ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická – Katedra počítačů

BAKALÁŘSKÁ PRÁCE

Databáze publikací

2006 Martin Vodička

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu. Uděluji souhlas s užitím tohoto školního díla ve smyslu § 60 zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 28.6.2006 Podpis

Anotace

Cílem této práce je implementovat databázi publikací s interaktivním webovým rozhraním, ve kterém bude možno vyhledávat podle několika různých kritérií. Výsledkem hledání mají být citace, zobrazené podle některé ze zvolených norem a jako přílohy k nim, reálné publikace v elektronické podobě. Součástí práce je jak diskuze nad možnostmi použití podobných již existujících systémů, tak i jejich srovnání s aplikací, kterou jsem sám implementoval. Práce nenásilnou formou seznamuje se základními rysy, vyskytujícími se u databází podobného typu, určených ke správě publikací a dává nahlédnout do problémů, které jsem při vývoji vlastní aplikace řešil. Součástí práce jsou také testy funkčnosti a bezpečnosti implementované databáze.

Abstract

The main purpose of this work is to implement a publications database with an interactive web interface in which users will be able to look up publications according to some several criteria. Result of this search should be citations displayed in format according to one of the chosen norm and attached electronic publications. There is a discussion about possible using of some already existing systems, so as a comparison to the application I have already implemented. This work spontaneously familiarizes with the essential characteristics of similar databases designated to manage publications and gives you an opportunity to peep into the problems, which I solved by developing my own application. Part of the work deals with function tests and security questions.

Obsah

1 Úvod a popis problému

Publikace se obvykle popisují citacemi, jejichž účelem je rychle identifikovat publikaci podle názvu, jména autora, případně dalších vlastností. Existuje obrovská spousta norem, podle kterých se mají citace správně zapisovat, patří mezi ně např. MLA, APA, Chicago a další. Aplikace, kterou mám za úkol implementovat, bude moci jednotlivé publikace zobrazovat také dle některé ze zvolných norem.

S aplikací se bude pracovat výhradně prostřednictvím webového prohlížeče. Bude navržená tak, aby bylo výhodné ji používat k evidenci a rychlému vyhledávání publikací. Má být schopna zobrazovat citace v nějakém jednotném formátu daném normami, přičemž těchto formátů bude k dispozici více, a uživatel si z nich bude moci vybrat. Hlavním přínosem není generování normovaných citací, ale fakt, že v databázi mohou být uloženy i reálné publikace v elektronické podobě.

Protože publikací může být mnoho, a vyznat se v nich by dělalo potíže, je vhodné je rozdělit do kategorií. To vyplývá i ze samotného zadání. Je tedy potřeba dělit publikace podle kategorií (oborů) a zajistit, aby šlo publikace do nich zařazovat a dle nich vyhledávat. V publikacích je zapotřebí vyhledávat podle autorů, názvu publikace a oborů (kategorií do které publikace patří).

Vzhledem k tomu, že požadavky na vlastnosti publikací by se mohly v budoucnu měnit, napadlo mě rozšířit zadání a přidat navíc funkcionalitu, s jejíž pomocí by šlo k publikacím přidávat další atributy a to bez zásahu do návrhu databáze či kódu aplikace. V dalším textu budu tyto dynamicky přidávané vlastnosti nazývat jako „rozšířené atributy“. Rozhodl jsem se tak proto, že nikdy nelze přesně říci, že souhrn vlastností publikací, které má systém zaznamenávat, je kompletní. S úplně novým typem publikace, který se v budoucnu může objevit, přibudou i nové vlastnosti, které by se u ní mohly nebo dokonce měly evidovat. S příchodem elektronických médií se tedy můžeme setkat s elektronickými články, weblogy apod., u kterých je najednou výhodné identifikovat např. URL adresu, na které nacházejí. Systém, který je předmětem mojí práce, bude i na toto připraven.

2 Analýza

2.1 Návrh řešení

Volba implementačního prostředí:

S vedoucím práce jsme se dohodli na platformě, na které bude aplikace běžet. Budu ji tedy psát v jednom z nejrozšířenějších programovacích jazyků na webu, v PHP 5, jelikož i na fakultě, kde má systém běžet je PHP podporováno. Výstup, který bude aplikace generovat bude podle normy XHTML 1.0 Strict a layout stránek bude tvořen pomocí CSS 2.

Jako úložiště dat bude sloužit databáze relačního typu. V podstatě jsem měl na výběr databázové stroje typu MySQL a Postgres. Nakonec jsem se rozhodl, že MySQL bude pro mne přijatelnější. To proto, že jsem aplikaci vyvíjel na notebooku pod systémem Microsoft Windows a k měl jsem starý disk, formátovaný na FAT32. Postgres sice od verze 8 podporuje Windows NT, ale vyžaduje ke svému běhu filesystem NTFS. Protože na mém pomalém stroji NTFS nepřipadá v úvahu, tak návrh, programování i testování provedu s databází MySQL, které FAT32 nevadí.

Verze MySQL, kterou jsem k vývoji použil je 4.1, později jsem testoval i MySQL 5. Volba konfigurace MySQL 5 a PHP 5 je velmi vhodná. MySQL od verze 4.1 podporuje cizí klíče, transakční zpracování a poddotazy, tudíž má to hlavní, co je pro vývoj aplikace zapotřebí. MySQL verze 4 se na vývoj takové aplikace příliš nehodí, šlo by ji použít, ale na úkor toho, že nebudou fungovat subselecty a transakce, proč tedy nevyužít stroj, který už všechno toto umí.

Bylo mojí maximální snahou udělat závislost na typu databáze co nejmenší. Proto mě napadlo, že názvy funkcí, které používá databázový stroj a liší se od standardu (v MySQL třeba funkce pro spojení řetězců, pro práci s datumy, časem atd.) budu využívat střídmě a když už nějakou takovou funkci použiji, uvedu ji v dokumentaci s patřičnými detaily, kde se sql dotaz nachází atd. Napadlo mě tyto nestandardní názvy funkcí vytáhnout do nějakého společného konfiguračního souboru, ale potom jsem tuhle myšlenku odmítl, protože se většinou liší pořadí a sémantika jejich parametrů.

Velmi výhodným krokem bylo, že jsem se k práci s databází rozhodl použít balík DB z Pear frameworku [1]. DB je abstraktní datová vrstva, zjednodušující práci s databází, objektově orientovaná, kompatibilní s PHP 5. Její korektní používání částečně pomáhá zlepšovat portabilitu aplikací na různé druhy databází. Používá prepre/execute příkazy, propracovaným způsobem ošetřuje chyby sql dotazů, podporuje transakce, podporuje databáze fbsql, ibase, informix, msql, mssql, mysql, mysqli, oci8, odbc, pgsql, sqlite a sybase.

PHP 5 nabízí oproti verzi 4 také mnohá vylepšení, všimněme si alespoň podpory výjimek a vylepšeného objektového modelu. Kdyby nebylo podpory výjimek, musel bych obsluhu chybových stavů řešit zbytečně složitě. Díky výjimkám je budu moci elegantně ošetřovat a rozdělovat je na více typů.

Abych usnadnil čtení PHP kódu, budu do něj umísťovat komentáře ve stylu javadoc a za pomoci systému phpDocumentator [2] budu moci vygenerovat ke kódu jeho HTML dokumentaci.

Už v prvotním návrhu jsem se rozhodl použít některý ze šablonovacích systémů. Umožňují totiž oddělit aplikační a prezentační logiku webových stránek. Díky tomu se potom kód snadněji udržuje. Společným rysem všech šablonovacích systémů je myšlenka oddělení logiky od obsahu. U všech se to děje podobným způsobem: aplikace se rozdělí na 2 části, na soubory skriptů a soubory šablon. Když má skript generovat

výstup ve formě XHTML streamu odesílaného klientovi, obvykle nahraje šablonu a dosadí do ní své proměnné prostřednictvím parseru šablonovacího systému. Použití tohoto způsobu oproti přímému generování obsahu ze skriptů má velké výhody. U softwarových produktů, na kterých pracuje více lidí, je možné dělit tým na programátory a designery, snadno se hledají příčiny chyb, refaktoring je jednodušší. Proto i já jsem se rozhodl šablonovaní systém k implementaci použít.

Jak již jsem se zmínil, tak šablonovacích systémů je mnoho. Mám dřívější zkušenosti se systémem obsaženým balíku Phplib [3]. Jelikož jsem se ale dozvěděl o výhodách systému Smarty [4], neodolal jsem a použil ho pro vývoj. Smarty má oproti starším systémům typu Phplib řadu výhod. Způsob jakém zpracovává šablony je rychlý a dokonale propracovaný, navíc je kombinace systémů Pear a Smarty výhodná.

Uživatelské role a jejich vlastnosti:

Ze samotného zadání je patrné, že je nutné rozdělit uživatele databáze publikací do skupin. První skupina budou nahodilí návštěvníci, kteří si budou chtít prohlížet obsah databáze a budou vyhledávat publikace podle zadaných kritérií – říkejme jim **uživatelé**. Druhou skupinou osob budou lidé, kteří chtějí do systému přidávat nové publikace, tito lidé by měli mít k dispozici k zadávání publikací vlastní rozhraní, ve kterém vyplní všechny detaily o publikaci a odešlou zadanou publikaci ke schválení, těmto lidem říkejme **zadavatelé**.

Po vložení publikace zadavatelem bylo dle mého názoru vhodné, aby se publikace neobjevila hned v katalogu, ale byla napřed někým zkontrolována a schválena, teprve pak aby byla zobrazena. Bude muset existovat ještě člověk s více právy, který bude mít na starosti údržbu aplikace a schvalování nových publikací. Člověka s touto uživatelskou rolí budu v textu nazývat **administrátorem.**

Je otázka, zda chápat uživatele a zadavatele jako samostatné uživatelské role nebo je sjednotit pod jednu roli. V této práci budu obě role rozdělovat, z důvodu větší srozumitelnosti výkladu. Je ale pravděpodobné, že v ostrém provozu v určitém prostředí budou obě role splývat. Pravděpodobně všichni lidé z dané komunity budou moci být současně těmi, kdo publikace zadávají i prohlížejí.

Je dobré se zmínit o tom, jak bude diferencován přístup jednotlivých uživatelských rolí k systému. Na základě konzultace s vedoucím práce panem Petrem Fišerem jsem se dohodli na tom, že vzájemné oddělení uživatelů a zadavatelů nějakým způsobem autorizace nemá smysl řešit. Protože databáze publikací poběží nejspíše ve fakultním prostředí, kde se uživatelé běžně přihlašují některým z hesel, které jim bylo přiděleno fakultou, nemá smysl aby se uživatelé museli registrovat za účelem získání dalšího přístupového hesla. Budu ale tuto aplikaci navrhovat tak, aby se v budoucnu dala připojit na centrální databázi uživatelů, přes kterou potom může probíhat už ona zmiňovaná autorizace uživatelů a zadavatelů. Nutit uživatele se registrovat, je zbytečné také z toho důvodu, že není ještě jasné, zda bude aplikace použita jenom pro členy akademické obce FEL. Zatím ze uvažuje o tom, že v testovacím provozu bude přístupná úplně všem a později se napojí na databázi uživatelů a bude vyžadovat autorizaci..

Jelikož nemám dostatek informací a oprávnění, budu řešit diferenciaci uživatelů pouze zjednodušeným způsobem. S vědomím, že autorizace v zadavatelském rozhraní bude v ostrém provozu zapotřebí (aby bylo možné např. rozlišit, kdo zadal do systému jakého autora nebo vydavatele, a pokud ještě není schválen, zajistit aby se tento zobrazoval jen v rozhraní zadavatele, který je vlastníkem konkrétního záznamu a ne nikomu jinému).

Autorizace při vstupu do administračního rozhraní je už nezbytností. Administrátor

bude pravděpodobně existovat jenom jeden a bude se do svého rozhraní přihlašovat se svým uživatelským jménem a heslem, které mu bude přiděleno ke vstupu do chráněné části aplikace.

Webové rozhraní umožňující vyhledávání publikací, bude muset obsahovat vstupní pole pro volitelné zadání názvu publikace, autora a oborů do kterých patří. Výstupem má být seznam referencí na publikace v nějakém formátu. Je tedy zapotřebí navrhnout databázi tak, aby položky publikace byly uloženy odděleně a potom je šlo na aplikační úrovni skládat podle předem dané normy. Také je důležité aby struktura těchto norem šla snadno upravovat v administračním rozhraní.

Administrační rozhraní bude navrženo tak, aby administrátor měl při správě publikací především možnosti publikace schvalovat, mazat a editovat. Mělo by tedy umět vypisovat seznam publikací, které byly přidány zadavateli, s možností výběru dalších akcí. Administrační rozhraní by mělo být schopné manipulovat i se schválenými publikacemi, editovat je atd. Důležitou součástí rozhraní budou i formuláře k editaci externích objektů[1], na něž se budou jednotlivé publikace odkazovat.

Musí tedy existovat formuláře pro manipulaci s autory, vydavateli a kategoriemi. Nejlepší se mi zdálo rozdělit administrační rozhraní na několik částí, podle účelu ke kterému jsou určeny.

