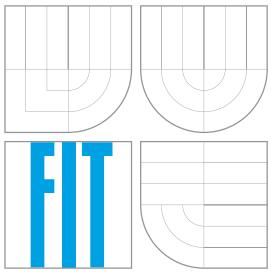
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

IMPLEMENTACE DISKUSNÍ SKUPINY PRO STUDENTY

IMPLEMENTATION OF DISCUSSION GROUP FOR STUDENTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL ŠEBEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ROMAN LUKÁŠ, Ph.D.

BRNO 2007

Zadání bakalářské práce/5115/2006/xsebek00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2006/2007

Zadání bakalářské práce

Řešitel: **Šebek Michal**
Obor: Informační technologie
Téma: **Implementace diskusní skupiny pro studenty**
Kategorie: Web

Pokyny:

1. Seznamte se s jazyky a prostředky pro tvorbu webových informačních systémů (XHTML, CSS, PHP, Javascript, MySQL).
2. Proveďte obecně analýzu požadavků pro diskusní skupinu pro jistou skupinu studentů. (např. pro konkrétní třídu na gymnáziu, pro přednáškovou skupinu na VŠ atd.)
3. Navrhněte vhodnou strukturu systému. Složitost vašeho návrhu konzultujte s vedoucím BP.
4. Daný systém implementujte tak, aby mohl být použitelný v praxi.
5. Zhodnoťte dosažené výsledky, porovnejte Váš systém s existujícími systémy, navrhněte další možné rozšíření do budoucna.

Literatura:

- Kosek, J.: PHP - Tvorba interaktivních internetových aplikací
- Cyroň, M.: CSS-kaskádové styly

Při obhajobě semestrální části projektu je požadováno:

- Body 1), 2) a 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 paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Lukáš Roman, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 15. května 2007

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: **Michal Šebek**
Id studenta: 84079
Bytem: Pavlovova 1514, 592 31 Nové Město na Moravě
Narozen: 16. 07. 1985, Nové Město na Moravě
(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: Implementace diskusní skupiny pro studenty
Vedoucí/školitel VŠKP: Lukáš Roman, Ing., Ph.D.
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

Tato práce se zabývá analýzou a vývojem informačního systému – diskusní skupiny určené pro používání studenty technické školy. Práce je zaměřena především na shrnutí poznatků při implementaci speciálních funkcí jakožto dynamického vkládání obrázků, tabulek a vzorců ve formátu systému LaTeX do komentářů. Rovněž diskutuje možnost použití umělé inteligence jako nástroje pro vyhledávání vulgarizmů v textu komentářů. Implementace systému proběhla s použitím jazyků PHP, MySQL, XHTML, CSS a Javascript.

Klíčová slova

diskusní skupina pro studenty, informační systém, internetové fórum, umělá inteligence, vkládání obrázků a tabulek, PHP, MySQL, XHTML, Javascript

Abstract

This thesis discusses analysis and development of information system, such as discussion group for students of technical school. It is focused to summarizing facts about implementation of special functions such as inserting pictures, tables and LaTeX math theorems dynamicly into comments. Also thesis discusses using artificial intelligence for vulgarisms detection in comments. The system was implemented in languages PHP, MySQL, XHTML, CSS and JavaScript.

Keywords

discussion group for students, information system, internet forum, artificial intelligence, pictures and tables insertion, PHP, XHTML, MySQL, Javascript

Citace

Michal Šebek: Implementace diskusní skupiny pro studenty, bakalářská práce, Brno, FIT VUT v Brně, 2007

Implementace diskusní skupiny pro studenty

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Romana Lukáše, Ph.D.

. . . . . .
Michal Šebek
15. května 2007

Poděkování

Na tomto místě bych rád poděkoval Ing. Romanovi Lukášovi Ph.D. za poskytnutí cenných rad a ochoty při řešení problémů při tvorbě této práce a studentům třetího ročníku za pomoc při beta-testování.

Obsah

Kapitola 1

Úvod

Jedním ze základů lidského poznání je touha po komunikaci. Ta se v průběhu vývoje lidstva měnila od gestikulace, řeči, až po zaznamenávání myšlenek jejich zápisem v podobě textu. Ten nakonec ukázal své nesporné výhody, tvrzení nebyla při opakované interpretaci měněna a text v psané podobě byl snadno dostupný. Možná i proto je v současné době v každé skupině lidí snaha o vytváření nějakých písemných komunikačních kanálů, kterými by byla možnost názory na daný problém diskutovat.

Skupinou lidí, ve které je komunikace ke vzájemnému vzdělávání doslova nutná, jsou rozhodně i studenti a snadnou cestou, jak rychle jejich nápady a názory šířit, nám poskytuje Internet. Proto vznikla myšlenka realizace univerzálního diskusního fóra pro určitou skupinu studentů, o jehož realizaci pojednává tato bakalářská práce.

Dnes používané technologie pro vývoj aplikací v prostředí Internetu jsou představeny v **druhé kapitole**. Nejdříve je zde nastíněna obecná struktura takto realizovaného systému a postupně jsou charakterizovány jednotlivé použité technologie PHP, MySQL, XHTML, kaskádové styly, RSS a JavaScript.

Stručná charakteristika diskusních skupin je v **kapitole třetí**. Zaměřuje se na historii, dříve používaná řešení a na řešení v podobě současných diskusních skupin.

Jelikož je systém vyvíjen, aby byl skutečně používán studenty technické školy – *Fakulty informačních technologií*, tak jsou ve **čtvrté kapitole** uvedeny jejich vyslovené požadavky na takové fórum. Tyto požadavky jsou zanalyzovány a roztříděny do modulů výsledného systému.

V páté kapitole je na základé požadavků navrhnuta struktura diskusní skupiny a ta je namodelována pomocí nástrojů pro konceptuální modelování UML. Je zde obsažen use-case diagram, ER diagram a ukázka převodu ERD na fyzické schéma databáze. Zároveň jsem se zde zaměřil na objasnění některých méně obvyklých situací řešených v rámci konceptuálního návrhu.

V kapitole šesté je čtenář podrobně seznámen s implementační fází systému. Podrobně jsou zde rozvedeny nestandardní implementované funkce jako dynamické vkládání obrázků, tabulek a matematických vzorců. Rovněž je zde popsán algoritmus, na jehož základě probíhá autocenzura komentářů.

Kapitola sedmá se podrobně zabývá testováním implementovaného systému a jeho uvedením do reálného provozu. Jsou zde nastíněny některé optimalizační problémy, které při testovacím provozu nastaly, a realizace jejich řešení. Zde se čtenář seznámí i s postupem instalace systému na server a spuštěním diskusní skupiny.

Srovnání implementovaného systému s existující skupinou a diskuze možného dalšího rozvoje je **v osmé kapitole**.

V závěrečné kapitole jsou shrnuty výsledky práce na projektu a jeho přínos do oblasti diskuzních skupin jakožto informačních systémů v podobě nových funkcí.

Kapitola 2

Technologie pro vývoj informačních systémů

U informačních systémů pracujících v prostředí internetu rozlišujeme jejich serverovou a klientskou část – obrázek 2.1. Jako serverovou chápeme část aplikace, která je na straně serveru interpretována a zajišťuje dynamické generování dokumentů obvykle na základě komunikace s určitým datovým skladem. Takovou často používanou kombinací je skriptovací jazyk PHP, který využívá pro uložení dat databázový server MySQL. Výsledkem běhu skriptu PHP je nejčastěji dokument ve formátu, který klientská strana umí zobrazit. Pokud chceme, aby dokument byl zobrazen v internetovém prohlížeči, volíme nějaký jazyk z rodiny SGML, například dnes již standardně HTML nebo nejlépe XHTML, které výslednou kompatibilitu s prohlížeči klienta ještě více rozšiřuje.

Abychom ale snížili vytíženost webového serveru často opakovanými dotazy, které mají obvykle charakter zobrazení nějakého potvrzovacího okna, nebo abychom zvětšili komfort při práci s aplikací, je vhodné doplnit aplikaci o kód, který bude prováděn na straně klienta. K tomuto účelu je vhodné využití jazyku JavaScript.

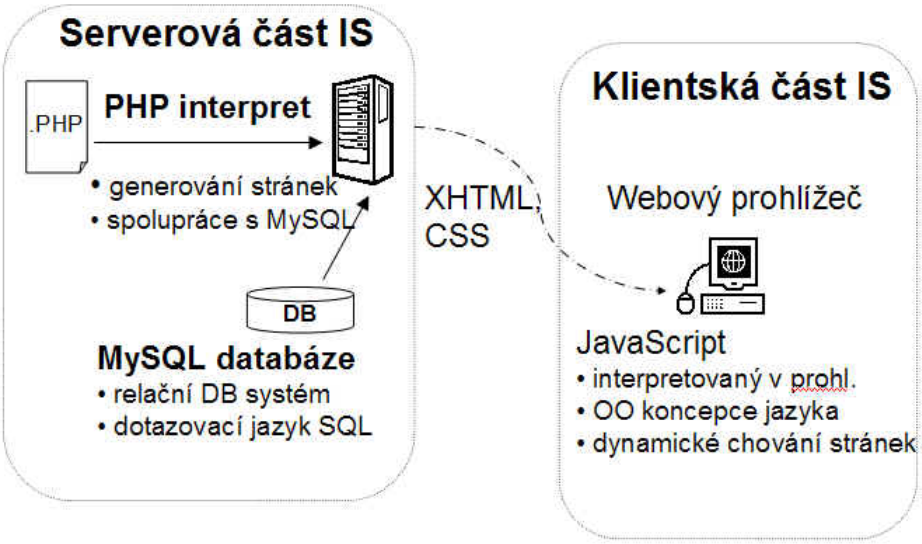
V následujících podkapitolách představím podrobněji zmíněné technologie a jejich využití v informačním systému.

2.1 Skriptovací jazyk PHP

Pro generování dynamických stránek na straně serveru je výhodné využití jazyka PHP. Mezi jeho výhody patří především jeho dostupnost, protože je instalován na většině webových serverů – zřejmě i proto, že je open-source. Kód aplikace v jazyce PHP je vkládán do speciálních značek přímo v kódu (X)HTML. Tento úsek kódu je následně zpracován interpretem a nahrazen jeho výstupem. To již nejčastěji bývá opět nějaký HTML dokument. Mezi základní vlastnosti jazyka patří, že je interpretovaný, procedurální, podporuje i objektový model, proměnné jsou typované implicitně a celý jazyk je case-sensitive.

Počátky projektu Personal Home Page (zkráceně PHP) sahají do roku 1994, kdy Rasmus Lerdorf vytvořil sadu skriptů pro evidenci přístupů k jeho stránkám tehdy ještě v jazyce Perl. Tuto sadu nazval Personal Home Page Tools. Protože ale spouštění skriptu neúměrně vytěžovalo webový server, tak systém přepsal do jazyka C.

Zároveň tehdy pracoval na programu, který by umožnil začlenit do stránek formuláře, komunikovat s databází jazykem SQL a zobrazovat výsledky těchto dotazů. Tento program se jmenoval Form Interpreter. Nakonec oba tyto projekty spojil a vznika Personal Home Page / Forms Interpreter (PHP/FI), které dosáhlo celosvětové popularity. Obsahovalo již součásti nynější funkcionali-



Obrázek 2.1: Struktura infornačních systémů

ty systému PHP. Jednalo se o programovací jazyk, jehož kód se vkládal přímo do HTML stránek. Syntax byla dost podobná jazyku Perl, ale jednodušší. Druhá přepsaná implementace PHP/FI 2.0 doznala už obrovské světové popularity.

Verze PHP 3.0 z roku 1997 je již základem současné verze jazyka. Jejími autory jsou Andi Gutmans a Zeev Suraski. Silnou stránkou této verze bylo poskytnutí velkého množství rozšíření, které umožňovaly spolupráci s mnoha databázovými systémy. V syntaxi jazyka se objevily objektově orientované rysy a celá syntax se stala konzistentnější. Celý projekt byl zkrácen na PHP, což znamená *PHP: Hypertext Preprocessor*.

V současné době je nejnovější verzí PHP5 z roku 2004, které podporuje plně objektový model a bylo doplněno mnoho funkcí do řady PHP modulů. Více lze najít na [1] a v [4].

