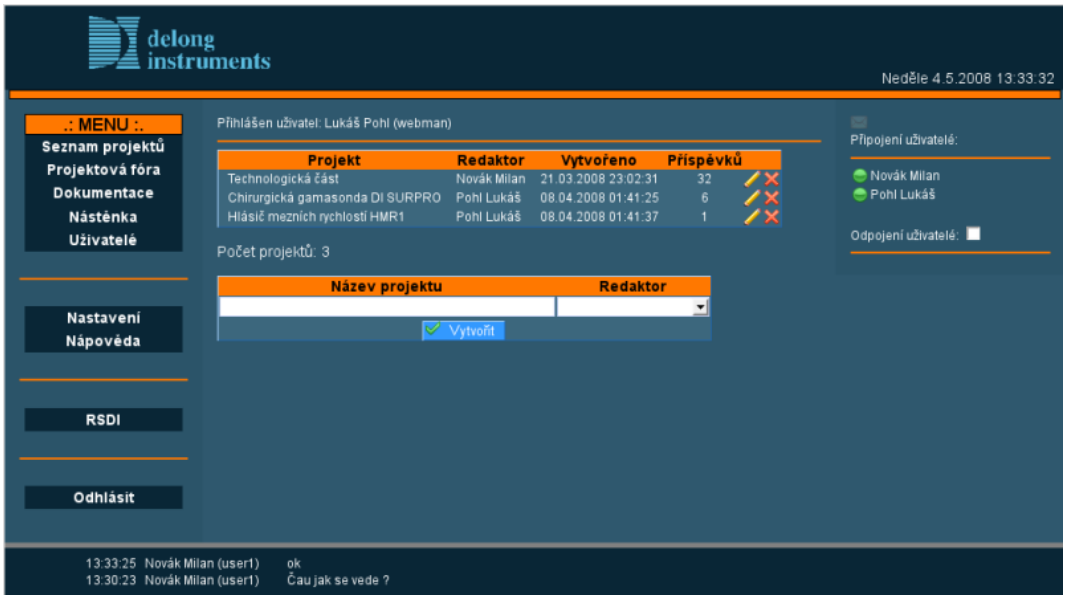
Příloha 1 – ER-diagram



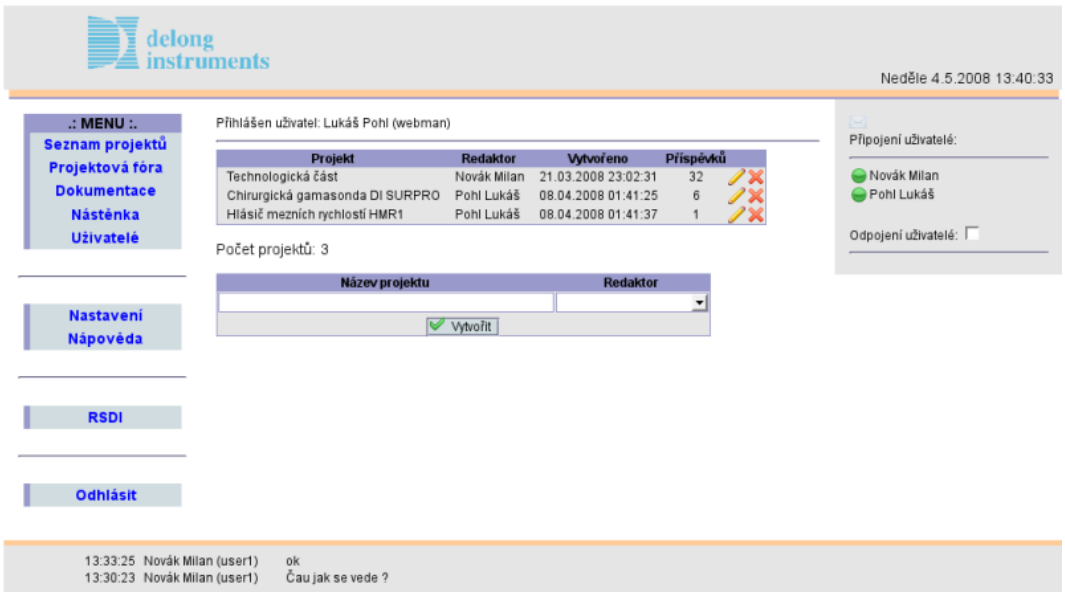
Příloha 2 – Use Case diagram



Příloha 3 – Tmavý vzhled



Příloha 4 – Světlý vzhled



Příloha 5 – Sahara vzhled