1. neschválené publikace
2. schválené publikace
3. správa rozšířených atributů (atributy, které se dají dynamicky přidávat)
4. správa kategorií
5. správa autorů
6. správa vydavatelů
7. správa formátů citací

Problematické části administračního rozhraní:

U částí administračního rozhraní 3, 4 a 5 musí být řešen následující problém: Když administrátor smaže například nějakou položku z tabulky reprezentující autora, musejí být vymazány i patřičné řádky z tabulky která dekomponuje vztah M:N mezi entitou, která reprezentuje publikace. Může tak nastat nepříjemný paradox, totiž že smazáním nějakého autora administrátor odebere nějaké publikaci její povinnou vlastnost (pokud měla pouze jednoho autora a i ten byl smazán). Tento problém by mohl být řešen tak, že administrátorovi zakážeme mazat záznam na který odkazuje publikace, která by po vymazání ztratila jedenu ze svých nejdůležitějších informací. Nakonec se mi zdá nejvýhodnější nic administrátorovi nezakazovat, ale pouze zobrazovat varování u autorů, kteří již jsou s některou publikací asociováni.

Zajištění konzistence databáze:

Protože do databáze budou vkládat své publikace více zadavatelů najednou, bude potřeba zajistit, aby se zadavatelé při své práci nijak neovlivňovali. Mohlo by se třeba stát, že zadavatel č. 1 přidá ke své publikaci nového autora. Kdyby to zjistil zadavatel č. 2 a použil toho samého, kterého vložil č. 1 a následně by si zadavatel č. 2 vše rozmyslel a onoho autora smazal, zadavateli č. 2 by aplikace zhavarovala, protože by vkládal

[1] V obecné analýze budu záměrně z důvodu lepšího pochopení výkladu nazývat autory, rozšířené atributy, vydavatele a kategorie souhrnným názvem **objekt**. V souvislosti s relační databází se hodí říkat spíše záznam (záznam v tabulce relační databáze).

publikaci, které odkazuje na neexistujícího autora. Řešení tohoto případu je následující: veškerá data, která vloží zadavatel do systému budou označena jako neschválená a bude k nim mít přístup pouze zadavatel, jemuž záznamy náleží. Z toho důvodu má rozhodně význam výše zmíněná autorizace zadavatelů, aby bylo možné identifikovat, komu která data patří a zobrazit je jen tomu, pro koho jsou určena.

V implementaci tuto funkci zahrnu a bude možné její chování ověřit ručním přepsáním loginu zadavatele v konfiguračních souborech. V ostrém provozu by měl být systém napojen na autorizační server. Ale pro testovaní účely se musí zavést entita, ve které bude systém uchovávat alespoň login zadavatele. Veškeré záznamy v databázi, které bude zadavatel moci vkládat budou odkazovat na konkrétní jeden záznam v této entitě (v mém návrhu se entita jmenuje submitter a má položky id a login, viz. kapitola 3.1 Datový model).

Mohlo by se také stát, že zadavateli, který odesílá do systému novou publikaci systém někde uprostřed vkládání do tabulek zhavaruje a rázem by databáze přišla o svojí konzistenci. Rozhodl jsem se tedy, že použití transakčního zpracování je nevyhnutelné. Pro databázi MySQL to znamená nutnost použít transakční tabulky InnoDB. Je jasné, že transakční zpracování nevyřeší všechno, ale v našem případě, kdy se při zadávání do databáze budou vkládat záznamy do několika tabulek u nichž je nutná vzájemná konzistence, transakce problém řeší. MySQL totiž provádí transakce, které jsou konzistentní (databáze se v průběhu transakce nejeví vůči jiným uživatelům v nějakém přechodném stavu) a izolované (právě probíhající transakce neovlivní transakci jinou).

Poslední, a zároveň těžko řešitelný problém je současně pracující zadavatel a administrátor. Neboť jestli je zadavatel ve fázi, kdy odesílá publikaci do systému a administrátor mu najednou z databáze umázne nějaký záznam, na který má odkazovat právě zadávaná publikace (třeba administrátor smaže z databáze nějaké autory, které měl zadavatel vybrané ve webovém formuláři), transakce ukládání publikace skončí s chybou, že vkládaná data nejsou konzistentní. Neboť odkazují na již neexistující záznam.

K řešení tohoto problému bychom museli hluboce zapřemýšlet, zda toto riziko opomíjet a vypsat zadavateli chybovou hlášku a nebo řešit problém jinak, například zamykat zadavatelské rozhraní, a přerušit tak práci všem aktuálně pracujícím zadavatelům, když administrátor vstoupí do svého rozhraní. Vzájemné vylučování zadavatelů a administrátora by pak fungovalo tak, že když se administrátor přihlásí do své sekce, všichni zadavatelé budou najednou vyhozeni a bude jim vypsáno oznámení, ve smyslu, že zadavatelské rozhraní bylo uzamčeno administrátorem. Podle mého názoru by tento přístup ale napáchal jen víc nepříjemností, neboť veškeré rozpracované publikace jednotlivých zadavatelů by byly ztraceny. Kdežto, když dojde k chybě, kterou jsem popisoval na začátku odstavce, množství celkově napáchaných škod bude nesrovnatelně menší.

Nejelegantnější způsob řešení by ale bylo, kdyby se těsně před vložením publikace zadavatelem tabulky uzamkly a ještě se zkontrolovalo, zda se v databázi vůči vkládané publikaci nic nezměnilo. A když ano, vypsalo by se zadavateli, co konkrétního se stalo (třeba že autor jeho publikace už neexistuje) a nabídlo se mu problém vyřešit. Kdyby tato kontrola neselhala, publikace by se normálně uložila a tabulky by se odemkly. Toto vylepšení ale nechám jako předmět budoucí práce.

Rozšiřitelnost atributů publikace:

Jak jsem již napsal v úvodním zhodnocení,. rozšířil jsem zadávání publikace o možnost přidání dalších atributů – tzv. **rozšířené atributy.** Původní představa byla taková, že by

v rozhraní pro zadavatele byly navíc vstupní políčka ve významu rozšířených atributů, které může zadavatel vyplnit. Počet a název těchto vstupních políček by závisel na počtu a názvu rozšířených atributů, které by vkládal administrátor do systému. Vkládal by je v případě, že by zjistil, že by se hodilo k publikaci přiřadit ještě nějakou vlastnost, se kterou se při návrhu databáze nepočítalo.

Nakonec jsem usoudil, že by i zadavatel publikace mohl navrhnout a vložit vlastní atribut. Proto jsem si dal za cíl tuto funkčnost do aplikace přidat. Jistě bylo opět dobré, si uvědomit to, že rozšířené atributy, které zadavatel navrhuje mají být do databáze uloženy s příznakem „neschválen“. Teprve na rozhodnutí administrátora, by se tento atribut stal veřejným a tedy i viditelným pro další zadavatele. Tímto popsaným způsobem se bude moje implementace chovat.

Formát generovaných citací:

Přemýšlel jsem o tom, jak splnit poslední bod zadání. Tedy jak generovat citace ve formátu daném příslušnými normami a hledal jsem vhodnou možnost jak záznamy patřící k dané publikaci generovat ve správném pořadí na výstup. Co mohlo být elegantnější než použít šablonovaní systém Smarty, který aplikace bude používat i ke generování vlastního GUI. Smarty se zdá být vhodným parserem jak pro již zmiňované oddělení logiky od obsahu webu, tak může být použit i k jinému účelu a toto je právě on.

Uživatelské rozhraní bude fungovat tak, že když nalezne v databázi publikaci, asociuje proměnné v šabloně s položkami publikace, které byly získány z databáze. Tento proces se provede pro všechny publikace, které si uživatel nechá zobrazit. Vznikne tím seznam publikací daného výběru. Administrátor by také měl být schopen prostřednictvím svého rozhraní tyto šablony editovat.

Šablony pro různé formáty citací, by měly jít pomocí administračního rozhraní jak editovat, tak i přidávat nové, resp. mazat. Z toho důvodu je lepší ukládat je raději v databázi než v souborovém úložišti. Smarty naštěstí umožňuje i tuto variantu. Stačí pouze pro něj napsat jednoduchý plugin.

Když už systém umí zobrazovat citace, zbývá jen domyslet jak udělat to, aby mohla být k citaci přiložen třeba samotná elektronická publikace (ve formátu pdf nebo jiném). Nabízí se zde možnost do vstupního formuláře, který bude mít zadavatel k dispozici přidat vstupní pole, do kterého bude vkládat své soubory. Konzultoval jsem toto s vedoucím práce a domluvili jsme se na tom, že počet souborů, které bude zadavatel vkládat nebude nijak omezen. Co se týče typů souborů, které bude moci vkládat, je vhodné z bezpečnostních důvodů povolit jen určité typy. Zkrátka nenechat vkládat všechny typy souborů. Tudíž je dobré aby někde v konfiguračních souborech byla direktiva, kde se typy povolených souborů dají nastavit. Administrátorovi lze umožnit, aby vždy vkládal jaký soubor uzná za vhodné, a zadavatele vést k tomu, aby zadával jenom soubory preferovaných typů.

Když shrneme veškeré požadavky, tak návrh databáze musí být takový, aby bylo možné generovat citace v jednotném formátu odpovídající normám.. Aby bylo možné dosáhnout tohoto cíle, je vhodné zadávat jednotlivé části citací jako samostatné objekty a ty jako záznamy vkládat do relační databáze. Protože se v průběhu času můžou měnit požadavky kladené na formáty citací, je chceme mít možnost rozšiřovat množinu atributů o nové kandidáty. Doplňkem citace bude odkaz na publikaci v „libovolném“ formátu. Pevně dané atributy publikace se rozlišují dle typu publikace. Jsou známy 3 typy publikací, které je nutno rozlišovat, jsou jimi „kniha“, „časopis“ a „konference“.

Seznam atributů společných pro všechny 3 typy publikací:

- název
- zadavatel
- vydavatel
- autoři
- země
- stát
- město
- jazyk

Seznam atributů pro publikaci typu „kniha“:

- ISBN
- rok vydání
- počet stran

Seznam atributů pro publikaci typu „časopis“:

- ISSN
- datum vydání
- svazek (volume)
- číslo
- stránky od – do

Seznam atributů pro publikaci typu „konference“:

- ISBN nebo ISSN
- jméno konference
- datum konání od – do
- stránky od – do

Všimněme si, že u všech publikací se objevuje atribut ISBN a ISSN, nemá smysl aby byly tyto atributy v databázi pojmenovány dvěma způsoby, budou se tedy oba ukládat do stejného sloupce se společným názvem. Aplikace rozliší, zda se jedná o ISBN nebo ISSN za pomoci atributu „typ publikace“. Když zjistí, že publikace je časopis, bude jasné, že jde o ISSN. Typ „konference“ nic nenapoví a proto si aplikace bude muset v rozhodování zda jde ISBN nebo ISSN pomoci zjištěním délky (je známo, že délka ISSN činí 9 znaků).

Pro publikace typu kniha a konference je třeba evidovat rok vydání, pouze u časopisů má význam uvádět i měsíc. U knihy nás zajímá počet stran, u časopisů je podstatné, na kterých stránkách je článek, jenž je předmětem citace. Podobně taky u konferencí, kde evidujeme stať rozsahem stránek ve sborníku.

Uvědomme si, že dělení publikací na 3 typy není úplně přesné a slouží jako zjednodušený model. Vyspělé systémy jako např. BibTeX (viz. kapitola 6.1.1) dělí publikace mnohem podrobněji. Na použitelnosti implementované aplikace to ale nic nemění. Nebude dělat problém publikace jiných typů vkládat, protože většina položek zadávaných publikací bude volitelná. Navíc si lze pomoci tzv. rozšířenými atributy – jejich podrobnější popis viz. kapitola 7.

2.2 Souhrn požadavků

2.2.1 Uživatelské role

Připomeňme si, že aplikace má být rozdělena do tří částí, podle uživatelských pravomocí osob, které do ní přistupují. 1. část aplikace se má chovat jako katalog publikací a nabízet uživatelům možnost vyhledávání. Této částí říkám uživatelské rozhraní a odpovídající uživatelskou roli jsem nazval jako „uživatel“.

Osobu, které je dovoleno vkládat publikace, jsem označil uživatelskou rolí „zadavatel“. Zadavatel bude mít možnost vložit publikaci do databáze prostřednictvím několika, k tomu určeným formulářům.

Administrátor bude mít k dispozici vlastní rozhraní, které má svou funkcionalitou dostačovat k mnoha potřebným úkonům, mezi něž patří správa objektů na které publikace odkazují (rozšířené atributy, kategorie, autoři, vydavatelé) a dále správa samotných publikací.