2.2 MySQL

Pokud potřebujeme na straně serveru uchovávat trvale nějaká data, je vhodné využít k tomuto účelu databázový server. Mezi používané patří především relační databázové servery. To prakticky znamená, že záznamy jsou uchovávány jako řádky – uspořádané n-tice (tj. relace) v tabulce. V dnešní době se pro práci s relačními databázovými systémy používá dotazovací jazyk SQL, který byl standardizován v roce 1992. Příkazy lze rozdělit do několika kategorií, mezi které patří především skupiny:

- DDL (Data Definition Language) – pro manipulaci s daty
- DML (Data Manipulation Language) – pro manipulaci se schématem databáze

Pro realizaci projektu byl vybrán databázový server MySQL. Ten je vhodný pro svou vynikající spolupráci s PHP a zároveň opět pro jeho rozšířenost na straně serverů. Mezi jeho nesporné výhody patří i vysoká stabilita a také fakt, že je šířen jako open-source. Vývoj započal v roce 1996 a nyní je ve své verzi MySQL 5 pro menší aplikace srovnatelný s profesionálními servery.

2.3 Značkovací jazyky HTML, XHTML a kaskádové styly

Výsledkem práce serverové části informačního systému je dokument, který je přenesen na stranu klienta. Pokud chceme, aby byl dokument zobrazitelný ve webovém prohlížeči na počítači uživatele, použijeme k formátování dokumentu jazyk HTML (*HyperText Markup Language*) nebo XHTML (*Extensible HyperText Markup Language*, bude zmíněno dále). Tento jazyk pochází z rodiny jazyků SGML (*Standard Generalized Markup Language*), která popisuje značkovací jazyky určené především pro elektronické platformě nezávislé publikování dokumentů. Jazyk HTML se z něj vyčlenil specifikací konkrétních značek pro použití v oblasti WWW, kterým je přiřazena přesná sémantika (ta u univerzálního SGML není). Původním autorem je Tim Berners-Lee. První specifikace se dočkalo v roce 1995 v oficiální verzi HTML 2.0 podle RFC1866. Standardizaci HTML nyní zastřešuje W3C (*World Wide Web consortium*, více na [8]) a poslední verzí je HTML 4.01.

Celá syntax HTML je ovšem příliš robusní. Implementace analyzátorů začaly být pro speciální platformy (kapesní počítače, apod.) obtížné. Navíc se začala projevovat snaha do dokumentů HTML vkládat části kódu ve stále populárnějším formátu pro popis dokumentů XML [2] a naopak, kde už se projevovala nekompatibilita. Proto vzniká jako nástupce HTML 4.0 standard XTHML 1.0, který vypustil řadu značek, má striktnější syntax (case-sensitive, atributy je třeba uvádět vždy v uvozovkách, každá značka musí být uzavřena, atd.). Zatím poslední standardizovanou verzí je XHTML 1.1 a vyvíjena je verze XHTML 2.0.

Zpočátku byla snaha vystihnout formátování stránek přímo značkami v HTML. Vznikla proto sada značek, které přímo určovaly vzhled textu v dokumentu. Postupně ovšem autoři vyžadovali větší kontrolu nad vizuální stránkou obsahu a vzniká standard CSS 1 (*Cascading Style Sheets*). Ten dává možnosti pro definici základního vzhledu elementů na stránce a od verze 2 umožňuje zcela oddělit formátování vzhledu dokumentu od jeho obsahu. Více v [3].

Výhoda tohoto přístupu se projevuje ve snadnější modifikovatelnosti celého vzhledu všech dokumentů, které jeden stylopis používají, a v možnosti přizpůsobit formátování konkrétnímu výstupnímu zařízení (obrazovka, tiskárna, zařízení pro nevidomé, . . .). Standardizaci provádí jak u HTML a XTHML organizace W3C a nyní je připravováno vydání standardu pro CSS 3.

2.4 Technologie RSS

Samotná zkratka RSS je vykládána několika způsoby. Jeden z nich je *Really Simple Syndication*. Myšlenka formátu RSS, který by shrnoval novinky z webu, sahá až do roku 1999. Byla navržena firmou *Netscape*. V podstatě se jedná o XML dokument, který má definovanou sémantiku značek. Dříve byl formát využíván pro předávání aktualit mezi servery. Teprve později dochází k tomu, že je formát stahován *čtečkami* uživatelů. Položka v RSS se skládá z několika prvků. Lze přenášet pouze název a odkaz do webu, nebo i ucelený text s dodatečnými informacemi o čase publikování, či obrázek.

2.5 Klientský JavaScript

O klientském JavaScriptu hovoříme v případě, že jde o analyzátor jazyka JavaScript, který je integrován do prohlížeče klienta. Původním autorem je Brendan Eich, který jej pod názvem *Mocha*, později *LiveScript*, integroval do prohlížeče Netscape Navigator. Názvu *JavaScript* doznal až do něj byla implementována podpora technologie Java. Mezitím se firma Microsoft myšlenkou skriptování na straně klienta také inspirovala a do prohlížeče Internet Explorer integruje tzv. *JScript* a později *VBScript*.

Standardizací prochází až v roce 1997 asociací ECMA (*Europen Computer Manufacturers Association*) a o rok pozdější i společností ISO. Je pravděpodobné, že právě dlouhá doba ke standardizaci je příčinou toho, že každý prohlížeč podporuje JavaScript v mírně odlišné syntaxi, což brání jeho spolehlivému užívání. Svou syntaxí připomíná jazyky C a C++, mezi jeho základní rysy patří:

- interpretován na straně klienta
- case-sensitive
- objektově orientovaný
- v klientské verzi umí zasahovat do modelu DOM (*Document Object Model*, [6])
- nelze se spoléhat na jeho provedení

Poslední bod je pro tvůrce stránek dost kritický, neboť pokud použijí JavaScript, nemohou se nikdy spolehnout na fakt, že bude vykonán. Prezentace proto musí vždy zachovávat základní funkčnost i s absencí tohoto jazyka. Například mu lze svěřit kontrolu vyplnění formuláře před odesláním, ale serverová strana musí provést kontrolu opětovně. Javascript pouze odlehčí zátěž serveru před chybnými formuláři z prohlížečů, kde je zapnutý.

Kapitola 3

Diskusní skupiny

Diskusní skupiny, občas také označované jako internetová fóra [5], zaznamenaly rozvoj kolem roku 1995. Postupně převzaly funkčnost newsových skupin (Usenet), které sahají ještě o několik let nazpět.

3.1 Usenet, NNTP

Usenet využívá pro přenos zpráv směrem k uživateli protokol NNTP nebo UUCP. Zprávy jsou uložené centrálně na news serveru. Články jsou tříděny do tzv. konferencí. Ty mají hiearchickou strukturu (např. vutbr.news.talk). Kromě zpřístupňování musí server ještě zajistit uložení zprávy odeslané na server pomocí protokolu pro přenos zpráv (nejčastěji e-mailových) SMTP. Tyto diskuse mohly být moderované i nemoderované a přístup mohl být povolen všem osobám (veřejné), nebo omezen na lokální sítě (privátní). Za nevýhodu by se ale dalo považovat, že čtenář musí využít pro stažení příspěvků nějakého e-mailového klienta podporujícího NNTP, který nemusí být vždy dostupný.

3.2 Internetová fóra

Naproti tomu internetové fórum je v podstatě internetová stránka obsahující fóra, vlákna a jednotlivé komentáře. Fórem v tomto významu rozumíme kontejner založený administrátory, ve kterém jsou teprve vkládána uživateli vlákna. (Pozn. pro větší přehlednost budu používat pro označení fór termínu *kategorie*).

Podle přístupu uživatelů můžeme fóra dělit na

- vyžadující přihlášení uživatelů
- anonymní.

První skupina vyžaduje, aby se uživatelé, kteří chtějí na fóru číst, či na něm přispívat, registrovali a při přístupu přihlásili. Pro ověření registrace se často využívá proces ověření existence e-mailové adresy. Uživatelé o sobě publikují vybrané osobní údaje často doplněné o ikonku (zvanou *avatar*). Anonymní přístup na fórum a práva nepřihlášených uživatelů je na zvážení administrátorů.

Existuje už i řada softwarových řešení diskusních skupin, které jsou naprogramované v různých jazycích – PHP, ASP, Java, apod. Mezi velice známé kompletní řešení diskusních skupin se řadí open-source produkt **phpBB** (detaily na [7]).

Kapitola 4

Analýza požadavků a návrh systému

4.1 Požadavky na diskusní skupinu

Jelikož se jedná o produkt, který bude reálně využíván uživateli, bylo nejvhodnější se při zjišťování požadavků zeptat přimo budoucích uživatelů. Průzkum proběhl na první verzi fóra, které právě mělo být inovováno. Zde uvádím několik hlavních bodů, které byly uživateli uvedeny. Seznam není kompletní, jde spíše o požadavky, které jsou specifické a vylučují použití obecných diskusních skupin:

- intuitivní a komfortní ovládání celého systému
- vstup na fórum omezit pouze pro určenou skupinu uživatelů
- každý uživatel má svoje oblíbené kategorie nezávisle na ostatních
- různé názory na třídění komentářů – třídit do kategorií nebo vypisovat sekvenčně za sebou
- rozlišit prioritu komentářů
- možnost zobrazit pouze dosud nepřečtené komentáře a mezi komentáři vyhledávat na základě různých kombinací podmínek
- vzkazy mezi uživateli by měly být jak klasicky veřejné, tak i soukromé
- bylo by vhodné mít možnost ke komentářům připojovat souborové přílohy
- do komentářů vhodně vkládat tabulky, obrázky a matematické vzorce
- možnost definovat některé kategorie jako oblíbené a ty zvýraznit či oddělit od ostatních
- integrovat do systému kalendář se soukromými a veřejnými poznámkami
- pro jednoduché průzkumy mezi uživateli zahrnout do systému anketní systém

4.2 Analýza požadavků

Zjištěné body v kapitole 4.1 jsem následně analyzoval a vytvořil na jejich základě neformální popis systému.

Jako dobrý způsob působí aplikaci rozdělit do několika funkčně nezávislých modulů. Ty by bylo možné implementovat a ladit téměř samostatně. Moduly by se potom zaintegrovaly do celkového systému. Charakteristiky jednotlivých modulů jsou rozepsány v následujících podkapitolách.

4.2.1 Modul správy uživatelů

Při návrhu tohoto modulu vyvstanula podstatná volba, zda-li bude systém volně přístupný, nebo s ním budou moci pracovat pouze řádně přihlášení uživatelé. Vzhledem k faktu, že předpokládané nasazení systému je pro vymezenou skupinu studentů (například studenty jedné školy), rozhodl jsem se jakoukoliv práci s diskusní skupinou umožnit pouze přihlášeným uživatelům.

Modul tedy umožní potencionálním uživatelům registraci. Ověřit, zda-li daná osoba je skutečně oprávněna užívat fórum, lze pomocí e-mailové adresy, kterou mají studenti většinou zřízenu školou a má tedy společnou doménu. Na e-mail je poté zaslána ověřovací zpráva s kódem, který registraci potvrdí a zaručí, že adresa nebyla smyšlená. Pokud by studenti společné adresy neměli, je možné registraci povolit bez oveření nebo by musel ověření provést administrátor. Předpokládám možnost využívání fóra více studijními skupinami, a proto uživatelé mají možnost výběru své skupiny, od níž je odvozeno jejich výchozí nastavení.

Hlavní funkcí modulu bude následně kontrolovat přihlašovací proces uživatelů. Umožní také uživatelům nastavovat jejich personalizované vlastnosti fóra (počet vypisovaných komentářů, autoobnovení, vzhled, atd.).

4.2.2 Modul pro práci s komentáři

Jedná se o jádro celé diskusní skupiny. Především na tomto modulu se uplatnily připomínky uživatelů, aby práce s celou diskusní skupinou byla intuitivní a komfortní.

Prvním rozporem v požadavcích uživatelů bylo, že požadovali třídění komentářů do kategorií, ale někteří už odmítali procházení více úrovněmi navigace. Rozhodl jsem se tedy pro řešení, které by vyhovovalo oběma skupinám. Komentáře by se třídily do kategorií (např. školní předměty) a do vláken (jednotlivá témata probíraná v kategoriích), ale systém umožní jak procházení touto poměrně přehlednou strukturou, tak i zobrazení komentářů sekvenčně seřazených podle času přidání s tím, že kontext komentáře bude uveden v hlavičce každého komentáře. Pro větší přehlednost by si uživatelé měli možnost volit své oblíbené kategorie, které by měly být ve výpisu upřednostněny.

Uživatelé by také chtěli mít přehled o svých přečtených a nepřečtených komentářích. Proto jsem se v této souvislosti rozhodl ještě pro implementaci možnosti vypsání komentářů, které uživatel dosud nepřečetl, a nemusel je složitě vyhledávat.

Pro vyhledání komentářů splňujících zadané podmínky je výhodné obsáhnout vyhledávací systém, který by umožnil vyhledání komentáře na základě obsahu, autora, data vložení a dalších klíčů.