Zde je seznam činností, které jsou typické pro jednotlivé role:

Uživatel:

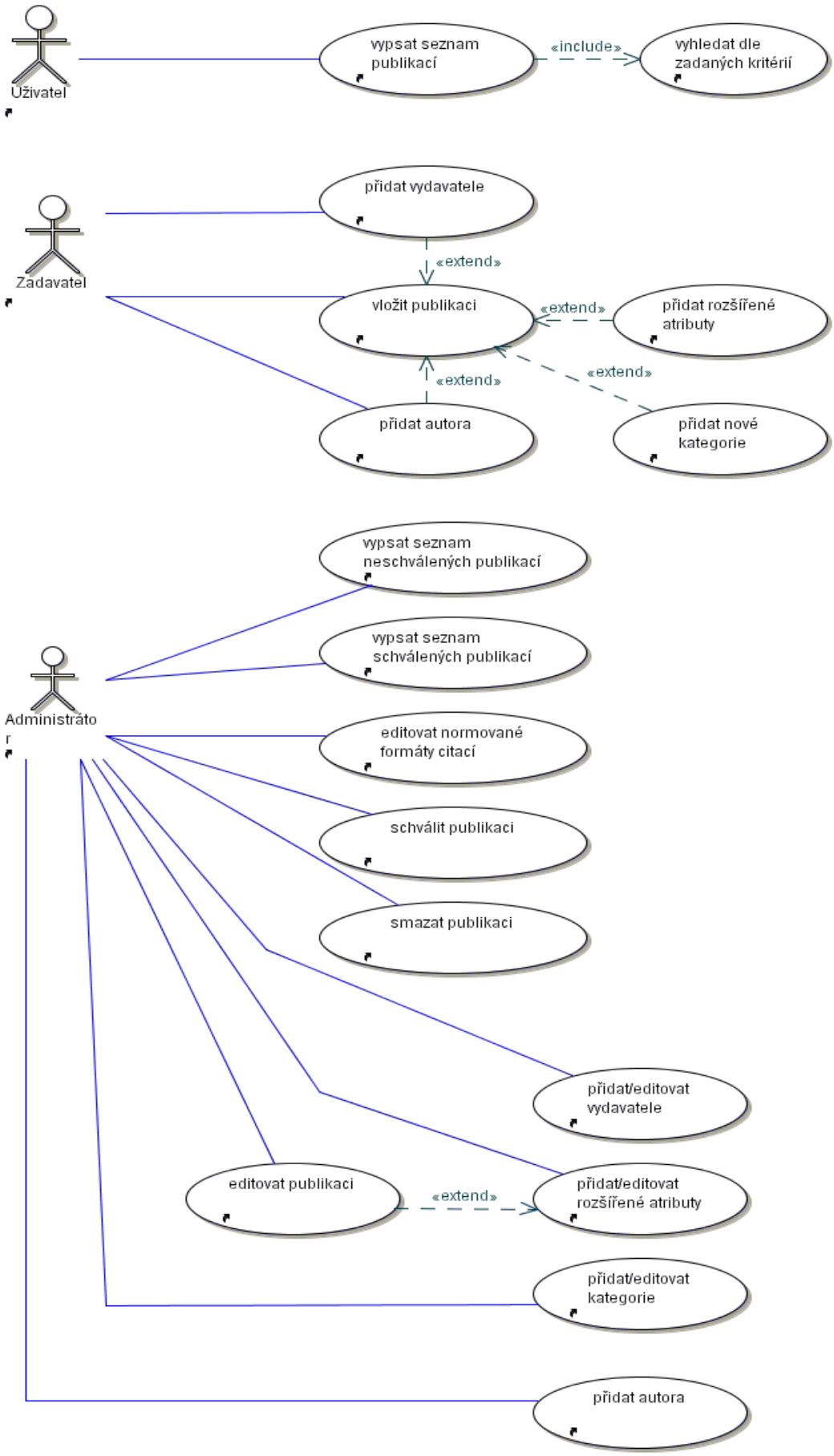
- prohlížení publikací
- vyhledávání publikací dle názvu, autorů a kategorií

Zadavatel:

- přidání autora
- přidání vydavatele
- přidání kategorie
- přidání rozšířených atributů k vlastní publikaci
- vložení publikace

Správce:

- prohlížení schválených publikací
- prohlížení neschválených publikací
- editace schválených publikací
- editace neschválených publikací
- schválení publikace
- smazání publikace
- přidávání a editace rozšířených atributů
- přidávání a editace kategorií
- přidávání a editace autorů
- přidávání a editace vydavatelů
- editace formátů citací



Obrázek 1 - diagram případů užití

2.2.2 Datové požadavky

Publikace povinně musí odkazovat na záznam typu „autor“, říká se tím: „tento člověk je autorem publikace xyz“. Zcela vždy je zapotřebí znát pouze jméno a příjmení autora, tedy nikde v citacích nechceme vidět informaci, kdy se autor nerodil, jaké získal akademické tituly a už vůbec nepožadujeme, aby byl tento člověk nějak jednoznačně identifikován. Jediné do nás zajímá je tedy jenom jméno a příjmení. V mém případě, kdy k implementaci používám relační databázi nás zajímá, v jakém vztahu má být autor a publikace. Obecně platí, že autor může napsat jednu a více publikací, přičemž každá publikace musí mít alespoň jednoho autora. Je zde vztah M:N.

Publikace může odkazovat na záznam typu „vydavatel“. U vydavatele potřebujeme evidovat pouze jeho jméno. Obecně platí, že publikace může mít jen jednoho vydavatele. Vydavatel může vydat více knih. Zde je vidět vztah 1:N.

Publikace může patřit do nějaké kategorie, a dokonce může být obsažena i ve více kategoriích zároveň. Z výkladu je zřejmé, že publikace bude odkazovat na záznam typu kategorie. U kategorie potřebujeme znát pouze její jméno. Platí, že kategorie může patřit žádné nebo více publikacím a publikace může být zařazena do žádné nebo více kategorií. Opět máme relaci typu M:N.

Publikace může mít přiřazeny rozšířené atributy. U rozšířených atributů je nutné identifikovat název a popisek (vysvětlující význam rozšířeného atributu) a konkrétní hodnotu atributu, která bude specifická pro každou z publikací. Opět platí, že více publikací může odkazovat na záznam typu „rozšířený atribut“ a více různých atributů může být referencováno jednou publikací. Opět vztah M:N.

(ER diagram viz. kapitola 3.1 Datový model)

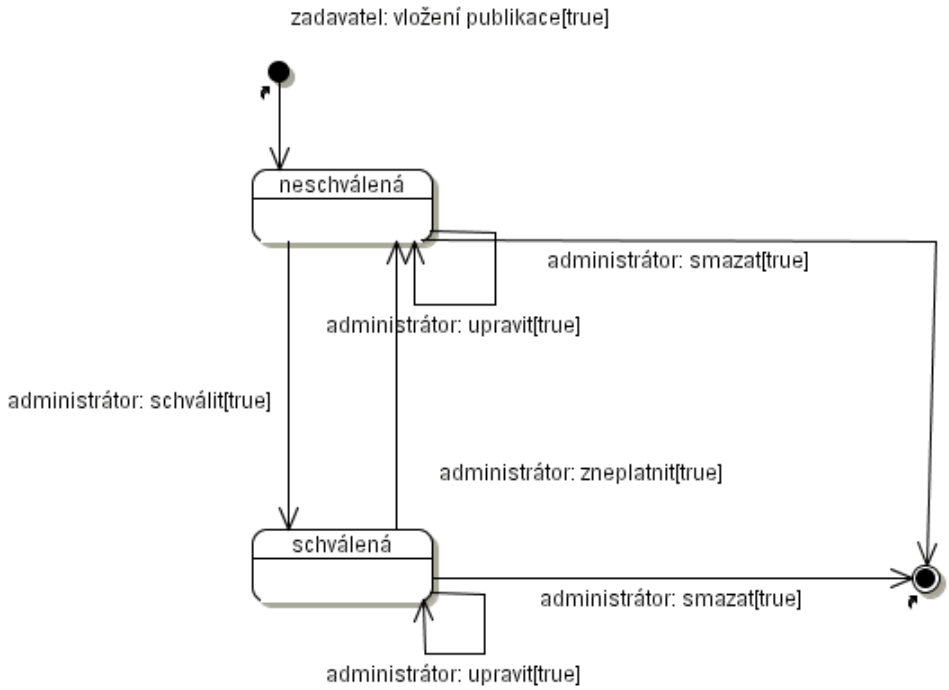
2.2.3 Procesní požadavky

Na tomto místě naznačím v jakých stavech se mohou nacházet jednotlivé záznamy, které spolu dohromady tvoří pojem „publikace uložená v databázi“. V úvodu analýzy jsem popisoval mechanizmus, jak bude aplikace nové publikace přijímat, že publikace po vložení zadavatelem je v neschváleném stavu, a že všechny nově vložené objekty jsou také neschválené (mám na mysli databázové záznamy: autor, vydavatel, kategorie a rozšířený atribut).

Celý tento proces je potřeba upřesnit, pokusím se o to slovním popisem a UML diagramy. UML diagram je zapotřebí brát jen jako názorný výklad k doprovodnému textu popisovaného procesu.

Publikace:

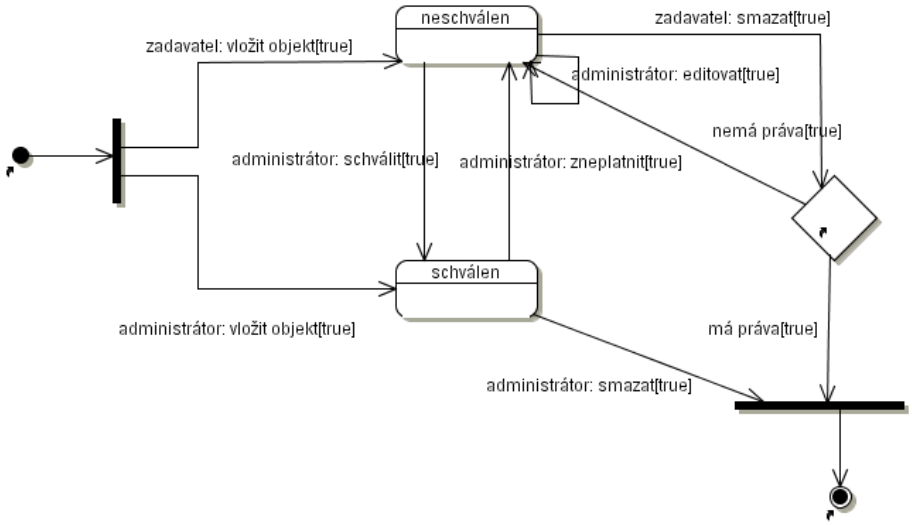
Zadavatel vstoupí do svého rozhraní, vyplní formuláře, vytvoří odkaz na autory a vydavatele, celý proces se bude odehrávat jako dočasný. Teprve když budou správně vyplněny všechny povinné údaje, smí být publikace vložena do databáze. Po vložení může administrátor publikaci upravit, schválit nebo smazat. Nachází-li se publikace ve schváleném stavu, administrátor jí může editovat, označit jako neschválenou, resp. smazat.



Obrázek 2 - stavový diagram publikace

Záznamy autor, kategorie, vydavatel a rozšířený atribut:

Tyto záznamy mohou být vloženy do databáze přes zadavatelské nebo administrační rozhraní. V praxi to znamená, že jak zadavatel, tak i administrátor mohou vložit do databáze nové autory, rozšířené atributy, vydavatele a kategorie. Když zadavatel vyplní povinná pole je záznam okamžitě vložen do databáze s příznakem neschválen. Administrátor bude moci všechny tyto nové záznamy prohlížet ve svém rozhraní. Může záznam upravit, schválit nebo smazat. Zadavatel může mazat pouze záznam, na který má práva (je jeho zadavatelem) a to ještě za podmínky, že je záznam neschválen a odkazuje na něj max. 1 publikace. Když administrátor schvaluje publikaci, všechny záznamy, které jsou s publikací v relaci mají být také schváleny. V jiném případě, když administrátor publikaci smaže, mají být smazání spolu s ní i všechny neschválené záznamy, které jsou s publikací asociovány a jsou asociovány pouze s touto jedinou publikací, v opačném případě se tento záznam mazat nesmí.