Rovněž vhodná je realizace sledování komentářů přes kanál RSS, kde by byly publikovány nepřečtené a nejnovější komentáře. Tento kanál ovšem je třeba rovněž zabezpečit proti neoprávněnému přístupu.

Zřejmě proto, že průzkum byl realizován u studentů technické školy, byl projeven zájem o možnost komfortního dynamického vkládání technických dat přímo do komentářů a to i bez znalosti programování v jazyce (X)HTML. Jako vhodná vypadá podpora vkládání obrázků, tabulek a sazby matematiky.

Vzhledem k úvaze, že by diskusní skupina mohla sloužit i pro oficiální účely, rozhodl jsem se do systému integrovat možnost cenzury jednotlivých komentářů buď ze strany administrátora, nebo i automaticky. Automaticky by mohly být mazány komentáře obsahující vulgarizmy, obsahově nevhodné komentáře by již musel ručně cenzurovat pověřený administrátor, či moderátor.

4.2.3 Modul soukromých vzkazů (vzkazník)

V tomto případě jde o klasickou součást pokročilých diskusních skupin. Často je třeba adresovat nějaký vzkaz pouze určitému uživateli a ne vkládat komentář přímo do fóra. Kromě zobra-

zování vzkazů by vzkazník měl uživatele informovat o jeho nových nepřečtených vzkazích a poskytl možnost uživateli jednoduše odepsat.

4.2.4 Anketní modul

V požadavcích se vyskytla i vhodnost integrace anketního podsystému do diskusní skupiny. Přístup k této části se u různých produktů liší. Některé umožňují ankety vytvářet uživatelům, některé tuto pravomoc dávají pouze administrátorům. Já jsem se rohodl spíše pro variantu druhou. Používání anket jsem rozšířil o možnost udělení více hlasů v jedné anketě. Počet hlasů má možnost nastavit administrátor. Stejně je užitečná možnost nastavit datum začátku a konce ankety.

4.2.5 Modul s kalendářem

Hodně specifickým požadavkem studentů bylo zahrnutí integrovaného kalendáře s plánovačem přímo do diskusní skupiny. Do kalendáře by uživatelé vkládali události podle jejich uvážení. Nezbytná je také volba, zda-li jejich událost bude publikována všem (tyto ale bude moci regulovat administrátor).

Podstatou kalendáře je, aby uživatelé byli upozorněni na nejbližší události, které se uskuteční. Události uvažuji dvojího druhu – dlouhotrvající a události, které se týkají jednoho okamžiku. Podle toho bude vhodné rozlišit jejich značení v kalendáři.

4.2.6 Administrační modul

Nad celým systémem bude implementován rozsáhlý administrační systém. V případě běžného provozu by neměla vznikat potřeba zasahovat přímo do databáze. Při návrhu jsem volil mezi různými možnostmi integrace administrace do systému. Nakonec jsem zvolil, že administrátoři budou mít stejná přihlašovací jména a pouze po standardním přihlášení budou mít rozsáhlejší nabídku týkající se právě administrace.

Části, které by administrátoři měli ovlivňovat:

- správa uživatelů
- správa anket
- správa kategorií
- cenzura komentářů
- správa kalendáře
- přidělování práv administrátorům

K jednotlivým položkám administrátorům posléze přístup povolovat podle jim přidělených práv.

4.2.7 Modul s dodatečnými funkcemi

Tento modul jsem přidal hlavně z důvodu ucelenosti návrhu. V modulu se zahrnou vlastnosti diskusní skupiny, které nebude možné zahrnout jinam. Mohlo by se jednat o podpůrné výpisy (přihlášení, statistiky, apod.).

Kapitola 5

Konceptuální model

Pro formalizaci celého návrhu informačního systému je vhodné použít modelovacích technik vycházejících z jazyka UML (*Unified Modeling Language*).

5.1 Modelování pomocí UML

UML je standardizovaný modelovací postup softwarového inženýrství vyvinutý především společností Rational Rose (ale i dalšími) pro grafický návrh abstraktního modelu systému. Zahrnuje řadu diagramů pro modelování statických objektů i dynamického chování. Pro namodelování diskusní skupiny jsem využil 2 diagramů ze sady UML.

5.1.1 Use-case diagram

V překladu diagram případů užití charakterizuje, jaké účastníky bude systém uvažovat a jaké operace se systémem budou moci provádět. Modeluje tedy dynamické chování systému. Je využíván především pro ujednocení operací, které budou později implementovány pro jednotlivé uživatele.

5.1.2 ER diagram

Název vychází ze zkrácení slov Entity-relationship diagram, nebo-li entitně vztahový diagram. Jedná se o techniku, kterou lze srozumitelně namodelovat, jak budou později uložená data v databázi. Diagram nepopisuje dynamické chování dat, ale pouze strukturu uložení statických dat. Celý model by měl být maximálně srozumitelný, neboť jej často vytváříme společně se zákazníkem.

Celý model obsahuje 2 základní elementy – *entity* a *vztahy* mezi nimi. Jednotlivé entity potom zahrnují své atributy, což v podstatě později budou samotné hodnoty v databázi. V rámci modelování lze užít složených atributů, ale pro uložení v relační databázi je třeba je převést na hodnoty atomické. Aby bylo možné identifikovat nějaký konkrétní záznam v entitě, je použito prímárních klíčů (značeno <<pk>>). Vztahy užité pro modelování lze rozdělit podle počtu entit, které spojují na reflexivní, binární a ternární. Každý vztah je popsán několika vlastnostmi, z nichž musí být uvedena hlavně *kardinalita*. Ta vyjadřuje počet vztahů daného typu, které může tato entita vytvářet.

5.2 Navržený use-case diagram systému

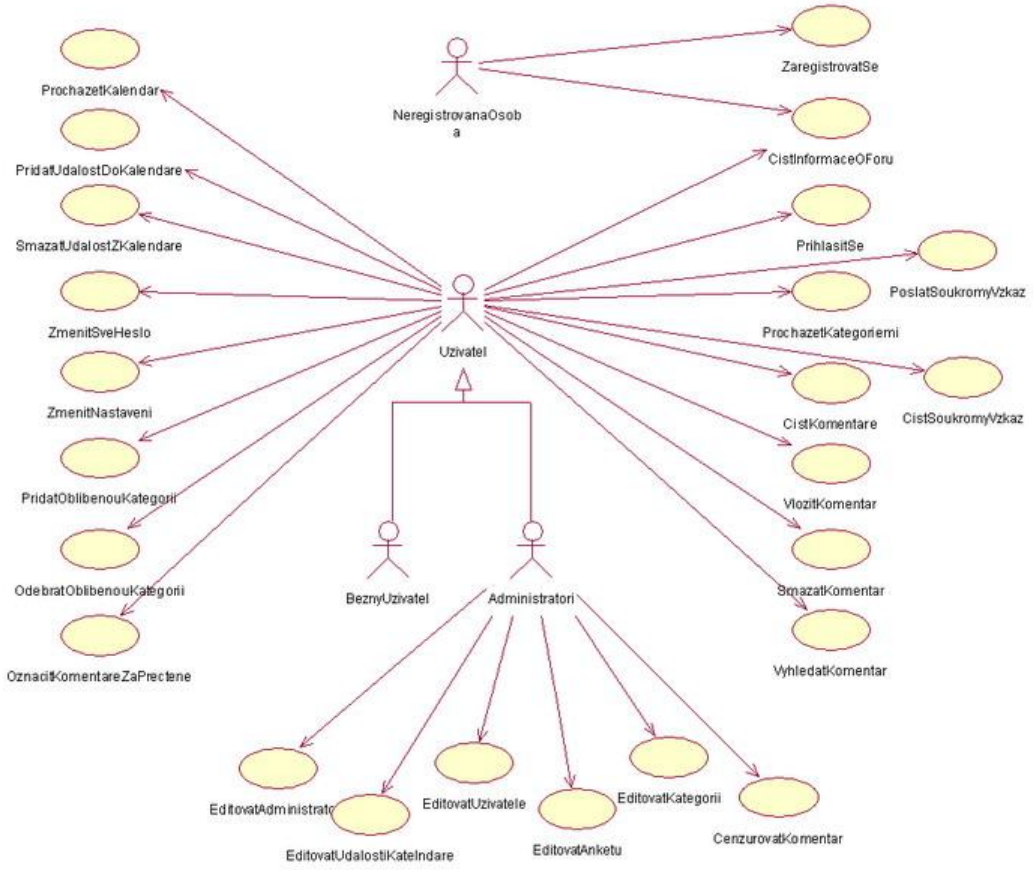
Na obrázku 5.1 je znázorněn use-case diagram systému. Za účastníky v systému jsou považováni neregistrované osoby a uživatelé. Shrnutí diagramu případů užití:

Neregistrovaná osoba má v systému velmi omezené pravomoce a prakticky se může pouze registrovat.

Uživatel je generalizovaným účastníkem běžného uživatele a administrátora, zahrnuje uživatele, kteří jsou přihlášeni do diskusní skupiny pomocí ověřených údajů.

Běžnému uživateli jsou přidělena standardní práva v přístupu do systému jako práce s komentáři, vzkazníkem, kalendářem, atd.

Administrátoři krom případů užití Uživatelů mají ještě další práva týkající se administrace diskuzní skupiny. Ty se dají odděleně přidělovat konkrétním osobám.



Obrázek 5.1: Use-case diagram diskusní skupiny

5.3 Navržený ER diagram systému

Obrázek 5.2 je ER diagramem modelovaného systému diskusní skupiny. Zde bych pouze rozvedl vybrané méně obvyklé namodelované situace.

5.3.1 Ukládání soukromých vzkazů

Podstatou je, že soukromý vzkaz je adresován od jednoho uživatele právě jednomu (jinému) uživateli. Uživatelé si mohou v průběhu času samozřejmě poslat více vzkazů, proto situaci nelze modelovat jako reflexivní vztah nad entitou `users`. Zavedl jsem tedy novou entitu `messages` s atributy vhodnými pro zprávu. Vztahy je nutné mezi těmito entitami vytvořit dva. Oba mají na straně uživatelů kardinalitu 1 (právě jeden odesílatel a adresát) a na straně vzkazů N (uživatel může odeslat nebo obdržet více vzkazů). Pro větší názornost jsou role uživatelů v diagramu pojmenovány.

5.3.2 Modelování oblíbených kategorií

Jak bylo již navrženo v kapitole 4.2.1, uživatelé jsou rozděleni do skupin. Z důvodu většího komfortu mají uživatelé možnost zvolit svoje oblíbené kategorie. Bylo by ovšem zároveň žádoucí, aby pro každou skupinu uživatelů existovaly výchozí kategorie, které jí přiřadil administrátor, a ty si uživatel mohl podle svých potřeb změnit a doplnit.

Toto by bylo možné realizovat tak, že se oblíbené kategorie patřičným způsobem na pokyn administrátora modifikují všem uživatelům ve skupině. Zde by ovšem uživatelé, kteří se zaregistrují po modifikaci oblíbených kategorií, neměli přidanou žádnou.

Zvolil jsem tedy řešení, že pro každou skupinu uživatelů administrátor modifikuje v podstatě jistou šablonu kategorií. Změny se následně také promítnou do samotných oblíbených kategorií uživatelů. Proto byl vytvořen vztah `group_default_categories`, který je šablonou oblíbených kategorií pro každou skupinu, mezi entitami `groups` a `categories`.

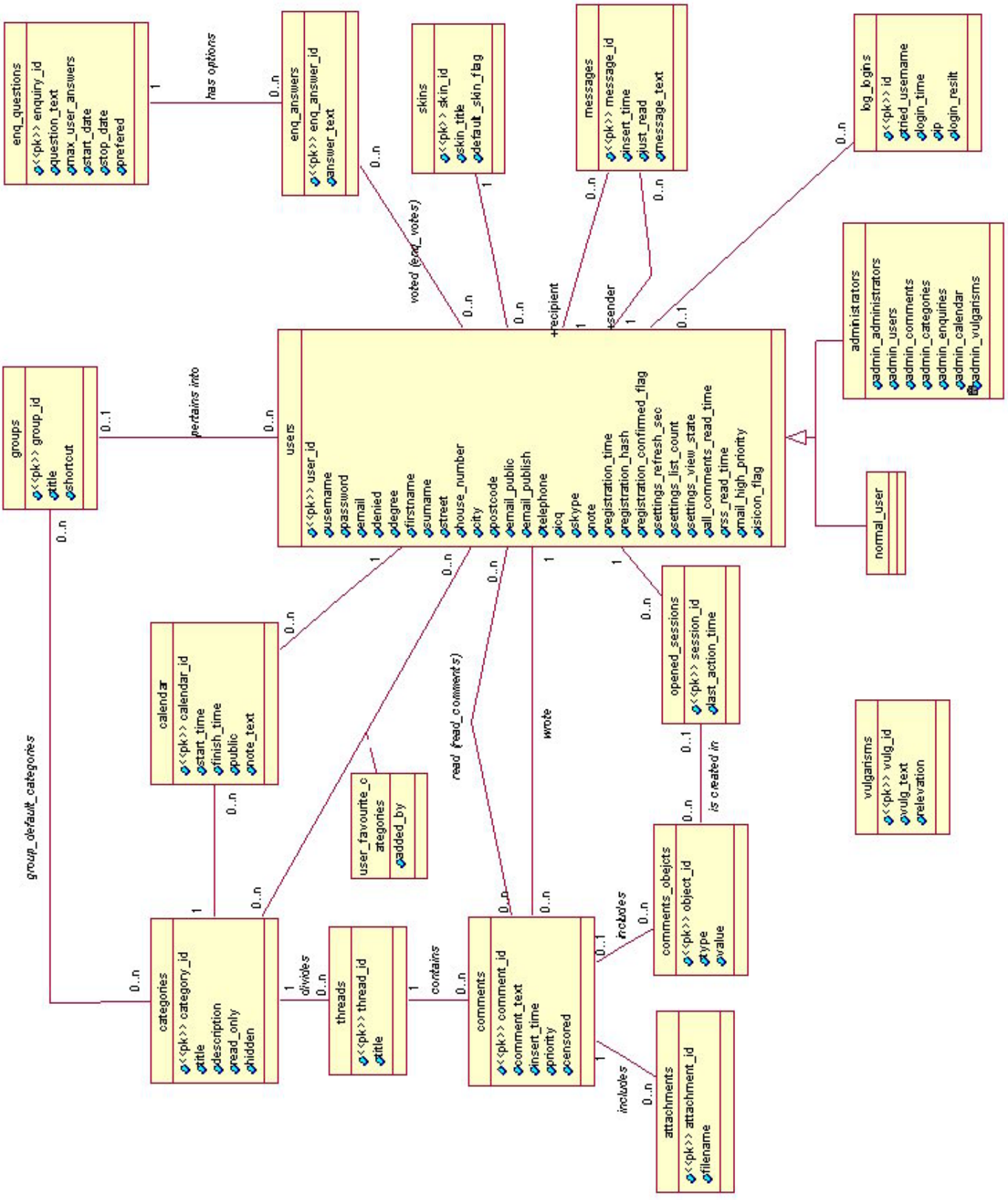
Každý uživatel tím má svoje zobrazované oblíbené kategorie uchovány díky vztahu `user_favourite_categories` rovněž mezi entitou `categories`, ale nyní už s entitou `users`. U existujících uživatelů se změny administrátora mění přímo v rámci vazby `user_favourite_categories` a nově registrovaným uživatelům je stav zreplikován právě ze vztahu (resp. přesněji vztahové entity) `user_favourite_categories`. Aby nedocházelo k nechtěným mazáním oblíbených kategorií uživatelům, kteří si je přidali ručně, je vztah doplněn o atribut `added_by`, který toto reflektuje.

5.3.3 Uživatelé a administrátoři

Zde se vyskytla zajímavá situace k modelování ve smyslu existence 2 typů uživatelů - **běžných uživatelů** a **administrátorů**. Ovšem vzhledem k tomu, že oba typy uživatelů mají většinu atributů podobných, využil jsem modelovací techniky zvané generalizace. Entita `users` je generalizovanou množinou z množin `normal_user` a `administrators`. Entita `administrators` má atributy specifické pro administrátory (například příznaky pro jejich práva) a entita `normal_user` neobsahuje atributy žádné, v diagramu je pouze z formálních důvodů.

5.3.4 Objekty připojované ke vzkazům

Vzhledem k tomu, že ke vzkazům se budou připojovat různé typy objektů a kromě textu to mohou být i obrázky, rozhodl jsem se objekty nevkládat přímo do textu jednotlivých komentářů, ale ukládat je vzlášť a ke komentářům je připojit přes referenci (podrobně popsáno v části implementace 6.5). Objekty jsou reprezentovány entitou `comments_objects` a vázány jsou s entitou komentáře, do kterého jsou vloženy (vazba s `comments`) *nebo* s entitou reprezentující session, v rámci které jsou připrávovány pro vložení (tj. s `opened_sessions`). Proto je zde členství obou vazeb *0..1* – vždy platí jen jedna z nich.



Obrázek 5.2: Celkový ER diagram

5.4 Transformace ER diagramu na schéma relační databáze

Z ER diagramu na obrázku 5.2 je v dalším kroku třeba provést transfromaci na tabulky relační databáze. Ještě před samotným převodem je dobré se ujistit, že v návrhu není chyba. Jedna z nejčastějších je, že se v návrhu vyskytne zbytečná **redundance**. To znamená, že se jistá stejná informace v databázi vyskytuje na více různých místech současně. Toto zabírá více místa a nutí nás udržovat složitěji konzistenci, neboť tyto 2 různé výskyty je třeba měnit společně. Nicméně v opodstatněných případech lze pomocí výskytů nejaké redundantní informace (předpočítání agregačních funkcí, které jsou často dotazovány a jen sporadicky měněny) ušetřit čas výpočtu. Vhodné je, aby tabulky splňovaly tzv. Boyce-Coddovu normální formu.

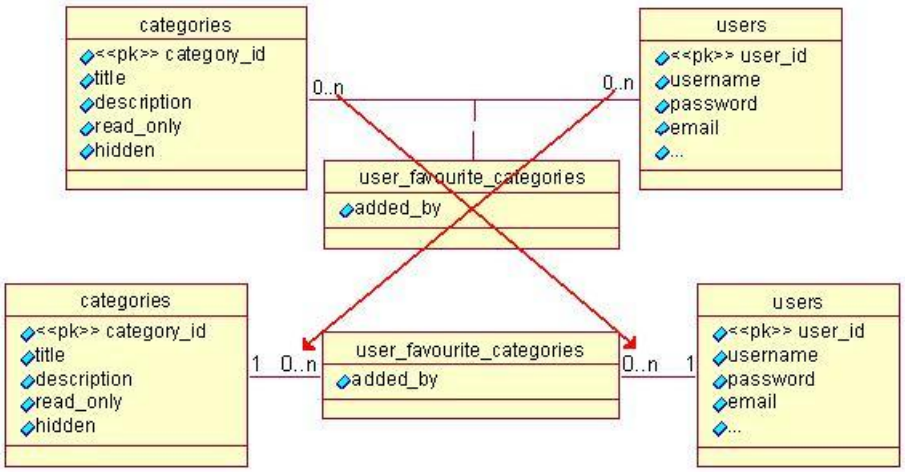
5.4.1 Ukázka transformace netriviálního vztahu na tabulky rel. databáze

Jelikož transformace celého ER diagramu je rozsáhlá, znázornil bych zde pouze transformaci vybraného, ne zcela triviálního vztahu, ze kterého je transformace dalších vztahů analogická.

Jako vzorový vztah pro demonstraci převodu jsem zvolil vztah mezi entitami `users` a `categories` pojmenovaný `user_favourite_categories` (význam vztahu byl popsán v části 5.3.2). Jedná se o vztah typu N:N, který je zároveň s atributem.

1. krok – odstranění vztahu N:N

Prvním krokem, který je třeba provést, je odstranění vztahu N:N, protože vztahy 1:N již lze převést snadno (viz. dále). Možným řešením situace je vytvoření další pomocné entity zvané *vazební entita*. Pro názornost ji pojmenuji stejně jako původní vztah. Tato entita bude mít jediný atribut, který převezme z původního atributu vztahu. Nyní musíme správně transformovat vazby. Obě původní entity budou vázány s novou vazební entitou a kardinalita na straně původních entit bude rovna **1**. Kardinalita na straně vazební entity je daná z původních kardinalit vztahu N:N (na základě úvahy dojdeme k závěru, že se kardinality zamění “do kříže”).



Obrázek 5.3: Transformace vztahu N:N na dva vztahy 1:N

2. krok – ověření atomicity hodnot

Dalším krokem, který už je stejný i pro transformace vztahů ostatních, je zajištění atomicity hodnot. V ER diagramu to prakticky znamená, že všechny složené atributy nahradíme jednoduchými – například pole adresa rozložíme na jeho složky (ulice, město, ...).

Tuto vlastnost již příklad splňuje, není tedy třeba nic upravovat.

3. krok – transformace entit a binárních vztahů na tabulky

Tento krok je klíčový pro celou transformaci. Spočívá v převedení entit, jejich atributů a vztahů na logické schéma databáze (tabulky relační databáze).

Tabulku vytvoříme pro každou entitu. Tím tedy vzniknou 3 tabulky:

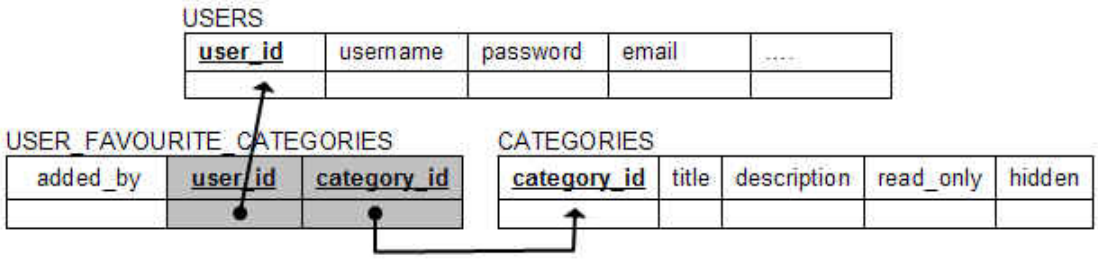
- categories
- users
- user_favourite_categories

Sloupce tabulek doplníme podle původních atributů tabulek.

Nyní zbývá transformovat vztahy. Vztah 1:N se transformuje tak, že na stranu tabulky, kde je u entity kardinalita vztahu N, se přidá sloupec s cizím klíčem odkazujícím na primární klíč entity, která je na straně s kardinalitou 1.

Poznámka: U vztahů 1:1 lze sloupec s cizím klíčem přidat do obou tabulek, záleží na uvážení programátora.

V mém příkladu tedy transformace vztahů spočívá v přidání sloupců `category_id` a `users_id` do tabulky `user_favourite_categories`. Tyto atributy jsou cizími klíči a společně tvoří složený primární klíč tabulky. Výsledné schéma je zobrazeno na obrázku 5.4, potržením jsou označeny primární klíče tabulek, barevným odlišením cizí klíče.



Obrázek 5.4: Výsledné schéma tabulek

5.4.2 Transformace generalizace uživatelů

Transformace generalizace z ER diagramu lze provést několika způsoby. Vždy je třeba zohlednit povahu modelované situace. Já jsem uvažoval, že množina administrátorů a normálních uživatelů nejsou disjunktní. Naopak výrazně menší množina administrátorů je pouze podmnožinou normálních uživatelů. Pokud by se všechny specializované atributy administrátorů přidaly do tabulky uživatelů `users`, byla by většina nevyužita. Toto řešení je tedy nevhodné. Naopak zvolit řešení s více tabulkami je ideální. Souhrnné atributy jsou v tabulce `users` a atributy administrátorů v tabulce `administrators`, kde mají záznam pouze uživatelé – administrátoři.

Kapitola 6

Implementace systému

6.1 Požadavky systému na použité technologie

Aplikace byla implementována pomocí těchto technologií:

- PHP4 (pro úplnou funkčnost je nezbytná verze PHP5)
- MySQL 5
- XHTML 1.1
- CSS 2
- Javascript

Z důvodu kompatibility ještě s většinou serverů nebyly pro implementaci využity objektové rysy verze PHP5 (postačuje verze PHP4). Nicméně některé funkce především pro práci s grafikou vyžadují grafickou knihovnu GD2 z nové verze PHP. V následujících kapitolách se budu věnovat především implementaci nestandardních vlastností.

6.2 Návrh grafického rozhraní systému

Základem použitelnosti každého webu je jeho přehlednost. Stránky musí být graficky navrženy tak, aby se i nově příchozí rychle a intuitivně zorientoval. Proto je vhodné na web správně umístit navigační prvky. V prvé řadě jde o **globální navigaci**, která má povahu nejdůležitější nabídky do hlavních sekcí webu. Jelikož bylo zjištěno, že uživatelé čtou informace na obrazovce ve tvaru písmene *F*, umístil jsem ji vodorovně do horní části hned po hlavičku. Pokud je uživatel zanořen v systému kategorií, je mu nabídnuta pod globální navigací i lišta s “drobečkovou navigací”.