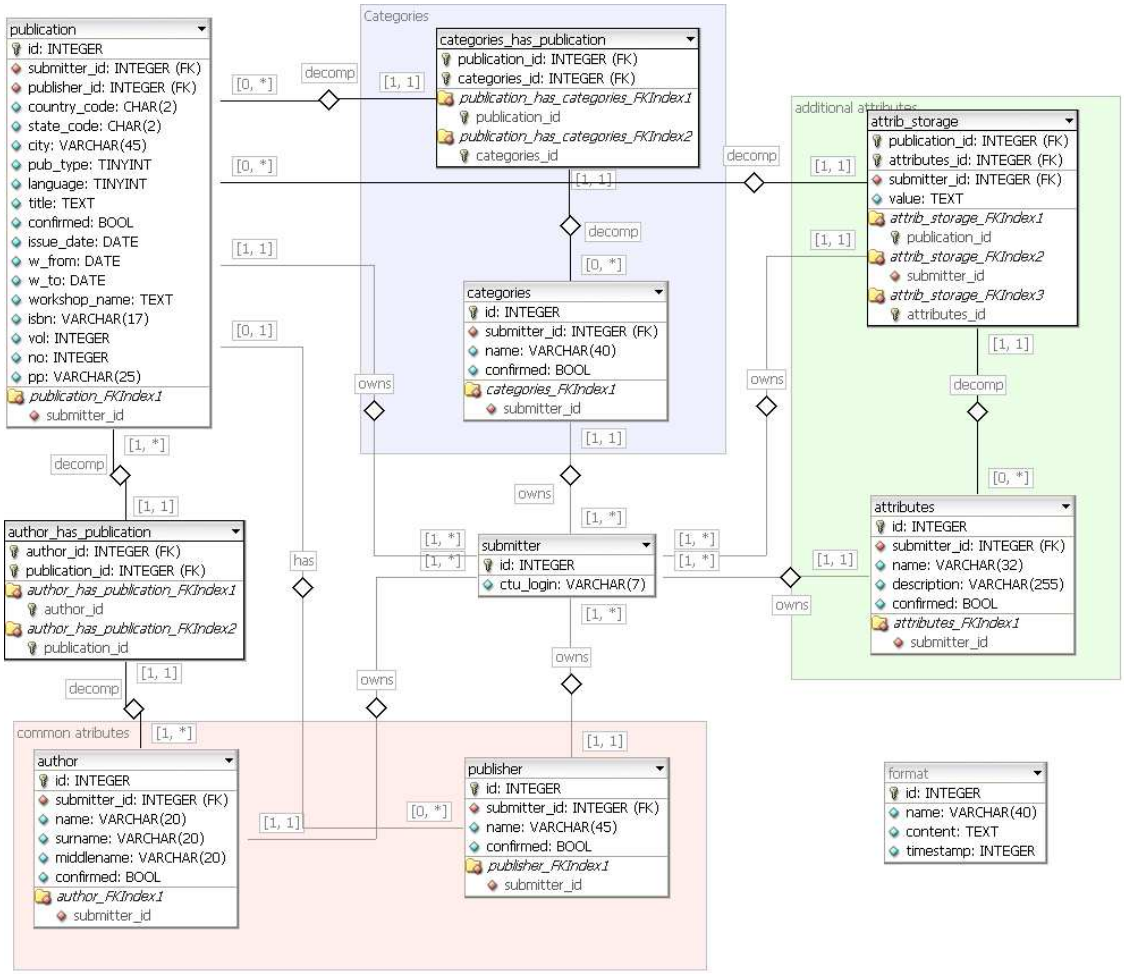
Obrázek 3 - stavový diagram objektů přidružených k publikaci

3 Popis implementace

Aplikace je rozdělena do 3 částí. Každá z nich bude sloužit jako vstupní bod pro uživatele s třemi odlišnými uživatelskými rolemi (uživatel, zadavatel, administrátor).

3.1 Datový model

Záznamy o publikacích budou ukládány do databáze MySQL. V následujícím textu popisuji jaký význam mají jednotlivé tabulky v databázi a přidávám popis jednotlivých sloupců.



Obrázek 4 - ER diagram

Tabulka submitter:

Obsahuje informace o uživatelské roli „zadavatel“. Je referencována tabulkami publication, autor, publisher, categories, attrib_storage a attributes. Tato kompozice splňuje požadavek na jednoznačnou identifikaci zadavatele, ať už jde o zadavatele konkrétní publikace nebo jen o osobu, která vložila do systému nové autory, vydavatele, kategorie, či rozšířené atributy.

Popis atributů tabulky submitter:

id (integer) ..... klíč

ctu_login (char)..... standardní login řetězec

Tabulka publisher:
Obsahuje seznam vydavatelů. Referencuje tabulku submitter.
Popis atributů tabulky publisher:
id (integer) ..... klíč
submitter_id (integer) ..... reference do tabulky submitter
name (char) ..... jméno vydavatele
confirmed (bool) ..... příznak, zda je záznam o vydavateli schválen, či nikoliv

Tabulka autor_has_publication:
Je dekompoziční entita, jenž svazuje řádky tabulky publication a tabulky autor.
Popis atributů tabulky autor_has_publication:
author_id (integer) ..... reference do tabulky author, je součástí klíče
publication_id (integer) ..... reference do tabulky publication, je součástí klíče

Tabulka publication:
Obsahuje základní atributy dané publikace Publiaction referencuje tabulku submitter v zájmu identifikace zadavatele. Dále obsahuje referenci na tabulku publisher a jednoznačně publikaci přiřazuje vydavatele. Mezi tabulkou autor a tabulkou publication je provedena dekompozice vztahu m:n na úrovni tabulky autor_has_publication, platí totiž pravidlo, že publikace může mít víc autorů a každý z nich může být autorem více publikací.
Popis atributů tabulky publication:
id (integer) ..... klíč
submitter_id (integer) ..... reference do tabulky submitter
publisher_id (integer) ..... reference do tabulky publisher
country_code (char) ..... znak země
state_code (char) ..... znak státu
city (char) ..... město, ve kterém byla publikace vydána
pub_type (integer) ..... typ publikace, nabývá hodnot v intervalu <1, 3>
language (integer) ..... jazyk publikace, nabývá hodnot v intervalu <1, 3>
title (text) ..... název publikace
confirmed (bool) ..... příznak, zda je publikace schválena, či nikoliv
issue_date (date) ..... datum vydání publikace
w_from (date) ..... datum zahájení konference
w_to (date) ..... datum ukončení konference
workshop_name (text) ..... název konference
isbn (char) ..... úložiště pro ISBN a ISSN
vol (integer). ..... číslo svazku
no (integer) ..... číslo
pp (char) ..... počet stran nebo rozsah stran

Tabulka categories:
Jsou do ní ukládány názvy kategorií, do kterých jednotlivé publikace patří. S entitou publication je ve vztahu m:n, tento vztah je dekomponován na úrovni tabulky categories_has_publication
id (integer) ..... klíč
submitter_id (integer) ..... reference do tabulky submitter
name (varchar) ..... název kategorie
confirmed (bool) ..... kategorie schválena / neschválena

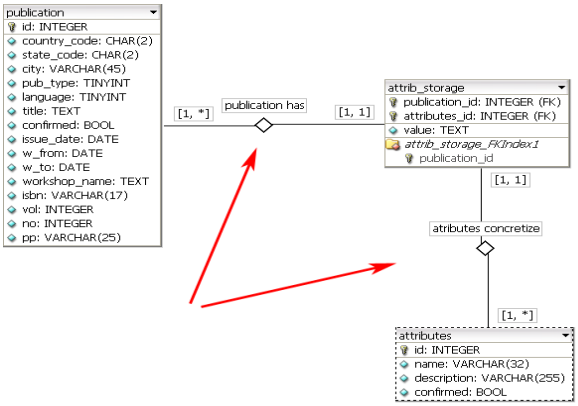
Tabulka categories_has_publication:
Plní funkci dekompoziční entity (upravuje relaci m:n mezi tabulkami publication a categories na dvě relace typu 1:n)..
Popis atributů tabulky categories_has_publication:
publication_id (integer) ..... reference do tabulky publication, je součástí klíče
categories_id (integer) ..... reference do tabulky categories, je součástí klíče

Tabulka attributes:
Je určena k evidenci rozšířených atributů k publikacím. S entitou publication je ve vztahu m:n, tento vztah je dekomponován na úrovni tabulky attrib_storage
Popis atributů tabulky attributes:
id (integer) ..... klíč
submitter_id (integer) ..... reference do tabulky submitter
name (char) ..... jméno atributu
description (char) ..... popis atributu
confirmed (bool) ..... příznak, zda je záznam o atributu veřejný, či nikoliv

Tabulka attrib_storage:
Plní funkci dekompoziční entity (upravuje relaci m:n mezi tabulkami publication a attributes na dvě relace typu 1:n). Navíc tato tabulka obsahuje úložiště konkrétních atributů, které náleží jednotlivým publikacím.
Popis atributů tabulky attrib_storage:
publication_id (integer) ..... reference do tabulky publication, je součástí klíče
attributes_id (integer) ..... reference do tabulky attributes, je součástí klíče
submitter_id (integer) ..... reference do tabulky submitter
value (text) ..... hodnota atributu

Tabulka format:
Je tabulka, která není v žádné relaci s tabulkami popsanými výše. Slouží k ukládání Smary šablon určených k formátování citací.
Polps atributů tabulky format:
id (integer) ..... klíč
name (char) ..... název normy, podle které je šablona sestavena
content (text) ..... vlastní Smarty šablona
timestamp (integer) ..... časové razítko šablony

Informace o rozšířených atributech se ukládají do tabulek attributes a attrib_storage. Tabulka attrib_storage je dekompoziční entitou vztahu m:n mezi tabulkami person a attributes, zároveň však funguje jako úložiště hodnot jednotlivých atributů, které příslušejí daným publikacím.



Obrázek 5 - vztah publikace a rozšířených atributů

3.2 Implementace uživatelského rozhraní

Uživatelské rozhraní je řízeno třídou User v souboru /User.inc.php, soubor stylů je v souboru /index.css. GUI je generováno z šablony /templates/user.tpl. Uživatelské rozhraní vyhledává a generuje citace, podle normovaného tvaru, čímž je myšleno rozmístění určené na základě analýzy šablon z databázové tabulky format. Uživatel může použitou šablonu formátu citací zvolit. Vybraný styl zobrazení má zůstat nastaven i po refreshi stránky, uživatel tedy musí mít povoleny cookies, aby šlo identifikovat jeho session. V případě, že jsou cookies vypnuté, po refreshi se mu formát nastaví zpátky na výchozí šablonu. Dosazování hodnot do šablon má na starosti třída Parser ze souboru /common/Parser.inc.php. Tato třída dosazuje do šablon proměnné, jimiž jsou atributy náležící dané publikaci, včetně jmen autorů, názvů vydavatelů a rozšířených atributů uložených v databázi, v tabulce attrib_storage. Z toho důvodu je zcela nezbytné, aby používané šablony byly validní a názvy proměnných v šablonách byly správně pojmenovány (což je ovšem starostí administrátora).

Popis funkce:

Vstupním bodem uživatelského rozhraní je soubor /index.php, V něm je vytvořen nový objekt typu User a konstruktoru jsou předány parametry, první parametr je pole obsahující data předaná metodou GET, vyfiltrovaná na výskyt speciálních znaků (jednoduchá uvozovka, dvojitá uvozovka, backslash). Třída je spuštěna prostřednictvím volání metody execute(). Další kód se už stará o vybrání správných publikací z databáze na základě kritérií, jenž byly specifikovány uživatelem pomocí filtračního formuláře. Tato kritéria byla předána přes parametry URL. Připmínám, že uživatel může specifikovat hledání fráze uvnitř názvu publikace, specifikovat kategorie, autory a počet zobrazených položek na stránku. Zde je tabulka vysvětlující význam proměnných předávaných prostřednictvím HTTP GET:

Keywords	Hledaná fráze
categories[]	Kategorie, ve kterých se hledá
Autor	Id autora
Bulk	Maximální počet položek na stránku
Page	Číslo stránky

Třída User umí v citacích vyhledávat. Vyhledávání je implementováno jako case insensitive hledání slovních spojení v názvech publikací (příkaz SELECT …LIKE '%fráze%'). U mojí databáze, je case insensitive vyhledávání podporováno pomocí collation utf8_czech_ci.

Fulltextové hledání v tomto případě nemá žádný smysl, to se vyplatí dělat jen u delších textů a ne u názvů s průměrnou délkou 20 znaků. V našem případě by i komplikovalo návrh databáze, protože fulltextové indexy se můžou definovat jen v tabulkách typu MyIsam.

Hledání je možné omezit na hledání pouze v určitých kategoriích a od konkrétního autora. To je realizováno spojením tabulky publication s tabulkou categories_has_publication a tabulky publication s tabulkou author_has_publication. Konkrétní kód se nachází ve třídě User, v metodě filter, ta vrací celý výsledek vyhledávání na základě omezujících kritérií. Metodě filter se předávají jako parametry veškeré proměnné z výše uvedené tabulky.

Když jsou hledáním vybrány správné publikace, stačí jen sesbírat ostatní informace (autory, vydavatele a rozšířené atributy), asociovat je s příslušnou publikací a zobrazit

seznam citací na výstupu. Zmíněné operace se provádějí v metodě display(). Posledním krokem je vytvoření a spuštění instance třídy Parser (zdrojový kód je v souboru /common/Parser.inc.php). Parser je autonomní třída, která dosadí proměnné do šablon správným způsobem za pomoci Smarty a vrátí string jako návratovou hodnotu metody get_parsed. Dosazení rozšířených atributů se provede, dostane-li třída Parser za 4. parametr metody get_parsed pole obsahující hodnoty: klíč (název rozšířeného atributu) a hodnota (text atributu). Třída Parser dosazuje všechny proměnné do jedné ze šablon umístěných v tabulce format. To, kterou šablonu použije závisí na výběru uživatele, který ji zvolí v XHTML prvku „select“ svého GUI, výsledkem jeho volby je proměnná typu celé číslo. Ta se dosadí jako 1. parametr metodě get_parsed.

Ve finále se do šablony /templates/user.tpl dosadí seznam nalezených citací, včetně souborů ke stažení a ke každé citaci je ještě přidána informace, do kterých všech kategorií je zařazena. Ve třídě User má své důležité místo metoda init_bars. V první řadě se stará o vyplňování checkboxů s kategoriemi a selectů se jmény autorů a limitem zobrazení. Ve druhé řadě vytváří navigační menu pod příliš dlouhým výpisem citací. V poslední řadě se zmíním o metodě clean, která je volána konstruktorem a zajistí, aby veškeré číselné proměnné, zaslané uživatelem měly vždy číselnou hodnotu a ne např. string.

Šablona /templates/user.tpl obsahuje javascriptový kód, jenž skrývá nepoužívané části filtračního formuláře. Pokud např. uživatel nespecifikuje autora nebo kategorie, ve kterých hledat, javascript tato pole skryje. Je počítáno i s možností, že javascript na klientském počítači nefunguje, v tom případě se při inicializaci žádná pole neskrývají a filtrační formulář zůstává stále funkční.