Pro hlavní informační panel jsem zvolil prostřední sloupec. To je standardní řešení, protože se na něj uživatelé při čtení obsahu zaměřují.

Naopak do pravého sloupce byly umístěny panely, které nejsou při čtení webu pokaždé nutné a uživatel se na ně může zaměřit pouze v případě zájmu – například anketa, kalendář, apod.

Pro realizaci byly zvoleny klasické příjemné kombinace barev - bílá, modrá, černá. Grafika nebyla tvořena přiliš složitě, spíše má podpořit rychlé načítání stránek a snadnou orientaci. Přesto je použito obrázků či přechodů jednotlivých boxů jak pro lepší vzhled, tak pro optické sjednocení prvků podobného významu (boxy v pravé části stránek, apod.). Implementován byl i návrh skinovatelnosti fóra, takže výměnou stylopisu lze docílit zcela jiného vzhledu. Z těch si poté mohou uživatelé vybrat jim nepříjemnější.

Pro lepší orientaci uživatelů diskusní skupiny byl integrován systém *Nápovědy*, který je dostupný on-line k sekci “O fóru”.

6.3 Implementace modulů systému

Implementace jednotlivých modulů systému proběhla v závislosti na analýze uvedené v kapitole 4.2.1. Nejdříve byl naimplementován skelet spolu s uživatelským rozhraním a následně postupně přidávány a integrovány jednotlivé moduly.

6.4 Sledování diskuse přes RSS kanál

Implementace RSS kanálu spočívala ve vytvoření skriptu, který by generoval příslušný XML formát. Aby nedošlo k porušení bezpečnosti fóra je nutné, aby před čtením kanálu RSS byla vyžadována autorizace. Z důvodu kompatibility byla implementována autorizace jak přes HTTP hlavičku (některé čtečky ji nepodporují), tak přes parametry v adrese. Po úspěšné autorizaci jsou přeneseny komentáře od posledního stažení RSS kanálu, nejméně však 30 posledních zpráv (některé čtečky bohužel nepodporují ukládání položek a při novém stažení se všechny dříve stažené ztrácí).

6.5 Implementace vkládání objektů

Celá diskusní skupina je zaměřená na studenty technické školy. V komentářích jsou často diskutována témata vyžadující zápis různých technických dat, vzorců či schémat. Často je jejich sázení dost nepřesné a dochází k omylům – nezřetelné indexy a mocniny, místo obrázků jsou vloženy pouze odkazy a tabulky jsou nahrazovány buď (většinou) nevalidním zápisem HTML kódu přímo do textu komentáře nebo ještě hůře zarovnáním do sloupců mnoha mezerami. Tyto důvody mě vedly k myšlence zahrnout do systému vizuální (dynamické) vkládání:

- tabulek,
- matematických vzorců,
- obrázků s možností editace před vložením.

Nejdříve jsem zvažoval, v jaké podobě je možné tyto objekty do komentářů vkládat. Jako reálné jsem uvažoval tři:

1. vkládat přímo do textu komentáře XHTML kód pro tyto objekty,
2. přiložit ke komentáři sadu odkazů, které by otevíraly okno, kde by byl objekt zobrazen,
3. vložit do textu pouze zástupný odkaz na objekt a informace o něm uchovávat externě.

Možnost první se ukázala velice nepraktická, neboť když už by se kód do komentáře vložil, nebylo by jej možné nikdy editovat ani jednoduše odlišit od textu uživatele. Navíc by tímto způsobem nešly vkládat obrázky, které by bylo nutné stejně ukládat na server – do textu by se vložil kód pro načtení obrázku.

Ani druhá možnost není ideální. Sice už jsou odděleny texty komentářů od generované části, ale zase není možné umístit objekt přimo do textu komentáře. Objekty by tím byly v podstatě jen přílohami.

Třetí možnost řeší problémy uvedené v předešlých bodech. Do textu komentáře je vložen místo kódu objektu pouze specifický symbol jednoduchý pro strojové zpracování a pomocí něho dohledán příslušný objekt. Tento přístup umožňuje abstraktní pohled na všechny typy objektů. Kód objektu je pak pouze primárním klíčem do databáze objektů, kde jsou teprve uloženy informace o typu objektu a jeho "hodnotě". Při zobrazení je tato reference nahrazena konkrétním objektem.

6.5.1 Vkládání objektů obecně

Jak již bylo naznačeno, při vložení se v databázi objektů vytvoří záznam pro tento objekt. Zároveň se do textu komentáře na vkládacím formuláři vloží reference na tento objekt do databáze. Pokud je komentář úspěšně vložen, jsou objekty přidruženy přes cizí klíč `comment_id` v tabulce `comments_objects` k vloženému komentáři. Toto spojení je vhodné například při mazání komentáře, protože už potom není nutné vyhledávat, jestli v komentáři existují nějaké reference, ze kterých by samozřejmě šly přidružené objekty získat také.

Problém ovšem nastává v případě, že komentář, do kterého již nějaký objekt vložen byl, je zahozen. Objekt v tuto chvíli je totiž již uložen na straně serveru. Aby ale nedošlo k jeho "ztracení" v systému (tím, že by nebyl svázán s žádným komentářem), rozhodl jsem se v této fázi objekt svázat s otevřenou *session*, ze které byl vložen – vazba s tabulkou `opened_sessions`. Jakmile je session z tabulky otevřených sezení mazána pro delší neaktivitu nebo při odhlašování, jsou zahozeny zároveň i všechny objekty, jejichž vložení nebylo dokončeno. Jak tedy znázorňoval diagram 5.2, je aktivní vždy právě jen jedna vazba přes cizí klíč – buď s tabulkou otevřených sezení, nebo s tabulkou komentářů.

Tím je tedy charakterizováno vkládání objektů z obecného pohledu a nyní bych se věnoval konkrétním situacím.

6.5.2 Implementace vkládání tabulek

Při návrhu jsem se snažil, aby bylo možné vkládat tabulky dynamicky a zároveň komfortně a intuitivně. Šablonu výsledné tabulky proto vygeneruje systém spolu s nevyplněnými poli pro jednotlivé buňky. Počet sloupců pro vytvoření šablony si uživatel specifikuje v prvním kroku a následně podle potřeby mění počet řádků.

Po vyplnění a vložení se celá formulářová struktura převede na XHTML kód tabulky, který je zapsán jako objekt v databázi objektů.

6.5.3 Implementace vkládání matematických vzorců

Při implementaci této funkce jsem volil, jakým způsobem by měl uživatel matematický vzorec zadávat. Vzhledem k tomu, že již existuje rozšířený standard pro sazbu matematických vzorců v podobě matematického prostředí programu LaTeX, převzal jsem jeho základní syntax.

Příkazy v tomto zápisu jsou v prefixové notaci (zápis zlomků), symboly pro dolní a horní index (tj. ^ a _) jsou zapisovány rovněž před text s indexem. Jako závorky se používají složené závorky { a }. Příkazy bez parametrů (sazba speciálních znaků) jsou vždy v místě jejich použití. Vzhledem k těmto faktům jsem se rozhodl syntaktický analyzátor implementovat na principu **rekurzivního sestupu**.

Celý zápis je chápán jako *výraz obsahující jednotlivé podvýrazy* (což jsou opět plnohodnotné výrazy). Proto je možné implementací funkce pro zpracování výrazů a jejím rekurzivním voláním zpracovávat většinu jazyka. Například při zpracování příkazu pro zlomek je volána funkce pro zpracování výrazů dvakrát – jednou pro zpracování výrazu v čitateli a jednou ve jmenovateli. Příkazy bez parametrů jsou zpracovány přímo.

V posledním kroku bylo nutné vyřešit analýzu závorek. Zde jde ale opět o analogický postup, při nalezení otevírací závorky dojde k volání funkce pro analýzu výrazu, ze které je návrat při nalezení ukončovací závorky. Díky rekurzi je párovost už hlídána automaticky.

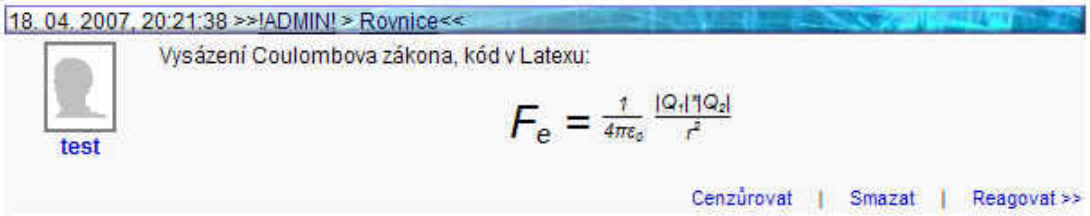
Výsledný kód XHTML se generuje přímo během analýzy. Pomocí CSS je formátován tak, aby vypadal jako vysázený. Problémem pro sazbu jsou nicméně zanořené zlomky, kde není řešeno dodržování úrovně hlavní zlomkové čáry. Při výskytu chyby během analýzy se systém z chyby snaží zotavit a analýzu dokončit. Seznam implementovaných příkazů naleznete v příloze B.

Příklad vysázení Coulombova zákona pro vakuum. V prostředí programu LaTeX je Coulombův vzorec vysázen kódem

```
F_e = \frac{1}{4\pi\epsilon_0} \frac{|Q_1|*|Q_2|}{r^2}
```

ve tvaru: $F_e = \frac{1}{4\pi\varepsilon_0} \frac{|Q_1|*|Q_2|}{r^2}$.

Vysázení v komentáři diskusní skupiny je zobrazeno na obrázku 6.1:



Obrázek 6.1: Vysázení vzorce Coulombova zákona

6.5.4 Implementace vkládání obrázků

Vycházel jsem z faktu, že někdy je nutné vložit nějaký obrázek, ale zrovna uživatel nemá k dispozici software pro úpravu obrázků. Z tohoto důvodu jsem krom samotného vkládání obrázků do komentářů implementoval i webový editor obrázků. Ten je založen na technologiích PHP a Javascript, kdy obrázek, se kterým se pracuje, je uložen na serveru a uživatel přes formulář v podobě panelu nástrojů pro úpravu fotografií posílá serveru příkazy pro změnu obrázku.

Implementována je podpora běžných typů obrázků jako JPEG, GIF, PNG a systém umožňuje konverze mezi nimi. Pro vhodné velikostní optimalitace obrázku je implementována změna velikosti. S využitím rozšířených funkcí grafické knihovny GD2 (v PHP5) je vytvořena i podpora aplikace filtrů obrazu. Nová verze knihovny je nutná z důvodu využívání pokročilé funkce `image_filter`.

Po vložení se narozdíl od předchozích případů vkládání tabulek a vzorců do databáze vloží pouze záznam o obrázku a z optimalizačních důvodů je obrázek uložen fyzicky přímo v souborovém systému serveru.

6.6 Implementace algoritmu pro autocenzuru

Na mnohých diskusních skupinách jsou problémem vulgární projevy jejích uživatelů. To mě vedlo k myšlence vytvořit algoritmus pro autocenzuru ihned po vložení. Některé diskusní skupiny především publicistických serverů tento způsob cenzury mají, ale je založen zřejmě pouze na databázi slov a odvozená nebo různě poupravená slova cenzora obelstí.

Před implementací jsem nejdříve na těchto serverech sledoval chování uživatelů, kteří cenzory založené na databázi obcházejí. V dalším kroku jsem v závislosti na této analýze navrhnul algoritmy pro hledání takto upravených výrazů. Jelikož je ovšem čeština velmi dynamický jazyk (skloňování, časování, odvozování, ...), bylo nutné, aby cenzor postupně svou databázi obohacoval o nalezená slova a z nich dále opět vycházel. V následujících podkapitolách se tímto řešením budu zabývat podrobněji.

6.6.1 Analýza chování uživatelů při obcházení cenzury

V této kapitole se budu zabývat vypozorovanými způsoby, kterými se uživatelé snaží cenzory obejít. Pro příklady jsem si zvolil slovo `demonstrace`, které předpokládejme, že je výchozím zadáním v databázi a slova jemu podobná požadujeme zacenzurovat.

Slova odvozená

Jsou základním typem, který by měl autocenzor hledat. V podstatě jde o to, aby administrátor nemusel zadávat slova, která mají stejný kořen a pouze jinou příponu. Tím lze zajistit, aby se hledaly vyskloňované a vyčasované tvary původních slov. Například se tedy bude jednat o slova `demonstrativní`, `demonstraci`, `demonstrovat` a podobně. U předpon je situace horší. Pravděpodobnost, že slovo má příponu je vysoká, ale předpony mají jen některá slova. Není tedy dobré předpony ignorovat, neboť by se tím mohl ztratit i kořen slova. Proto bude lepší do slovníku zadávat slova bez předpon a autocenzor si slova s předponami doplní sám.