3.3 Implementace rozhraní pro zadavatele

Zadavatelské rozhraní je řízeno kódem třídy Load (kód je v souboru /load/Load.inc.php). Za pomoci šablon load_st1.tpl, load_st2.tpl a load_st3.tpl, nacházejících se v adresáři templates, generuje 3 formuláře. V prvním je zadavatel dotazován na data, podle kterých se bude odvíjet obsah 2. formuláře. Ve 3. formuláři aplikace zobrazí zadavateli kontrolní zhodnocení všech zadaných informací a nabídne mu vstupní pole pro vložení souborů. Rozhraní pro zadavatele vyžaduje zapnuté cookies, používá je k identifikaci sezení (volání funkce session_start() v souboru /common/common.inc.php). Pokud by byly cookies vypnuty, nelze publikaci zadat, viz. vysvětlení v datlším textu.

Po odeslání kteréhokoliv z formulářů, se data z něj uloží jako session proměnných se jmény feedback_st1, feedback_st2 a feedback_st3. To z důvodu, aby bylo zaprvé možné uživateli vrátit k opravě špatně vyplněný formulář a zadruhé, aby šlo se mezi formuláři pohybovat pomocí tlačítek další a předchozí, tak aby se zadavatel mohl vracet zpět již k vyplněným formulářům. Proto jsou funkční sessions nezbytné.

Veškerá data, která přicházejí z formulářů prochází validací. Jednak na klientské, tak i na serverové straně. Úroveň kontroly formulářů, je nastavitelná 3. parametrem v konstruktoru třídy Load. Pokud je jako 3. parametr zadána hodnota 1, formuláře budou fungovat ve striktním módu, tzn. Budou vyžadovány téměř všechny položky jako povinné. Tato možnost tu je, implementoval jsem ji, ale nemá smysl aby byl strict mode zapnut, neboť by to komplikovalo práci zadavatelů. Implementoval jsem ho jenom jako možnost, kdyby v budoucnu chtěl někdo zpřísnit úroveň kontroly. Když je jako 3. parametr zadána hodnota 0, bude povinně vyžadována přítomnost položek: typ publikace, jazyk, název publikace a autoři. Což je zcela korektní k prvotnímu návrhu.

Třída Load:

Vstupním bodem zadavatelského rozhraní je soubor /load/index.php. Při spuštění tohoto skriptu se vytvoří instance třídy Load a konstruktoru jsou předány následující parametry:

1. data z odeslaného formuláře (POST)
2. login zadavatele (zatím provizorní řešení)
3. úroveň kontroly formulářů
4. povolené přípony pro vkládané soubory

Data z odeslaného formuláře jsou napřed ošetřena na výskyt uvozovek a znaků backslash. Funkcionalita třídy Load je odstartována metodou execute(). Uvnitř je řídící kód, který na základě elementárních dat rozhoduje o akcích, jenž se mají vykonat. (Např. se zde rozhodne, který ze 3 formulářů se má vlastně zobrazit nebo jakou akci vykonat, když od zadavatele přijde požadavek na vložení nového autora, kategorie nebo vydavatele, dále obsahuje ovladače výjimek atd).

Logování chyb:

Ve třídě Load se používají 2 typy výjimek. První Typu Exception, je vytvářena, když dojde k chybě, kterou může zadavatel nějakým způsobem napravit. Taková výjimka je vyvolána např. pokud nevyplní všechny povinné údaje nebo není formát údajů správně. Když je výjimka tohoto typu odchycena v metodě execute, vždy je zpráva o této chybě dosazena do šablony daného formuláře a zadavateli se zobrazí červeně. Druhým typem výjimky je SException (kód je v /common/SException.inc.php), jenž je odvozena od základní třídy Exception. SException znamená z uživatelského pohledu neřešitelný problém, který nejde opravit. Je napsána tak, aby sama logovala chybu do souboru /log/buglist.log. Do souboru se ukládají data v následujícím pořadí: datum, čas, text zprávy, název souboru, ve kterém vznikla výjimka a IP adresa klienta. Smyslem zavedení tohoto logu je hlavně snaha o identifikaci chyb a možných anomálií v provozu databáze.

Vkládání autorů, kategorií a vydavatelů:

Ve formuláři číslo 1, lze nejen vybírat z nabízeného seznamu autorů a kategorií, ale také do tohoto seznamu přidávat vlastní položky. Přidávání autorů je přímé, tzn. zadavatel pomocí formuláře přímo vkládá nové autory do databáze, podobně se to děje i ve formuláři č. 2 se vkládáním nových vydavatelů. Položky, které se do databáze takto vloží obsahují příznak confirmed=0. To z důvodu aby neschválené položky mohly být identifikovány a byly zobrazovány pouze a jen uživatelům, kteří je do databáze vložili. Neschválení autoři a vydavatelé mohou být mazáni uživateli, kteří je do databáze vložili. Po schválení položek administrátorem se položky stanou viditelnými pro všechny a nebude je již moci nikdo mazat. Podobně je tomu i s vkládanými kategoriemi, jen s tím rozdílem, že nově přidané kategorie se do databáze zapíší až nakonec, spolu s novou publikací. Celé by to šlo udělat i jinak, třeba, že by se všichni autoři, vydavatelé a kategorie vkládali až nakonec, spolu s novou publikací. A nebo by šlo aby se tyto položky vkládaly hned do databáze ještě před uložením celé publikace, oba přístupy jsou možné a nic nebrání tomu, aby to tak mohly být někdy implementovány.

Problém s duplikacemi:

Podívejme se do databázového modelu a všimněme si, že všechny tabulky kromě dekompozičních entit (autor_has_publication, categories_has_publication, attrib_storage), kromě tabulky publication a submitter obsahují sloupce submitter_id a confirmed. Sloupec submitter_id jednoznačně identifikuje zadavatele publikace a confirmed je příznak, zda je položka schválená administrátorem. Skrývání neschválených položek přináší určitý problém. Je pravděpodobné, že dva různí zadavatelé vloží do databáze vydavatele, autory, nebo kategorie, se stejným názvem – to proto, ještě neschválená položka, vložená prvním zadavatelem není viditelná zadavateli č. 2, ten v domnění, že položka neexistuje vloží také tu samou do databáze. V databázi jsou potom uloženy 2 publikace a každá z nich odkazuje např. na autora se stejným jménem ale s různým id.

Tento přístup, který umožňuje vkládání neschválených a následné schvalování atributů přímo vyžaduje aby položky (publisher.name, author.name, author.middlename, author.surname, categories.name a attributes.name) nebyly unique. Tento přístup je na jednu stranu výhodný (v zadavatelském rozhraní se nebudou v seznamech autorů, vydavatelů a kategorií objevovat nesmysly nebo nezkontrolované překlepy), ale na druhou stranu přináší komplikaci, jak se vypořádat s duplikacemi, které mohou tak snadno vzniknout a je nutné je řešit v administrační části systému.

2. a 3. formulář:

Formulář č. 2 se odvíjí od obsahu 1. formuláře. V případě, že typ je kniha, bude se formulář lišit v prvních třech polích (ISBN, rok vydání, počet stránek). Je-li to typ časopis (článek), bude se lišit v prvních 5 polích (ISSN, rok a měsíc vydání, vol., no., stránky od – do). Poslední typ – formulář pro konferenci, se od předchozích bude lišit v prvních 3 položkách (ISBN/ISSN, datum konání, stránky od – do). Všechny jmenované položky se ukládají do tabulky publication, proto je evidentní, že některé z atributů tabulky publication už z principu nebudou vyplněny.

Za těmito vstupními poli následuje prvek select pro vybrání vydavatele, jak už bylo výše zmíněno. V neposlední řadě je zadavateli nabídnuta možnost přidat ke své publikaci některý atribut z množiny schválených rozšířených atributů. A jistým rozšířením a posledním prvkem 2. formuláře je možnost zadavatele, navrhnout vlastní rozšířený atribut.

Rozšířené atributy se ve 2. formuláři přidávají podobně jako kategorie – napřed se uloží do session proměnné tmp_attr a teprve až na konci se do databáze uloží spolu s publikací.

Ve 3. formuláři se pouze zobrazí souhrn, jak jsou která pole publikace vyplněna a je zadavateli nabídnut formulář pro vložení souborů. Počet vkládaných souborů není omezen neboť počet polí formuláře se automaticky díky javascriptu zvětšuje podle potřeb zadavatele. Když zadavatel odešle tento poslední formulář kliknutím na tlačítko „dokončit“, zavolá se metoda finish, jenž se postará o vložení publikace do databáze.

Uložení publikace:

Ukládání publikace je úkon, který vyžaduje, aby byly vloženy záznamy do více než 1 tabulky (je možné, že uložení jedné publikace bude vyžadovat zápis až do 5 různých tabulek najednou). Ochrana konzistence databáze je zajištěna transakčním cyklem, který

začíná v metodě finish příkazem autoCommit(false) a končí příkazem commit(). Všechno se děje nad abstraktní vrstvou databázové vrstvy PEAR::DB. Úplně naposled se vloží přidané soubory do datového úložiště v adresáři /storage/. Soubory jednotlivých publikací jsou pak ještě děleny do podadresářů pojmenovaných pole id publikace. Samotné ukládání souborů má na starosti metoda save_documents, ta napřed zjistí, zda podadresář existuje. Když ne, vytvoří ho a uloží do něj všechny odeslané soubory. Soubory jsou ukládány se stejným jménem jako na klientském počítači. Pro přenos souborů z adresáře TEMP do cílového umístění je použita PHP funkce move_uploaded_file.

Kontrola údajů:

Kontrola údajů vkládaných zadavatelem má dvě úrovně. První úroveň je zajišťována javascriptovým kódem umístěným v šablonách load_st1.tpl a load_st2.tpl, tato kontrola je výhodná proto, že se hodnoty zkontrolují ještě dříve než se odešlou na server. Druhá úroveň je zajištěna PHP kódem na straně serveru, konkrétně jsou to metody validate_st1, validate_st2 a validate_st3 ze třídy Load. Kontrola na straně serveru zafunguje většinou v případě, kdy klientský prohlížeč nepodporuje javascript. Například hodnoty ISBN a ISSN, ačkoliv nejsou povinně vyžadovány, tak ale když je zadavatel odešle, regulárními výrazy aplikace zkontroluje, zda jsou validní. Podporovány jsou jak ISBN 10 tak ISBN 13, viz. metoda validate_st2. Javascriptová validace 3. formuláře je bezpředmětná, ale je nutné ji provést PHP kódem na serverové straně, tam je třeba zkontrolovat, zda má soubor správnou příponu a jestli byl opravdu uploadován přes http, viz. metoda validate_st3.

Autorizace zadavatelů:

V současné verzi je těžké rozhodnout, jak rozlišit identitu zadavatelů. Provizorní řešení k testovacím účelům zatím je nastavení proměnné $current_user v souboru /common/common.inc.php, neboť se počítá s napojením na centrální databázi uživatelů akademické sítě FEL. Tabulka submitter dočasně supluje funkci centrální databáze uživatelů a slouží dostatečně k testovacím účelům. Autorizace zadavatelů je odpojena, ale její zavedení v ostrém provozu nebude problém. Ve zkušební databázi (soubor s testovacími daty je obsažen na přiloženém CD) jsou uloženy 4 testovací položky znamenající identitu 4 uživatelů: admin, guest, vodicm1 a fiserp, které je možné použít k ověření plné funkčnosti rozhraní pro zadavatele.

Nestandardní MySQL funkce:

Pro pořádek zde uvádím, které SQL dotazy v souboru /load/Load.inc.php obsahují nestandardní MySQL funkci CONCAT_WS. zde je seznam metod ve kterých se nacházejí:

- metoda display_st1
- metoda display_st3

3.4 Implementace administračního rozhraní

Administrační rozhraní je řízeno celkem šesti třídami (AdminT12, AdminT3 .. AdminT7) z adresáře /admin/. všechny jsou finální a potomky společné nadtřídy Admin. O tom, kterou třídu použít, se rozhoduje pomocí proměnné tab, odeslané metodou GET. Vizuální rozhraní je vlastně 7 přepínatelných záložek. Šablony tohoto rozhraní se nacházejí stejně jako všechny ostatní šablony v adresáři /templates/. Jsou používány session proměnné, proto musí být zapnuté cookies (inicializace sezení session_start() je v souboru /common/common.inc.php). V případě vypnutých cookies, nebude úplně korektně fungovat formulář pro editaci publikace v záložce 1 a 2.

Třída Admin:

Vstupním bodem administračního rozhraní je soubor /admin/index.php. Na základě hodnoty proměnné parametru tab z URL se inicializuje některý z objektů, jehož nadtřída je abstraktní třída Admin, která definuje konstruktor s 5 parametry v tomto pořadí:

1. data POST
2. data GET
3. login
4. heslo
5. úroveň kontroly formulářů (0 nebo 1)

Úroveň kontroly formulářů má být nastavena na 0. Strict mode, který aktivujete předáváním hodnoty 1, je uvažován pouze pro budoucí rozšíření, jeho použití ale nemá zatím velký význam, stejně jako u rozhraní pro zadavatele (viz. 3.3). Zbývající metody jsou: abstraktní metoda execute, kterou musí definovat všichni potomci a finální metoda authenticate. Authenticate provádí http autorizaci a je vždy volána z konstruktoru.