Obelstění cenzora – proložení slova znakem

Velice často se na diskusích uživatelé snaží (úspěšně) cenzora obejít tím, že přestože cenzor má slovo ve slovníku, tak jej vhodně upraví, aby se jevilo jako neškodné. Toho dosahují uživatelé tak, že například zopakují každé (nebo některá) písmena vícekrát (`demoooonstrativníííí`) nebo slovo proloží nějakým nealfabetickým znakem (např. `d*e*m*o*n*s*t*r*a*c*e`). Tímto už většinou cenzora definitivně vyřadí. Jelikož český jazyk nemá příliš mnoho slov se zdvojenými písmeny, můžeme pro cenzora text upravit tak, aby zdvojené znaky ignoroval. Prokládání slova už ignorovat nelze, protože by mohlo dojít ke spojení více slov za sebou. Lepší je předpokládat, že nealfabetický znak je buď žádný znak, nebo právě jeden (viz další bod).

Obelstění cenzora – vynechání znaku

Z hlediska čitelnosti textu platí teorie, že pokud zachováme počátek a konec slova, zůstane text vždy pro člověka lehce čitelný. Vynechání začátku a konce slova už čitelnost ztěžuje. Proto uživatelé často volí způsob, že vynechají znak nebo skupinku znaků uprostřed slova a nahradí jej zástupným znakem (např. `demon***ativní`). Jak vidíte, slovo je čitelné, ale od původního se značně liší. Vhodné by tedy ale bylo, že když je uprostřed slova nějaký nealfabetický znak, předpokládat, že na jeho místě je nějaká obecná skupina znaků.

6.6.2 Implementace algoritmů pro cenzurování

Z předchozí analýzy je patrné, že způsobů je více, proto jsem implementoval více stupňů autocenzury založené na různých způsobech detekce. Zároveň uvažuji, že je lepší komentář v případě nejasnosti necenzurovat (ale třeba jen upozornit moderátora). Před samotným vyhledáním se nejdříve provede úprava textu, aby neobsahoval zdvojené znaky a je převeden na malá písmena. Pro detekci jsem implementoval 3 algoritmy:

Algoritmus pro odvození – každé slovo má nastaveno koeficient, jak moc lze od daného slova odvozovat (hodnoty 0 až 1 a speciální hodnota -1). Hodnota -1 znamená, že toto slovo je nevulgární a je při hledání ignorováno. Slouží především pro nadefinování výjímek ve vulgarizmech. Rozmezí hodnot 0 až 1 určuje relativní toleranci autocenzora.

Pro každé slovo z databáze se nejdříve spočítá tolerance pro hledání. Ta vyjadřuje, kolik znaků od konce slova může být vynecháno, nahrazeno jinými písmeny nebo o kolik může být nalezené slovo delší než hledané. Tolerance je dána vztahem (TOLERANCE_FORA je obecné nastavení skupiny, které udává maximální poměr tolerovaných znaků ku délce vulgarizmu) :

```
TOLER_ZNAKU = KOEF_VULGARIZMU * TOLERANCE_FORA * DELKA_VULGARIZMU
```

Ve zprávě je pak hledán řetězec vulgarizmu a pokud je nalezen příslušný vzorek, je ze zprávy vyseparováno celé slovo obsahující tento vzorek. Ve výsledku mohou nastat 2 situace. Pokud počet shodných písmen nalezeného slova vyhovuje toleranci, je označeno za vulgární. Slovo ovšem může být i delší, než je tolerance písmen. V tomto případě je zpráva označena za podezřelou a předána moderátorovi.

Algoritmem je docíleno toho, že se hledají slova se stejným kořenem a ignoruje se koncovka. Nově nalezená slova jsou následně zařazena opět do databáze, ale již s nižším koeficientem, aby nedošlo k odvození nechtěných slov. Přidání slov do databáze není podstatné pro tento algoritmus, který by samozřejmě slova se stejným základem našel i z původního slova, ale pro další algoritmus.

Algoritmus pro nealfabetické znaky – byl zvolen pro odhalování prokládání slov znaky a vynechání znaků (lze výhodně spojit v jeden případ). Algoritmus využívá funkce MySQL pro porovnávání se vzorkem. Všechny nealfabetické znaky jsou nahrazeny za znak %, což znamená libovolný řetězec v jeho místě 0-n znaků. Tím lze docílit hledání proložených slov i vynechaných znaků ve slově.

Algoritmus pro oddělovače slov znaky – doplňuje předchozí algoritmus, neboť znakem, kterým uživatel slovo proložil, může být i mezera. Potom se taková slova jeví pro předchozí algoritmus jako více jednopísmenných slov. Proto v tomto kroku jsou odstraněny veškeré nealfabetické znaky a ve vzniklém řetězci se opět pokouší algoritmus slova najít. Zde ovšem není jistota, že vulgarizmus nevznikl kombinací částí slov za sebou. Proto je tento výsledek zobrazen moderátorovi a je na jeho uvážení, zda-li komentář zacenzuruje. Zároveň se mu nabídne, aby slovo takto zamaskované přidal do databáze a v příštím cyklu bylo nalezeno už prvním algoritmem.

6.6.3 Shrnutí implementace autocenzury

Implementované algoritmy pro autocenzuru se zřejmě dají dalšími sofistikovanými způsoby obejít. Ovšem základní pokusy o znevážení diskuse by měly být odhaleny. Zároveň algoritmus musí být používán delší dobu, aby rozšířil svou databázi slov a vylepšil tím svoje vyhledávací schopnosti. Možným problémem by ale mohla být složitost algoritmu, která s počtem slov v databázi roste. Ovšem analýza textu se provádí pouze při vložení, a tak nejde o kriticky často opakovanou operaci a analýza textu je únosná.

Kapitola 7

Testování a nasazení systému do provozu

Systém byl testován v průběhu jeho vývoje, jestli jsou jeho výstupy a chování podle očekávání. Druhou fází testování bylo jeho zprovoznění místo původní verze, kterou nová verze nahrazuje. Při tomto testování bylo odhaleno ještě několik drobných chyb a problémů. Vybrané problémy a nástin jejich řešení uvádím v této kapitole.

7.1 Optimalizační problémy

Důležitým momentem vývoje internetových aplikací je ladění jejich výkonu. Aplikace většinou navíc běží na sdíleném serveru pro více podobných aplikací, a proto je důležité server nadměrně nezatěžovat. Obvyklé jsou optimalizace algoritmů s vysokým stupněm složitosti a také dotazování na databázový server, což může aplikaci výrazně zpomalovat.

7.1.1 Optimalizace počtu dotazů na MySQL server

Protože jde o aplikaci, kterou používá v jednom okamžiku více uživatelů, je vhodné, aby její běh měl minimální počet dotazů SQL. Při určování tohoto parametru jsem zjistil, že počet dotazů je poměrně velký, dosahoval až několika stovek dotazů. Důvodem bylo především opakované volání funkcí, které během jednoho běhu skriptu vracely pokaždé stejný výsledek, proto jsem zvolil řešení, že se dotaz vykoná pouze při prvním volání a výsledek je zapamatován po celou dobu skriptu. Při dalším volání už je hodnota navrácena přímo. Počet dotazů se snížil na řády desítek. I tak zde ale je prostor pro další případné optimalizace tohoto typu.

7.1.2 Optimalizace počtu záznamů o přečtených komentářích

Při testovacím provozu se projevil problém, který nebyl v návrhu řešen. Jednalo se o to, že informace, že uživatel daný komentář přečetl, se ukládala ve vazební tabulce `read_comments`. Tato vazební tabulka tedy pro každého uživatele obsahovala seznam všech jeho přečtených komentářů. Toto řešení zprvu nepředstavovalo problém, ale při 500 uživatelích (někteří aktivní, někteří ne) a zhruba 7000 komentářích začala tabulka obsahovat velké množství záznamů. Tím se značně zvětšila velikost této tabulky a to představovalo problémy s hostingem (každý den se zálohovalo zbytečně mnoho dat) a také s výkonem celého systému. (I při maximální snaze využít indexování se v MySQL tabulkách s více než 100 000 záznamy začíná projevovat stagnace výkonu.) Tento problém tedy bylo nutné řešit.

Jako první řešení se ukázalo například ukládat naopak záznam pro nepřečtenné komentáře každého uživatele. Tím by se počet záznamů v průběhu čtení snižoval. Ovšem z řádku 1 tabulky 7.1 lze snadno dopočítat, že toto by situaci spíše zhoršilo, protože v systému je mnoho uživatelů, kteří jsou neaktivní, nebo čtou fórum velice sporadicky.

Komplikovanější řešení, které se i po delším testování projevuje jako stabilní, bylo založeno na úvaze, že uživatelé, kteří fórum čtou, si často nechají zobrazit všechny nepřečtěné komentáře. Tím mají vždy vše přečtené. Myšlenka je tedy taková, že jakmile by uživatel přečetl vše, zaznamená se pouze tento čas, kdy má vše přečteno, a všechny starší komentáře jsou automaticky považovány za přečtené. Do tabulky `read_comments` by se ukládaly pouze záznamy do chvíle, než by opět měl vše přečteno. Výhoda tohoto řešení je ve snadné implementaci, která nepředstavuje výrazné zvýšení času výpočtu a je v podstatě kompatibilní s původním řešením (čili není nutné provádět radikální změny kódu).

Po 2 týdnech testování nového řešení došlo ke snížení počtu záznamů až na méně než třetinu a předpokládám, že při dlouhodobějším nasazení bude ještě učinější. Výsledky optimalizace jsou vystiženy v tabulce 7.1. Navíc lze předpokládat, že tento návrh nebude příliš ovlivněn ani kritickými okolnostmi původní realizace – tedy počtem uživatelů a komentářů. Neaktivní uživatelé totiž v tomto případě nemají záznamy žádné a aktivní, kteří sledují diskusi intenzivně, obvykle také žádné.

Stav	Max. záznamů	Reálný počet	Využití	Reálná velikost DB
Neoptimalizovaný	3 500 000	≐1 200 000	34.2 %	50 MB
Po optimalizaci	3 499 500	≐320 000	9.1 %	21 MB

Tabulka 7.1: Výsledky optimalizace pro 500 uživatelů a 7000 komentářů

7.2 Testování autocenzury komentářů

Testování cenzora bylo nezbytné. Aby byly výsledky věrohodné, podílela se na testování tohoto systému řada uživatelů. Zde bylo nutné po vyzkoušení zakázat odvozování od některých slov, neboť jejich základ byl obsažen ve slovech nevulgárních. Toto by bylo zřejmě třeba i v reálném provozu – pravidelně sledovat stav databáze a vhodně korigovat znalosti autocenzora.

Z vulgárních komentářů, jejichž obsah odpovídá znalostem v databázi, je nyní úspěšně odstraňováno cca 75 %. Algoritmus byl laděn, aby nedocházelo k chybné cenzuře. Ta se projevila zpočátku asi v 1 % případů.

Závěrem z testování je, že systém je použitelný v praxi, vyžaduje však občasný dohled experta, jenž by případně upravoval odvozovací koeficienty vulgarizmů a dohlížel by na komentáře, které autocenzor označil jako podezřelé.

7.3 Instalace systému

Instalce systému spočívá ve zkopírování zdrojových souborů na server. Je nutné povolit práva zápisu PHP skriptů do adresáře `/dtstore` (a jeho podadresářů). Následně je nutné vytvořit databázi a příslušnou strukturu. Schéma databáze je přiloženo v souboru `install/install.sql`. Pro systém je nutné nastavit správné kostanty pro běh (název serveru, připojení k MySQL databázi, ...) v souboru `config.inc.php`. Pro celý postup lze použít konfigurační skript `install/index.php`.

Kapitola 8

Porovnání s existujícími systémy a diskuze rozvoje

8.1 Porovnání diskusní skupiny s existujícím systémem

Celý systém byl navrhnut a realizován tak, aby spojil funkčnost standardních již existujících diskusních skupin s novými vlastnostmi, které by využili především studenti. Dnes standardním používaným řešením pro obecné diskusní skupiny je systém *phpBB*. Proto bych jej zde uvedl jako referenční řešení a srovnával mé řešení s tímto konkrétním systémem.

Pokud se zaměříme na strukturu kategorií a vláken tak zjistíme, že řešení jsou svojí strukturou podobná. Rozdíly se ukazují až v detailech. V obecném diskusním fóru je sice možnost fóra (kategorie) rozčlenit do skupin, ale již zde není možnost definovat své oblíbené skupiny. To byl ovšem jeden z požadavků studentů, neboť každý může mít zaregistrované jiné (volitelné i povinné) předměty a zajímat se o jinou problematiku. Zároveň považuji za vylepšenou indikaci nepřečtených komentářů. Zaměřil jsem se na to, aby i samotné komentáře, které uživatel dosud nečetl, byly zvýrazněné.