Manipulace s publikacemi:

Třída AdminT12 je společná pro první i druhou záložku. Vyplývá to z toho, že v obou záložkách je zapotřebí stejná funkcionalita, jen s tím rozdílem, že 1. má na starosti neschválené a 2. schválené publikace. Pro obě záložky se tedy inicializuje objekt stejné třídy, pouze se předá do konstruktoru jiná hodnota parametru $tab. (třída AdminT12 přepisuje konstruktor z nadtřídy Admin a přidává navíc parametr $tab).

Třída AdminT12 umožňuje schvalovat, mazat a editovat veškeré publikace. Když administrátor zvolí v nabídce k některou z publikací, otevře se mu náhled na publikaci a nabídnou se další možnosti – tuto funkcionalitu zajišťuje metoda display_publ_list_short(). Zobrazení formuláře pro editaci publikace má zase na starosti metoda display_modify_publication(). Podobně jako je tomu u rozhraní pro zadavatele, hlavní rozhodovací konstrukce se nachází v metodě execute. Pokud aplikace dostane přes parametry GET id publikace, spustí metodu display_pub_list_short, která zobrazí na výstup krátký přehled o vlastnostech celé publikace a nabídne administrátorovi možnosti: přidat soubor, smazat soubory, schválit (zveřejnit) přidané rozšířené atributy, schválit/označit jako neschválené, smazat a upravit.

Když administrátor zvolí, že chce publikaci upravit, zavolá se metoda display_modify_publication, která zobrazí formulář umožňující úpravy. Krátký výpis i složitější modifikační formulář jsou oba generovány ze šablony /templates/admin_tab12.tpl, to co bude ze šablony vygenerováno závisí na šablonové proměnné $modify. Jedním z nešvarů, který by mohl při editaci publikace vzniknout je možnost, že administrátor změní typ publikace a adekvátně k tomu by se mu nezměnil

jeho editační formulář, neboť atributy 3 typů publikací se v některých detailech liší. Což je např. v zadavatelském rozhraní řešeno 3 stupňovým formulářem. V administračním rozhraní je tento problém řešen javascriptem. Ve formuláři jsou zahrnuta všechna vstupní políčka bez ohledu na typ publikace, ale v závislosti na aktuálním typu publikace se políčka povolují nebo zakazují.

O uložení změn publikace se stará metoda update_publication. Veškeré SQL dotazy jsou uzavřeny v transakci, což zajistí, že když při vykonávání této metody někde v půlce některý z databázových dotazů zhavaruje, všechny předtím provedené změny se neprojeví a databáze se vrátí do původního stavu.

Řešení chybových stavů:

Podobně jako u rozhraní pro zadavatele, i administrační rozhraní používá 2 typy výjimek. První (typu Exception) k řešení potíží, které může administrátor zapojením minimálního úsilí snadno odstranit (problémy jako nevyplněná povinná políčka atd.). Pro závažnější chyby se využívá třídy SAException, definované v souboru /common/SAException.inc.php. Je-li výjimka tohoto typu vyhozena, automaticky při její inicializaci vznikne záznam v souboru /log/admin_buglist.log. Do tohoto souboru se uloží informace v následujícím pořadí: datum, čas, text zprávy, název souboru, ve kterém vznikla výjimka a IP adresa klienta. Smyslem zavedení tohoto logu je hlavně snaha o identifikaci provozních anomálií.

Třídy Admin_T3, AdminT4, AdminT5, AdminT6:

Záložky 3 – 6 jsou si značně podobné. Pomocí nich lze přidávat, upravovat a mazat rozšířené atributy, kategorie,. autory a vydavatele. Nenechám bez povšimnutí, že všechny tyto záložky jsou rozhraním ke správě všech informací, které nebylo vhodné ukládat do tabulky publication, ale jsou umístěné v oddělených tabulkách a s entitou publication jsou všechny kromě tabulky publisher ve vztahu M:N. Jediná tabulka publisher je vůči publication ve vztahu 1:N.

Podrobně se zaměřím pouze na třídu AdminT5, která má na starosti správu autorů. Neboť kód ostatních tříd je velice podobný logiku fungování si lze u nic snadno odvodit. Šablona, ze které se generuje XHTML výstup se jmenuje admin_t5.tpl. Po inicializaci objektu typu AdminT5, se vždy z konstruktoru volá metoda execute. V ní se rozhoduje, jaké akce se na základě získaných parametrů provedou. PHP kód, který zobrazuje na výstup seznam autorů se nachází v metodě display_authors_list. Metoda display_authors_list nezískává z databáze pouze jména autorů, ale zjišťuje i to, zda jsou daní autoři již přiřazeni k některým publikacím či nikoliv.

Výpisy dat je možné řadit. Administrátorovi se nad seznamem hodnot daného sloupce zobrazuje odkaz, na který když klikne, odešle se v parametrech URL proměnná order. Její číslo znamená, který sloupec řadit (viz. metoda display_authors_list). Veškeré zobrazené položky jdou editovat, mazat, schvalovat a přidávat nové, k tomu slouží metody display_edit_author, save_author, delete_author, confirm_authors a add_author.

Správa rozšířených atributů – třída AdminT3:

Za zmínku stojí, že rozšířené atributy, ač mají ve svém datovém úložišti (v tabulce attributes) příznak confirmed, tak tento příznak má spíše význam „public – veřejný“. Administrátor může ponechat některé rozšířené atributy neveřejné (confirmed=0), kliknutím na tlačítko „uložit viditelnost“ se zavolá metoda save_scope a podle toho, u

kterých atributů byly zašktrnuty checkboxy se upraví viditelnost pro všechny rozšířené atributy.

Správa normovaných formátů citací – třída AdminT7:

Třída AdminT7, zobrazuje obsah vybrané šablony z tabulky format v prvku textarea. O zobrazování obsahu se stará metoda display_template. Administrátor kliknutím na tlačítko „uložit“ dá příkaz aplikaci, aby zavolala metodu save_template.

I zde je určité riziko, administrátor musí totiž dbát na validitu jím upravovaných šablon a kontrolovat výstup generovaných citací v uživatelském rozhraní. Zvláště musí dbát na validitu Smarty tagů, neboť stačí jen malá chyba v zápisu tagů a šablonovaní systém Smarty při parsování nevalidní šablony zhavaruje. V tomto případě s důsledkem úplné nebo částečné nefunkčnosti uživatelského rozhraní. Dobrým vylepšením by bylo zajisté donutit Smarty, aby prováděl spolu s ukládáním změn i syntaktickou analýzu šablon. Snadno by se tak dalo přijít na chyby v šablonách mnohem rychleji. Naneštěstí Smarty samotnou syntaktickou analýzu nepodporuje, kvůli dalšímu vylepšení by se tedy muselo sáhnout na kód Smarty a přidat do něj funkci, která bude provádět syntaktickou analýzu. Funkci, která při nalezení první chyby v šabloně neukončí skript, ale vrátí výsledek kontroly syntaxe.

Tabulky v příloze 11.1 obsahují seznamy proměnných, které jsou do šablon z tabulky format standardně systémem přiřazovány.

Nestandardní MySQL funkce:

Pro pořádek zde uvádím, které SQL dotazy obsahují nestandardní MySQL funkci CONCAT_WS. zde je seznam tříd a metod, ve kterých se nacházejí:

- třída AdminT12, metoda display_modify_publication
- třída AdminT12, metoda display_publ_list_short

Způsob získávání posledního vloženého id u autoinkrementačních polí:

V souboru /common/db_fns.inc.php je funkce last_insert_id, která má vracet id posledně vloženého záznamu pro konkrétní databázové spojení. V současné verzi podporuje funkce MySQL a Postgres. V případě potřeby nasazení na jiný databázový stroj se musí funkce vhodně doplnit.

4 Zabezpečení

4.1 Zabezpečení přístupu uživatelů

Navrhl jsem a implementoval aplikaci, která se v budoucnu pravděpodobně bude integrovat do intranetového prostředí ČVUT FEL. Jakým způsobem fakulta přiděluje uživatelům hesla a jak se autorizují její uživatelé nevím. Integrace databáze publikací do tohoto prostředí nebylo mým úkolem. Jediné co bylo nutné udělat, je zabezpečit administrační rozhraní před nepovolanými zraky. Uživatelské rozhraní a rozhraní pro zadavatele zůstávají v této verzi otevřené a nefunguje u nich žádná autorizace. Pouze jsem navrhl jakýsi základní model, jak bude databáze identifikovat své zadavatele (v databázovém modelu tomu odpovídá tabulka submitter). Aplikaci jsem navrhl a funkci autorizačního modelu jsem ověřil tak, aby nebyl problém v dalším rozšiřování databáze o autorizaci zadavatelů resp. i běžných uživatelů.

Zabezpečení přístupu k administračnímu rozhraní se děje pomocí standardní HTTP autentizace typu basic. Zde je mou povinností poznamenat, že metoda basic není bezpečná, jako by byla autorizace pomocí http metody digest nebo ještě lépe pomocí HTTPS. Basic posílá hesla po sítí v otevřeném tvaru a digest (dostupná až v http 1.1) posílá po síti md5 otisky. Na druhou stranu použití https závisí tom, zda ho bude školní server podporovat. Proto jsem zvolil nejjednodušší variantu, která byla k dispozici, metodu basic.

Autorizace v implementaci administračního rozhraní je realizována metodou, která je volána při každém požadavku o vstup do administračního rozhraní (viz 3 Popis implementace).

4.2 Ošetření vstupů systému

Mojí snahou při implementaci, bylo ošetřit vstup od uživatelů tak, aby se v první řadě zamezilo vzniku rizika skriptové a SQL injekce. V druhé řadě bylo nutné zajistit, aby systém neinterpretoval texty obsahující HTML.

Script injecion:

Injekci skriptu jsem vyloučil už od prvopočátku, neboť nikde nedávám útočníkovi možnost aby přes parametry URL adresy, ani POST daty mohl manipulovat s nějakými soubory. Problém s injekcí skriptu by ale mohl nastat v případě, že by útočník skript poslal systému prostřednictvím vstupního formuláře k zadávání publikací. Je zde přeci možnost ke každé publikaci přiložit nějaké soubory. Když bych povolil vkládání jakéhokoliv souboru, zákonitě by musel nastat problém. Neboť kdyby předmětem uploadu byl třeba PHP skript, mohl by útočník převzít nad systémem absolutní kontrolu. Proto ve vstupních formulářích zadavatelského rozhraní testuji typy vkládaných souborů. Nejlepším způsobem bylo, jasně specifikovat jaké typy souborů se mohou vkládat.. Toto nastavení je k dispozici ve skriptu /common/common.inc.php. Záměrně se však netestují typy souborů, které vkládá administrátor.

SQL injection:

Jak jsem se zmínil v úvodním odstavci, bylo pro mě důležité se vyhnout riziku SQL injekce. Dovolím si jmenovat pár bodů, které říkají něco o základních bezpečnostních vlastnostech kombinace PHP s MySQL:

- PHP nepovoluje vykonávání několika SQL dotazů v jednom volání funkce mysql_query(), takže tím odpadá problém s tím, že by útočník vložil za prováděný SQL dotaz svůj vlastní kód. Provádí se vše, co se nachází před prvním středníkem.

- V MySQL neexistuje nic, co by mohlo zavolat externí aplikaci jako je tomu například u MSSQL, takže odpadá spousta dalších potenciálně nebezpečných dotazů.

- Pokud v databází neuchovávám PHP kód který následně pomocí funkce eval() vykonávám, nehrozí ani provedení cizího kódu na serveru.

Aby se tedy zamezilo riziku, bylo zapotřebí zajistit následující: proměnné získávané z XHTML formulářů musejí být ošetřeny tak, aby nebylo možné v nich poslat samotné znaky typu uvozovka a dvojitá uvozovka. Tato filtrace se standartě děje na serverech s PHP interpreterem, se zapnutou funkcí magic quotes přidáváním znaku backslash (\) před speciální znaky. Protože se na to ale nedá spoléhat, vybavil jsem kód vlastní filtrací. Ta funguje tak, že napřed otestuje, jestli jsou magic quotes zapnuté, a když ne, vykoná se na všechny uživatelské proměnné funkce addslashes(). Jelikož se ale přes vstupní formuláře posílají i data typu pole, bylo zapotřebí zajistit, aby se addslashes provedlo i pro všechny prvky pole. K tomu slouží rekurzivní funkce clean_rec definovaná v souboru /common/misc_fns.inc.php.

Už v úvodu analýzy jsem se zmínil o tom, že aplikace je celá založená na objektovém návrhu. Co se týče dat, která do aplikace vstupují, ty jsou předávány metodou POST a GET. Na tato pole hodnot je vždy zavolána funkce clean_rec a jejich výsledky jsou předány konstruktoru. Uvnitř objektů se pracuje vždy a pouze s těmito hodnotami a už se niky neodkazuje na superglobální proměnné $_POST a $_GET. Takto získané hodnoty se ve všech případech ještě vkládají do řetězce s SQL dotazem mezi dvojici jednoduchých uvozovek. V uživatelském a zadavatelském rozhraní se ještě provádí záměrné přetypování všech číselných hodnot na typ int.