Za podstatné rozdíly ovšem považuji funkce při vkládání komentářů. Zde jsem integroval funkce, které obecné diskusní skupiny neobsahují. Jde především o připojování různých souborových příloh ke komentářům, možnost vložit obrázek, tabulku, či matematický vzorec přímo do textu komentáře. V obecné skupině lze většinou pomocí speciálních symbolů vložit pouze odkaz na obrázek, ale to vyžaduje, aby již byl umístěn na Internetu. Rovněž hlídání nevulgarity diskuse je na obecných skupinách poměrně problém. Musí být řešena ručně přítomností řady moderátorů. Já jsem tento proces alespoň částečně zautomatizoval.

Zcela novou funkcí je v mojí skupině integrace kalendáře přímo do systému kategorií. Tím může být ke každé zadána událost, na niž mají být uživatelé upozorněni (termín projektu, zkoušky, obhajob, . . .). Integrace podobné funkce v obecné skupině chybí – zřejmě zde není podobné funkce ani zapotřebí. Studenti ovšem tuto funkci jistě ocení.

Rozsáhlá administrace celé diskusní skupiny je vybudována nad oběma systémy. V obou lze uživatelům přidělit jistá práva ke správě, zde jsou výsledná řešení podobná.

Vzhledem k uvedení předchozích bodů jsem došel k názoru, že využití obecné diskusní skupiny, přestože postačuje, není v akademickém prostředí zcela vhodné. Názor mi podpořil i průzkum, který diskutoval nasazení obecné diskusní skupiny, nebo specializovaného systému přímo mezi uživateli. Uživatelé by jasně volili specializovanou skupinu, kterou by si mohli co nejvíce přizpůsobit svým potřebám a integrovala požadavky na funkčnost uvedené v analýze systému.

8.2 Diskuze možného rozvoje systému

Celý systém lze i nadále v mnohém rozvíjet. Nyní bych uvedl některé možnosti dalšího vývoje informačního systému:

Statistické výsledky anket – Anketní modul by bylo možné do budoucna doplnit o možnost generování rozsáhlejších statistik nad výsledky. Nyní lze zobrazit ke každé anketě její výsledky, ale zajímavé by bylo seskupovat výsledky více anket. Sledovat, jaké závislosti jsou mezi hlasy v jedné anketě a v druhé anketě – například z anket "Navštěvujete přednášky" a "Rozumíte většinou látce" vyhodnotit například výsledek kolik uživatelů na přednášky nechodí a látce potom nerozumí.

Kromě anket s přednastavenými odpověďmi by bylo dobré uvažovat i o implementaci systému pro zadávání otázek s "otevřenou odpovědí", kde by uživatel sám napsal svůj názor.

Sledování jednotlivých kategorií přes RSS – Prozatím je v systému možnost sledování celé skupiny. Vhodné by ale bylo, aby si uživatel mohl sám zvolit, které kategorie by mu RSS nabízelo. Následně by se mu stahovaly například jen oblíbené nebo vybrané kategorie.

Použití technologie AJAX – V systému je nyní na technologii AJAX implementováno pouze zobrazování přihlášených uživatelů. Vhodné by bylo tuto technologii využít i v dalších částech fóra, například pro ohlášení nových zpráv v oblíbených kategoriích, rychlé obnovení vlákna o nové komentáře a podobně.

Rozvoj autocenzora – V této oblasti lze vývoj rozhodně navázat. Problematika odhalování vulgarizmů je na všech diskuzních skupinách otevřená a lze předpokládat, že jak bude postupovat vývoj, tak i uživatelé budou přizpůsobovat techniky pro jeho obelhání. Námětem k impelementaci by mohlo být vyhledání záměrně vloženého písmene do slova, aby nebylo nalezeno. Zde je nutné ale řešit, aby se autocenzor nedopouštěl častých omylů.

Na vývoji budu i v závislosti na předchozích bodech možného rozvoje nadéle pokračovat.

Kapitola 9

Závěr

V dnešní době, kdy se Internet řadí mezi hlavní komunikační média, je existence diskusních skupin samozřejmostí. Avšak studenti mají pravděpodobně trochu odlišné požadavky na celkový návrh a funkčnost systému. Některé funkce obecných skupin nepořebují, ale jiné naopak chybí. Proto byla v rámci této práce implementována diskusní skupina zaměřená na potřeby studentů technické školy.

Za svůj přínos do oblasti diskusních skupin a informačních systémů považuji analýzu a implementaci některých speciálních požadavků studentů, jako je **vkládání tabulek**, **matematických vzorců** sázených zjednodušeným kódem programu LaTeX a **obrázků pomocí editoru** přímo do textu komentářů. Pro snadnou orientaci a dobrou použitelnost byla do systému implementována funkce pro **zvýraznění oblíbených kategorií** v závislosti na studijní skupině studenta a jeho vlastním osobním nastavení.

Rovněž pro zachování nevulgární diskusní atmosféry jsem navrhl **algoritmus autocenzury komentářů**, které obsahují vulgarizmy. Současné algoritmy vyhledávají na základě slov v databázi, ale já jsem se v této práci zaměřil i na odhalování vulgarizmů zamaskovaých různými způsoby (např. zdvojení písmen, vynechání písmene, jiné slovní tvary, ...). Algoritmus prozatím vykazoval vysokou úspěšnost detekce.

Při testovacím provozu jsem řešil i optimalizační problémy, zde jsem především navrhl úsporný způsob uložení přečtených komentářů. Tento způsob umožňuje reálně zmenšit velikost databáze až o cca 70 % původní velikosti.

Další možná rozšíření byly diskutovány v kapitole 8.2. Kompletní systém je nyní úspěšně nasazen v reálném provozu. Do budoucna jej lze rozhodně ještě v mnohém vylepšit a vyladit, ale i v současné podobě je dobře použitelný. Tomu nasvědčuje mimo jiné i to, že jej s oblibou používá celý ročník fakulty a systém plní svoji funkci.

Literatura

[1] Bakken, S.: *Manuál PHP*. 2007, `http://www.php.net/manual/cs/`, [cit. 3. 4. 2007].

[2] Bradley, N.: *XML - kompletní průvodce*. Praha: Grada, 1. vydání, 2000, s. 302-336, ISBN 80-7169-949-7.

[3] Cyroň, M.: *CSS - kaskádové styly*. Praha: Grada, 1. vydání, 2006, ISBN 80-247-1420-5.

[4] Kosek, J.: *PHP - tvorba interaktivních internetových aplikací*. Praha: Grada, 1998, ISBN 80-7169-373-1.

[5] Wikipedia: *Internet Forum*. 2007, `http://en.wikipedia.org/wiki/Internet_forum`, [cit. 10. 4. 2007].

[6] Wikipedia: *Document Object Model*. 2007, `http://en.wikipedia.org/wiki/Document_Object_Model/`, [cit. 4. 4. 2007].

[7] WWW stránky: *Internet Forum*. 2007, `http://www.phpbb.com/`, [cit. 10. 4. 2007].

[8] WWW stránky: *World Wide Web Consortium*. 2007, `http://www.w3.org/`, [cit. 4. 4. 2007].

Seznam použitých zkratek a symbolů

AJAX (*Asynchronous JavaScript and XML* – asynchronní JavaScript a XML) – jde o techniku vývoje stránek, kdy je dotaz na server odeslán na pozadí pomocí JavaScriptu. Server pak navrátí nějaká data, ty klient zpracuje a zobrazí. Změna obsahu tak probíhá bez nutnosti načtení celého dokumentu.

ASP (*Active Server Pages* – serverově aktivní stránky) – jde o alternativu k PHP, kterou vyvinula firma Microsoft. Zdrojové skripty jsou na straně serveru nejdříve interpretovány a následně odeslány klientovi.

CSS (*Cascading Style Sheets* – kaskádové styly) – jde o jazyk, kterým jdou aplikovat styly na dokument ve značkovacím jazyce. Stylopis je sada instrukcí, které jsou pomocí selektorů aplikovány na jednotlivé elementy. Technologie je standardizována konzorciem W3C.

DDL (*Data Definition Language* – jazyk pro definici dat) – jde o rodinu jazyků, kterými lze měnit struktura databáze (vytvářet, modifikovat, odstraňovat, či jinak ovlivňovat strukturu databáze).

DML (*Data Manipulation Language* – jazyk pro manipulaci s daty) – jde o rodinu jazyků, pomocí kterých dochází k manipulaci s daty, tj. jejich vkládání, aktualizací či mazání z databáze. V současné době je velmi populární verze DML daná standardem SQL.

DOM (*Document Object Model* – objektový model dokumentu) – rozhraní, které umožňuje objektový přístup k jednotlivým elementům dokumentu. Metody objektů umožňují modifikaci dokumentu.

ECMA (*European Computer Manufacturers Association*) – je mezinárovní standardizační organizací. Vznikla v roce 1994 a nyní standardizuje především technologii JavaScript.

ER diagram, ERD (*Entity-relationship diagram* – vztahově-entitní diagram) – je modelovacím nástrojem z rodiny nástrojů UML. Umožňuje modelovat statickou strukturu uchování dat v databázi. Obsahuje 2 základní součásti – entity a vztahy mezi nimi.

GD2 (*GD Graphics Library* – GD grafická knihovna) – jde o knihovnu pro manipulaci s obrázky. Byla integrována do řady jazyků včetně PHP. Podporuje formáty GIF, JPEG, PNG a další.

GIF (*Graphics Interchange Format* – grafický výměnný formát) – bezztrátový grafický formát používaný v oblasti internetové grafiky. Umožňuje použít paletu o 256 definovaných barvách a průhlednost. Patentován byl v roce 1985.

HTML (*HyperText Markup Language* – hypertextový značkovací jazyk) – značkovací jazyk používaný pro formátování internetových stránek. Jde v podstatě o definici dokumentu vycházející z rodiny jazyků SGML.

HTTP (*HyperText Transfer Protocol* – protokol pro přenos hypertextu) – komunikační protokol pro přenost obsahu internetových stránek mezi serverem a klientem.

JPEG – formát pro uložení obrazu pomocí ztrátové komprese. Narozdíl od formátu GIF umožňuje využít více barev, proto je vhodnější např. pro fotografie.

MySQL – je open-source řešení databázového serveru. Databáze je relační a obsahuje dotazovací jazyk založený na standardu SQL. Aktuální verzí je MySQL verze 5.

NNTP (*Network News Transfer Protocol* – protokol pro přenos novinek po síti) – jde o aplikační protokol, který umožňuje z newsového serveru stahovat jednotlivé články. Protokol je popsán v RFC 977.

PHP (*PHP:Hypertext Preprocessor*) – freewarový serverový jazyk, který se používá k dynamickému generování stránek na straně serveru. Počátky vývoje sahají do roku 1994 a dnes je jedním z nejrozšířenějších serverových skriptovacích jazyků.

phpBB – jde o univerzální open-source internetové fórum v jazyce PHP. Název vznikl ze zkrácení *PHP Bulletin Board*. Šířeno je pod licencí GNU GPL. Nyní je k dispozici verze 3.

PNG (*Portable Network Graphics* – obrázek přenostitelný po síti) – jde o nový bitmapový formát obrázků, který byl vyvinut pro použití v síti Internet. Narozdíl od formátu GIF není chráněn patentem.

SGML (*Standard Generalized Markup Language* – standardní jazyk pro zobecněné značkování) – byl vytvořen jako standard pro vkládání značek do textových dokumentů. Pro konkrétní dokument je pak specifikována jeho struktura. Takto se vyčlenilo například HTML.

SMTP (*Simple Mail Transfer Protocol* – protokol pro přenos zpráv) – je jednoduchým textovým protokolem pro přenos e-mailových zpráv mezi klientem a severem a pak mezi servery samotnými. Je definován v RFC 821. Později byl rozšířen o některé příkazy jako ESMTP.

SQL (*Structured Query Language* – strukturovaný dotazovací jazyk) – používaný deklarativní jazyk pro komunikaci s databázovými servery. Jeho historie sahá až do doku 1970 a poslední standard je z roku 2006.

W3C (*WWW Consortium* – WWW konzorcium) – mezinárodní standardizační konzorcium, které zahrnuje více než 120 organizací. Zabývá se především standardizací technologií, které se používají v síti Internet.