HTML ve vstupních datech:

Opomenutí rizika interpretace HTML kódu, který mohl útočník do textu vložit není tak strašlivé jako hrozba SQL injekce, ale i tak je to poměrně závažné bezpečnostní riziko. Filtrovat HTML entity je zapotřebí nejen kvůli útočníkům, ale i kvůli nic netušícím uživatelům, kteří by rádi vkládali znaky třeba „<>” a další jiné. Já jsem problém řešil tak, že data, která uživatel vloží se uchovají v databázi v nezměněné podobě a teprve při jejich výstupu, je hlídán výskyt HTML entit. Někde data filtruji PHP funkcí htmlspecialchars, jinde využívám možností propracovaného systému Smarty a filtraci HTML entit provádím v šablonách. Obecně jsem se snažil aby se HTML filtrovalo hlavně v šablonách. Jinde, kde to není vhodné používám funkci htmlspecialchars.

5 Testy

Aplikaci jsem testoval vložením 30 reálných publikací všech typů (konference, knihy, články). Provedl jsem testy zabezpečení vkládáním speciálních znaků (‘, “, \) a HTML entit.

Na všech stránkách jsem provedl test validity XHTML kódu. Jako validátor jsem použil Tidy Validator 0.7.9.3 [5]**,** který jsem si nainstaloval jako plugin [6] do prohlížeče Mozilla Firefox. Testy jsem provedl s nastavenou úrovní přístupu „normal“. XHTML kód byl označen bezchybným, validátor nehlásil žádné chyby ani varování.

Systém byl testován na prohlížečích Mozilla Firefox 1.5 pro Windows a Linux, Microsoft Internet Explorer 6, a Opera 8 pro Windows v rozlišení 800x600 – 1152x864. Ve všech prohlížečích funguje naprosto spolehlivě. Následně jsem zkusil použít prohlížeč Mozilla Firefox, u kterého jsem vypnul interpret javascriptu. Dokázal jsem, že funkčnost ani bezpečnost systému není absencí javascriptu ohrožena.

Nakonec byl proveden akceptační test 5 člennou skupinou testerů z výpočetního oddělení společnosti Chemopetrol a.s. Jeden z testujících namítal, že rozhraní pro zadavatele by nemělo být přístupné bez autorizace ani pro účely testování, vysvětlil jsem mu však, že autorizace zadavatelů,. není dle slov vedoucího práce P. Fišera mým úkolem. Vedoucí testů ing. Pavel Vítovec (pavel.vitovec@chemopetrol.cz), včetně 5 členné skupiny testerů se k produktu vyjádřili převážně jen pozitivně.

6 Zhodnocení existujících systémů

6.1 Open Source systémy

6.1.1 BibTeX

BibTeX je nástroj pro formátování seznamů referencí používaný LaTeXem. Jeho autoři Oren Patashnik a Leslie Lamport ho přivedli na svět v roce 1985 [7]. BibTeX je vlastně součástí LaTeXu, usnadňuje citovat zdroje v seznamech použité literatury a zobrazovat je způsobem, shodným s patřičnými normami. Efektu je docíleno oddělením bibliografických informací a formátovacích dat. Podobně jako je tomu například v oblasti webdesignu, kde je výhodné dbát na oddělení obsahu od stylů (XHTML a CSS) nebo oddělení obsahu od aplikační logiky (šablonovaní systémy).

Bibliografická informace může být formátována rozdílnými způsoby - v závislosti na tom, který ze standardních citačních stylů nasadíme, k dispozici jsou např. styly APA (American Psychological Association), MLA (Modern Language Association), CMS (the Chicago Manual of Style) a další. V závislosti na souboru stylů, může BibTeX změnit pořadí jmen autorů, vynechávat atributy obsažené v souboru dat (soubor s příponou bib), definovat vlastnosti textu atd.

Systém BibTeX je často používaný pro akademické a vědecké účely ke generování referencí na použitou literaturu. Na internetu je volně ke stažení množství šablon formátů citací, podle různých standardů.

BibTeX umožňuje skladovat všechny informace o použité literatuře v jednom textovém souboru. Potom je snadné se na ně z LaTeXového dokumentu odkazovat a v tomtéž dokumentu generovat citace. Máme-li jeden soubor, ve kterém jsou informace o literatuře uloženy, může být používán i více LaTeXovými dokumenty. Tomuto souboru se říká bibliografická databáze. Tento centralizovaný přístup sdílení bibliografických

dat vede k usnadnění práce.

Také lze mít bibliografických databází víc. Pokud píšete práci v jednom oboru, třeba skripta z oblasti elektrotechniky a další skripta z oblasti fyziky, může mít každý dokument svojí databázi citované literatury. Když píšete práci v oblasti, která pokrývá fyziku i elektrotechniku, můžete oba bibliografické soubory k dokumentu přilinkovat. V dalším textu se podíváme, jak vypadá formát bibliografického souboru, více informací viz manuál [7].

Soubor bibliografických dat (.bib):*

BibTeX používá soubor bibliografických dat, jede o textový soubor obsahující elementy, jimiž mohou být např. články, knihy atd. Bibliografické soubory mají běžně příponu bib. Bibliografické elementy, které může bib soubor obsahovat dělíme na následující typy:

@article
: Novinový článek, nebo článek z časopisu

@book
: Kniha s jednoznačně identifikovatelným vydavatelem

@booklet
: Publikace, která je vytištěna a svázána, ale je bez identifikace vydavatele

@conference
: Zápisy z konference nebo z jednání

@inbook
: Část knihy, kterou může být buďto kapitola, nebo text vymezený rozsahem stran

@incollection
: Část knihy mající svůj vlastní název

@inproceedings
: Článek z konference

@manual
: Technická dokumentace

@masterthesis
: Diplomová práce

@misc
: Nezařazeno

@phdthesis
: Disertační práce

@proceedings
: Sborník prací.

@techreport
: Report publikovaný školou nebo jinou institucí

@unpublished
: Dokument, který má autora i název, ale není publikován

Další elementy jsou tyto, ale nedoporučují se:

@collection
: Kolekce prací, totéž co sborník prací

@patent
: Patent.

Standardní atributy:

address
: Obvykle adresa vydavatele, běžně se vynechává.

annote
: Anotace, běžně však není bibliografickými styly využívána.

author
: Jména a autorů

booktitle
: Název knihy, část, která se cituje

chapter
: Kapitola (nebo sekce)

crossref
: Křížová reference

edition
: Vydání knihy. Má být napsáno slovně s prvním písmenem velkým.

editor
: Jméno editora

howpublished
: Jak bylo publikováno, první písmeno velké.

institution
: Sponzorská instituce

journal
: Název časopisu

key
: Užívá se k řazení, vzájemnému odkazování v dokumentu

month
: Měsíc, ve kterém byla práce publikována. Pro nepublikovanou práci kdy byla napsána.

note
: Any additional information that can help the reader. The first word should be capitalized.

number
: Číslo časopisu nebo práce v sérii. Časopis je obvykle identifikován svazkem a číslem.

organization
: Organizace, která sponzoruje konferenci nebo vydává manuál

pages
: Rozsah stránek

publisher
: Jméno vydavatele

school
: Název školy, kde byla práce napsána

series
: Jména sérií nebo souborů knih.

title
: Název práce

type
: Typ technické zprávy, např. výzkumná zpráva

volume
: Svazek (časopisy nebo vícesvazková kniha)

year
: Rok vydání, u nepublikované práce rok, kdy byla napsána

affiliation
: Vzájemný vztah autorů

abstract
: Abstrakt.

contents
: Obsah

copyright
: Informace o copyrightu

ISBN
: The International Standard Book Number, (identifikace knih)

ISSN
: The International Standard Serial Numer, (identifikace časopisů)

keywords
: Kličová slova

language
: Jazyk dokumentu

location
: Obvykle město, kde se konala konference

price
: Cena dokumentu

size
: Rozměry výtisku

URL
: URL reference na dokument.

Příklad:

Soubor bibliografických dat (***.bib**) může obsahovat následující položky, popisující příručku:

```
@Book{jansa+wolf,
        autor ="Vlastimil Jansa a Josef Wolf",
        title = "Matematika do kapsy",
        publisher = "Grada",
        year = 1989,
        address = "Praha",
        edition = "Druhé přepracované vydání",
        isbn =  "0-422-52754-1"
}
```

*Soubor bibliografických stylů (***.bst**):*

Soubory stylů s příponou bst se píší ve speciálním postfixovém jazyce, kterým se specifikuje jak mají být bibliografické elementy formátovány. Program bibtex formátuje v závislosti na souboru stylů a generuje příkazy systému LaTeX, nicméně existují i soubory stylů, které generují HTML výstup.

LaTeX zpracovává citace příkazem \cite s použitím souboru stylů. Program bibtex generuje dočasný soubor s příponou **.bbl**, jenž obsahuje bibliografickou informaci o citované literatuře.

6.1.2 Webový systém BibAdmin

Když jsem hledal webové systémy podobné tomu, který jsem vyvíjel, zaujal mne systémem BibAdmin [8]. Dovolím si citovat jeho hlavní rysy.

BibAdmin je webový správce bibliografických dat, skládá se z kolekce PHP skriptů, pracující s databází MySQL. Publikace se ukládají do databáze a mohou být přidávány, editovány a mazány prostřednictví webového rozhraní. Je navržen hlavně pro výzkumné týmy, ke správě citací a publikací. Uživatelé mohou vkládat veřejné nebo soukromé bibliografie, které se pak dají použít ke kompilaci LaTeX dokumentů.

Vlastnosti

- Dva profily: první určen pro správu kompletních publikací a druhý ke správě citací. Přístup uživatelů může být diferencován podle toho jestli jde pouze o citace nebo celé publikace
- Vyhledávání podle klíčových slov
- Upload souborů publikací typu PDF a PS a slidů (PDF, PPT)
- Přidávání URL odkazů k citacím
- Download BibTeXových bibliografických databází (souborů s příponou bib)
- Licence GNU General Public Licence (GPL)

Po zběžném prohlédnutí tohoto systému musím konstatovat, že je v něm pár dobrých myšlenek. Např. podporuje formát bib, což se dobře uplatní při psaní LaTeX dokumentů. Citaci je možné stáhnout ve formátu bib databázového souboru. Zadávat publikaci jde buď přímo vložením celého bib elementu (je vyžadována znalost formátu .bib souboru) a nebo intuitivně pomocí formuláře, v němž jsou vyznačeny povinné a nepovinné položky. BibAdmin dělí uživatele na skupiny a na základě jejich oprávnění jim nabízí ke stažení pdf a jiné dokumenty. V administračním rozhraní lze přidělit těmto dokumentům rozsah viditelnosti, určující, kteří uživatelé mohou dokument spatřit.

6.2 Komerční systémy

6.2.1 Biblioscape

Biblioscape [9] od firmy CG Information je software určený pro platformu MS Windows., Je součástí balíku programů, z nichž je dobré jmenovat ještě jeho zjednodušenou, neplacenou verzi BiblioExpress a webový server BiblioWeb. Biblioscape se používá k organizaci dokumentace, referencí a k získávání bibliografických dat na internetu. Biblioscape může být používán ke sbírání referencí na literaturu různých typů, k prohlížení bibliografických zdrojů na internetu a prohlížení citací v různých formátech.

Program jsem zkoušel a působil na mě dobrým dojmem, uživatelské rozhraní je rychlé a přehledné. Citace jsou rozděleny do složek, podle kterých se dělí např. do různých oborů, kategorií atd. Zpravidla v pravé dolní části programu se zobrazuje výsledný formát citace daný některou ze zvolených norem. Dobrou funkcí programu, je možnost vytvářet odkazy na kompletní texty, ať už jde o URL odkazy nebo soubory uložené na lokálním počítači. Úplnou samozřejmostí jsou pokročilé metody vyhledávání, zálohování databází, kontrola duplicit atd. Vkládání nových citací je možné buď ručně nebo pomocí funkce „hot import“. Tato funkce umožňuje importovat citace přímo z webové stránky některého z katalogů (seznam katalogů je v programu zabudován).

Zaměřme se nyní na možnost, že pomocí programu BiblioWeb můžeme publikovat

citace do prostředí internetu. BiblioWeb server generuje katalog publikací přístupný přes protokol http. Uživatelé do něj mohou nejen nahlížet, vyhledávat záznamy, ale mohou i do katalogu přidávat a editovat publikace. Na straně serveru stačí jen spustit soubor BiblioWeb.exe. Nastavení serveru je celkem intuitivní záležitost, webové rozhraní, které generuje, je uživatelsky přívětivé a částečně se blíží svojí funkcionalitou samotnému programu Biblioscape. V katalogu lze vyhledávat, filtrovat, zobrazovat publikace, editovat atd.

Co se týče typů publikací a jejich povinných a nepovinných údajů, Biblioscape si vede svůj vlastní standard, oproti BibTeXu rozděluje třeba typ article na journal article, magazine article a newspaper article. Přidává některé nové typy publikací, jako např. map, electronic source, film or broadcast, statute, hearing, artwork a další.

6.2.2 Ostatní

V minulé podkapitole jsem popsal hlavní výhody programu Biblioscape. Protože Biblioscape není jediný z programů tohoto typu, je mojí povinností, uvést zde příklady některých dalších. Shrnující přehled některých systémů, které zde zmiňuji viz [10].