WWW (*World Wide Web* – celosvětová pavučina) – je službou, která je založena na protokolu HTTP. Server WWW nejčastěji naslouchá na portu 80. Umožňuje přenášet mezi serverem a klientem dokumenty, obrázky a jiná data. Dohlíží na něj organizace W3C.

XHTML (*Extensible HyperText Markup Language* – rozšířitelný hypertextový značkovací jazyk) – vychází z jazyků XML a HTML. Používá XML syntax a z HTML převzal sémantiku některých značek. Na rozdíl od HTML musí být například každá značka uzavřená a rozlišuje se velikost písmen.

XML (*Extensible Markup Language* – rozšířitelný značkovací jazyk) – jde o univerzální značkovací jazyk, kde sémantika značek není přesně daná. Používá se ke sdílení informací nezávisle na použitém informačním systému.

Seznam příloh

Dodatek A – Ukázky uživatelského rozhraní

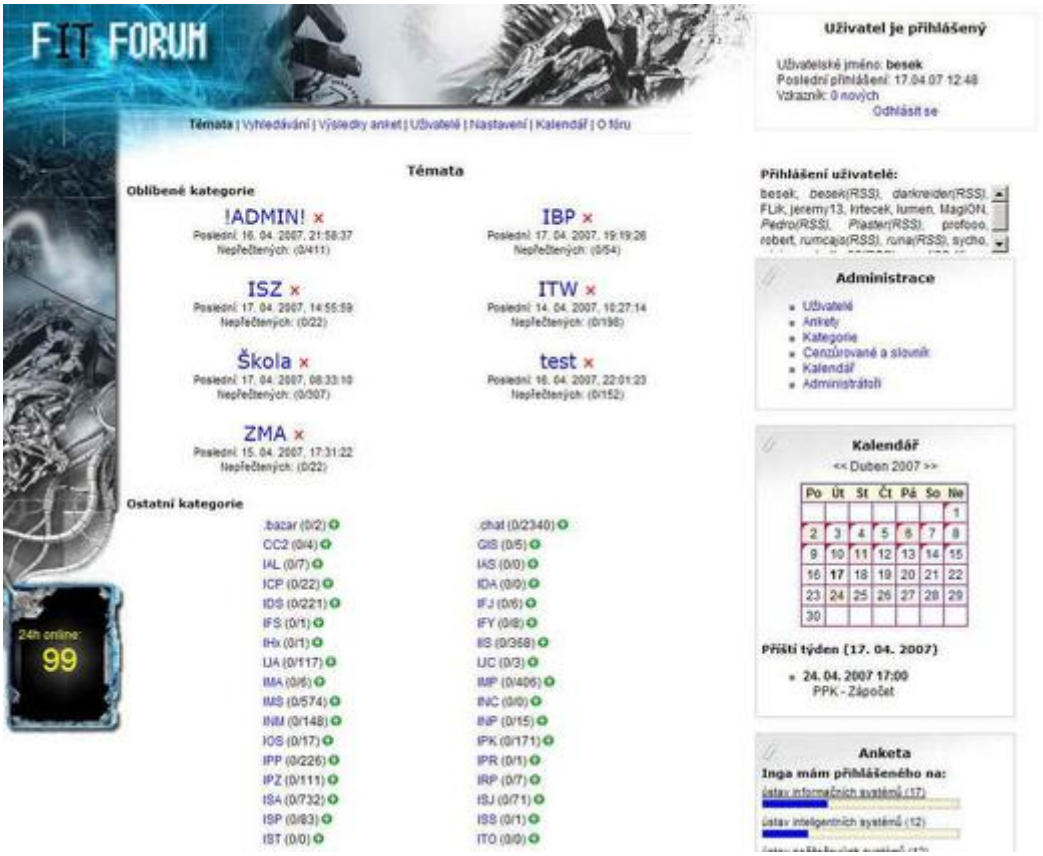
Dodatek B – Seznam implementovaných příkazů pro vkládání vzorců z prostředí LaTeX

Dodatek A

Ukázky uživatelského rozhraní

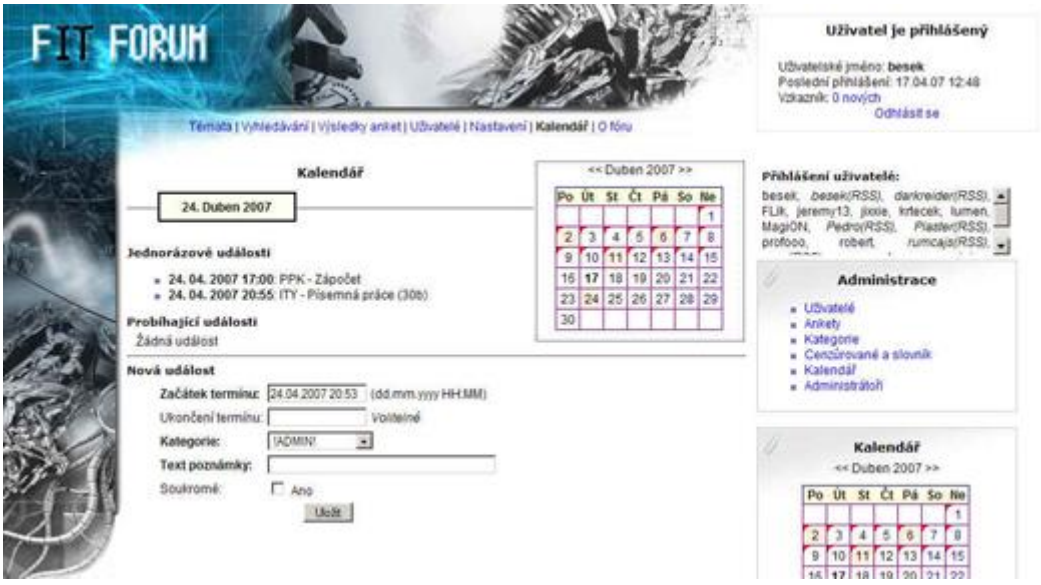
V této části jsou zobrazeny ukázky uživatelského rozhraní diskuzní skupiny. Náhled po přihlášení je zobrazen na obrázku A.1. Na obrázku A.2 je vyobrazena probíhající diskuze. Napravo je vidět i miniatura kalendáře, kde jsou vyznačeny podstatné dny. Detail kalendáře je pak zobrazen na obrázku A.3.

Z dynamického vkládání objektů je na obrázku A.4 editor obrázků. Dále pak obrázek A.5 zobrazuje vysázenou rovnici s indexy, zlomky a mocninami a obrázek A.6 editor tabulek.

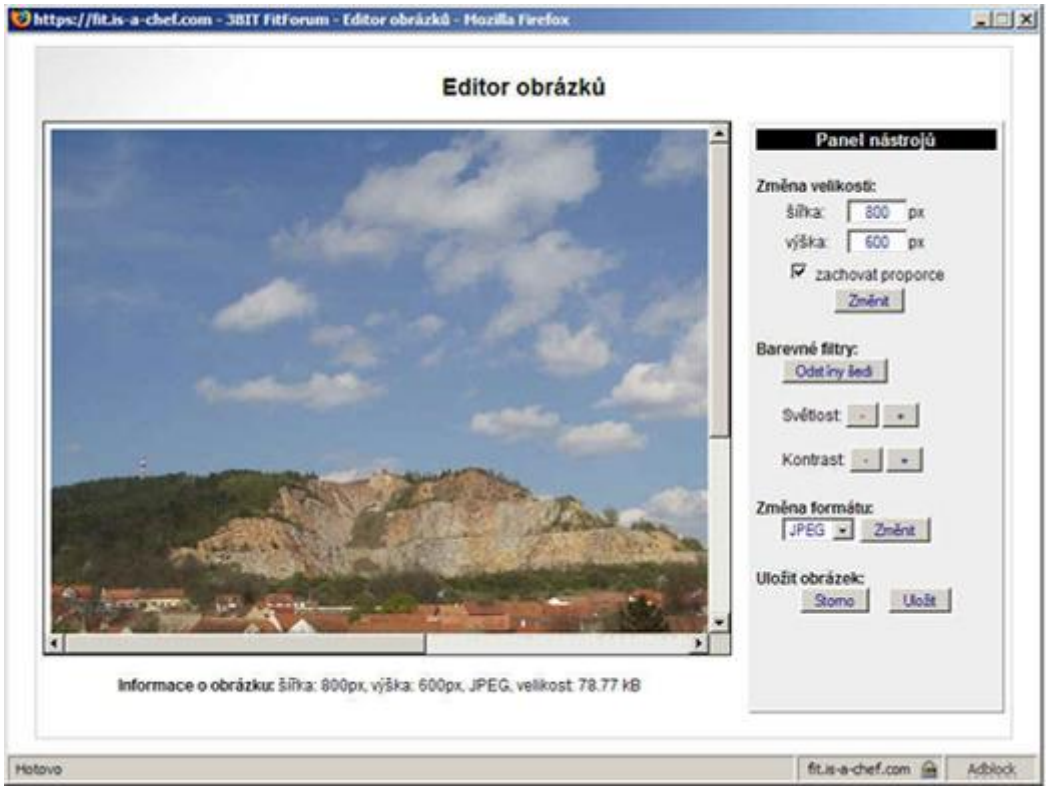


Obrázek A.1: Uživatelské rozhraní diskuzní skupiny

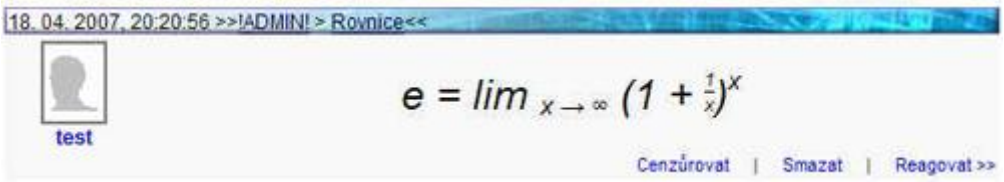
Obrázek A.2: Ukázka diskuze na fóru



Obrázek A.3: Detail kalendáře



Obrázek A.4: Editor pro dynamické vkládání obrázků



Obrázek A.5: Vysázení rovnice v textu komentáře

Obrázek A.6: Editor tabulek

Dodatek B

Seznam implementovaných příkazů pro vkládání vzorců z prostředí LaTeX

B.1 Řecká abeceda

Symbol	Příkaz	Symbol	Příkaz
α	\alpha	β	\beta
χ	\chi	δ	\delta
ε	\epsilon	η	\eta
γ	\gamma	ι	\iota
κ	\kappa	λ	\lambda
μ	\mu	ν	\nu
ω	\omega	o	\omicron
ϕ	\phi	π	\pi
ψ	\psi	ρ	\rho
σ	\sigma	τ	\tau
θ	\theta	υ	\upsilon
ξ	\xi	ζ	\zeta
ϑ	\vartheta	ς	\varsigma

Tabulka B.1: Příkazy řecké abecedy

Poznámka: Velká písmena řecké abecedy lze sázet stejnými příkazy, které ovšem začínají velkým písmenem, např. `\Gamma`.

B.2 Množinové symboly a operátory

Symbol	Příkaz	Symbol	Příkaz
$\ni$	\ni	$\leq$	\leq
$\geq$	\geq	$\equiv$	\equiv
$\sim$	\sim	$\neq$	\neq
$\subset$	\subset	$\subseteq$	\subseteq
$\supset$	\supset	$\supseteq$	\supseteq
$\vee$	\vee	$\wedge$	\wedge
$\cup$	\cup	$\cap$	\cap
$\emptyset$	\emptyset	$\forall$	\forall
$\exists$	\exists	$\neg$	\neg
$\in$	\in		

Tabulka B.2: Množinové symboly a operátory

B.3 Šipky a ostatní symboly

Symbol	Příkaz	Symbol	Příkaz
$\leftarrow$	\leftarrow	$\Leftarrow$	\Leftarrow
$\rightarrow$	\rightarrow	$\Rightarrow$	\Rightarrow
$\uparrow$	\uparrow	$\Uparrow$	\Uparrow
$\downarrow$	\downarrow	$\Downarrow$	\Downarrow
$\leftrightarrow$	\leftrightarrow	$\Leftrightarrow$	\Leftrightarrow
$\ldots$	\ldots	$\angle$	\angle
∂	\partial	$\sum$	\sum
$\int$	\int	∞	\infty
	\quad		\qquad

Tabulka B.3: Šipky a ostatní symboly

B.4 Syntax ostatních příkazů

Vysázení zlomku – pro vysázení zlomku lze použít příkaz `\frac{CITATEL}{JMENOVAT}`.

Horní index – (mocnina) se sází uvedením symbolu `^` před text mocniny.

Dolní index – se sází uvedením symbolu `_` před text indexu.