EndNote:

Endnote [11] běží na platformě MS Windows. Je dílem společnosti Thomson ISI Researchsoft. EndNote je jedním z mnoha bibliografických produktů pocházejících z této firmy. K hlavním rysům patří např.:

- Vkládání obrázků a jiných souborů k publikacím
- Přístup k několika stům katalogů světových a univerzitních knihoven a bibliografickým databázím pomocí protokolu Z39.50
- 37 předdefinovaných typů citací (knihy, periodika, tabulky, CD apod.), možnost přidávat další
- Pokročilé vyhledávání v knihovně citací s využíváním booleovských operátorů
- více než 1100 předdefinovaných stylů pro formátování citací, možnost tvorby vlastních
- Cite While You Write (vyhledávání a vkládání citací přímo z programu MS Word)
- možnost vystavení knihovny v prostředí internetu
- kontrola pravopisu, slovníky klíčových slov a synonym
- import citací z širokého spektra online i offline zdrojů

WriteNote:
WriteNote [12] je také dílem vývojářů z Thomson ISI Researchsoft. WriteNote je pouze webová služba, která umožňuje archivovat citace na vzdáleném serveru. Ke stažení je nabízen i WriteNote toolbar pro Microsoft Internet Explorer a MS Word. Má v sobě zabudovanou funkci „Cite While You Write“. Jde tak snadno udržovat konzistentní citace v dokumentech Wordu.

V podstatě všechny podobné komerční systémy toho umějí poměrně dost. Vyhledávání v databázích velkých knihoven, je u většiny naprostou samozřejmostí. Podporují hodně formátů pro import/export publikací. Podporují funkci „Cite While You Write“, která umožňuje uživatelům Wordu rozumným způsobem udržovat seznam referencí.

Nebudu se dál zabývat detailním popisem dalších systémů, ale pozastavím se ještě u toho, jak tyto programy získávají bibliografické informace na internetu. Byla zde

zmínka o protokolu Z39.50, uvedu zde krátký popis, který ho ve zkratce charakterizuje, více viz. [13].

ANSI/NISO Z39.50 definuje standardní způsob pro komunikaci dvou počítačů za účelem získávání informací z rozsáhlých informačních databází. Z39.50 podporuje výměnu dat v distribuovaném prostředí, kde klientský počítač zadává dotazy a server na ně odpovídá. Z39.50 je široce používán v prostředí knihoven. Kořeny tohoto protokolu sahají do roku 1970, dalo by se tedy mluvit o „předwebové“ technologii. Současné výzkumy se snaží tento protokol nahradit, ale přitom si ponechat dotazovací syntaxi původního Z39.50. Např. protokol SRU, který posílá příkazy přes URL HTTP protokolem. Nebo protokol SRW, který používá SOAP. Oba protokoly očekávají výsledky v XML formátu.

6.3 Příklady normovaných citací

V předchozím textu jsem se několikrát zmínil o normách zápisu citací, řekl jsem, že jich je nepřeberné množství, ale mojí povinností je, zde uvést alespoň některé příklady citací formátované dle následujících norem.

MLA:

McCleary, B. V. "Enzymatic Modification of Plant Polysaccharides." Int. J. Macromol 8 (1986): 349-354.

APA:

McCleary, B. V. (1986). Enzymatic Modification of Plant Polysaccharides. Int. J. Macromol, 8, 349-354.

ACS:

McCleary, B. V. *Int. J. Macromol* **1986,** 8, 349.

IEEE:

R. W. Lucky, “Automatic equalization for digital communication,” *Bell Syst. Tech. J.*, vol. 44, no. 4, pp. 547–588, Apr. 1965.

ČSN:

RUSSELL, B.: Vagueness. Australian J. Phil. 1., 1923, pp. 84-92.

7 Porovnání s existujícími systémy

V předchozí kapitole jsem hovořil o systémech ke správě referencí. Je vidět, že některé mohou být používány pouze jedním uživatelem při generování referencí na použitou literaturu, některé jsou použitelné i v distribuovaném prostředí. Zástupcem první skupiny je BibTeX nebo proprietární systém BiblioExpress. Do druhé skupiny můžeme zařadit multiuživatelský open source BibAdmin a komerční produkt BiblioWeb.

Systém, který jsem implementoval, se s ohledem na prostředí do kterého je nasazen dá porovnat s BibAdminem. Ten je totiž implementován jako webová aplikace, postavená na PHP a MySQL. BibAdmin ukládá publikace jako elementy používané systémem BibTeX v bibliografických soborech **bib** do databáze MySQL. Syntaxe elementů je stejná. Obrovskou výhodou je, že uživatel jediným kliknutím získá soubor obsahující popis publikace v BibTeXovém formátu.

Vyhledávání v BibAdminu je koncipováno trochu odlišným způsobem. Na rozdíl od mojí implementace BibAdmin nemůže hledat zároveň v názvech publikací a definovat filtry do jakých kategorií, resp. od jakých autorů se má vyhledávat. BibAdmin má pouze jedno vstupní pole na zadání hledaného textu a selectem vedle něj, se má určit v čem vyhledávat (jména autorů, publikací, konferencí, časopisů a kategorií). Součástí BibAdmina je katalog, ve kterém můžeme dát kliknutím příkaz, aby se zobrazily publikace z daného roku atd.

Nedostatek, kterým BibAdmin trpí je, že není možné změnit formát citací jinak, než přímou editací zdrojového PHP kódu. Aby BibAdmin plně fungoval, musí být u klienta zapnut javascript. Za větší nedostatek považuji absenci jakékoliv dokumentace, ať už dokumentace kódu nebo instalační či uživatelské příručky. Žádný s těchto textů není k dispozici. PHP kód obsahuje minimum komentářů, nejsou používány objekty ani šablony, HTML a PHP kód jsou kombinovány do sebe, což působí nepřehledně. Na druhou stranu je třeba říct, že myšlenka využít výhody BibTeXu a přenést je do webového prostředí je dobrá.

Moje implementace umožňuje měnit formát zobrazovaných citací dle formátu, který si vybere sám uživatel z nabídky. Administrátor může šablony upravovat, a to bez zásahu do zdrojových kódů aplikace.

Moje implementace dále svým způsobem zjednodušuje náhled na typy publikací. Protože rozsáhlý seznam typů publikací jak jej známe z BibTeXu, zužuje na typ kniha, časopis a konference, ostatní dílčí typy lze pod ně zařadit také, neboť je zadavateli dána možnost vynechat mnoho nepovinných parametrů. Takže například nepublikovanou knihu (v BibTeXu řadíme pod unpublished) zadá pod typ kniha, ale vynechá ISBN. Disertační práci může zadat pod typ kniha s vynecháním ISBN a do poznámky blíže specifikuje, že se jedná o disertační práci. Stejným způsobem lze do poznámek vložit i další libovolné údaje, jako např. poznámku k vydání, kontakt na autora atd.

Výhodu, kterou má můj systém oproti jiným implementacím je to, že lze rozšiřovat množinu vlastností, které publikace mají. V předchozím odstavci jsem naznačil, k čemu je tato funkce dobrá. Administrátor a hlavně i zadavatel mohou vymyslet novou vlastnost, která by šla k publikacím přiřazovat. Tuto vlastnost nazývám rozšířeným atributem, jak již bylo řečeno.

Řazení publikací do kategorií zvládají téměř všechny zde popisované systémy, včetně mojí implementace. Stejně tak i vytváření souborových příloh je u většiny samozřejmostí. Komerční programy (většinou pro platformu Windows) navíc umějí importovat reference na citace ze světových knihoven pomocí protokolu Z39.50. Systém, který jsem implementoval, import neumožňuje.

8 Alternativy dalšího vývoje

Co se týče implementačních záležitostí navrženého systému, bylo mojí maximální snahou ponechat otevřené dveře dalším možným užitečným rozšířením této webové aplikace.

Napadlo mě, že by se mohlo hodit hledáni podle klíčových slov a typických citátů. Znamenalo by to u publikace evidovat ještě další dvě vlastnosti navíc.

Stávající systém umí vyhledávat publikace podle informací, které jsou uložené v databázi. Hodilo by se, kdyby šlo hledat i v samotném textu publikací, tzn. aplikovat fulltextové vyhledávání např. na přiložené soubory pdf.

Zajisté by bylo výhodné mít možnost importu referencí na publikace z databází světových knihoven. Pravděpodobně by to mnohonásobně zvýšilo atraktivitu této aplikace.

Značným vylepšením v administračním rozhraní, by bylo provést změnu záložky na úpravy šablon a donutit Smarty, aby prováděl ještě před s uložením změn syntaktickou analýzu těchto šablon dřív, než je uloží do tabulky format. Snadno by se tak dalo přijít na chyby, ještě předtím, než se cokoliv projeví na výstupu. Smary samotnou syntaktickou analýzu neumožňuje, muselo by se tedy zasáhnout do jejího kódu.

9 Závěr

V této práci se mi povedlo obecně zhodnotit problémy návrhu systému, který má na starosti archivaci, vyhledávání publikací a formátování referencí. Tento krok mi dal dostatek informací k tomu, abych mohl splnit hlavní bod zadání, kterým je implementovat databázi publikací s webovým rozhraním.

Podařilo se mi navrhnout systém tak, aby do něj bylo možné vkládat jakékoliv publikace (včetně elektronických kopií), ať už to jsou publikace vydané s vlastním ISBN nebo jde o publikace bez identifikace vydavatele, články na internetu a jiné. Tím, že systém rozděluje publikace na typ „kniha“, „časopis“ a „konference“ se v podstatě liší od myšlenky BibTeXu, který striktně rozlišuje dokumenty na přesně specifikované, v detailech se lišící typy.

Rozšířil jsem zadání a navrhl aplikaci tak, aby bylo možné publikacím přiřazovat další vlastnosti, kromě těch základních (jako název, autoři, ISBN), které jsou pevně dané. Zvýšil jsem tím její univerzálnost vzhledem k novým i neznámým nárokům na publikace.

Splnil jsem požadavek na to, že si uživatelé budou moci zvolit, jak se jim budou citace zobrazovat - zkrátka, jaký formát bude použit k jejich výpisu. S výhodou jsem k tomu použil šablonovaní systém Smarty. Výsledný formát citací je tedy řízen Smarty šablonami.

Rozdělil jsem uživatele databáze publikací do tří skupin, čímž se zvýšila bezpečnost a důvěryhodnost celého systému. Všechny vkládané publikace vstupují do procesu schvalování, kde uživatel administrátor může kontrolovat jejich validitu a poté co uzná za vhodné publikace učinit viditelnými pro ostatní, publikace schválí.

Výsledkem mojí práce je aplikace pracující s databází MySQL, napsaná pro platformu PHP 5 s využitím knihovny Smarty. V celém procesu implementace jsem dbal na přenositelnost kódů i na ostatní typy relačních databází, k čemuž mi trochu pomohl i modul PEAR DB.

Ze závěrečného zhodnocení již existujících systémů a porovnání s mojí implementací lze vidět, že přínos tato práce rozhodně měla. Ačkoliv je možné použít některý

z velkého množství kvalitních komerčních systémů pro platformu MS Windows i v prostředí internetu (viz. kompaktní systém BiblioWeb v kapitole 6.2.1), tak naproti tomu systémy v PHP, pracující s open source relační databází v této oblasti chybí.

10 Seznam literatury

[1] Daniel Convissor, Pear DB Abstraction Layer http://pear.php.net/package/DB

[2] Joshua Eichorn, PHP Documentator http://www.phpdoc.org/

[3] PhpLib Template Engine http://www.sanisoft.com/phplib/manual/template.php

[4] Smarty Template engine homesite http://smarty.php.net/

[5] Dave Raggett, HTML Tidy library project http://tidy.sourceforge.net/

[6] Marc Gueury, Firefox Tidy Add-on https://addons.mozilla.org/firefox/249/

[7] Dana Jacobson, The BibTeX format
http://www.ecst.csuchico.edu/~jacobsd/bib/formats/bibtex.html

[8] Chelcea Sergiu, BibAdmin – The BibTeX bibliography server
http://www-sop.inria.fr/axis/personnel/Sergiu.Chelcea/software.php?soft=bibtex

[9] Biblioscape – bibliographic software homepage http://www.biblioscape.com

[10] Šimral Břetislav, Citační software. *Ikaros* 2004, vol. 8, no. 11. ISSN 1212-5075

[11] Thomson EndNote homesite http://www.endnote.com

[12] Thomson WriteNote homesite http://www.writenote.com

[13] William Moen, The ANSI/NISO Z39.50 Protocol - Information Retrieval in the Information Infrastructure
http://www.cni.org/pub/NISO/docs/Z39.50-brochure/50.brochure.toc.html

[14] Monte Ohrt and Andrei Zmievski, Smarty Manual, 2005 New Digital Group, Inc.
http://smarty.php.net/distributions/manual/ru/Smarty-2.6.11-docs.pdf

11 Přílohy

11.1 Tabulky proměnných pro šablony formátu citací

Zde je seznam proměnných, které doplňuje aplikace v uživatelském rozhraní do šablon, získaných z tabulky format. Tabulky pomohou při psaní vlastních stylů pro formátování citací. Veškeré hodnoty předávané do těchto šablon, jsou typu string, kromě proměnných $language a $pub_type, které nabývají celočíselných hodnot z intervalu <1,3>. Tabulky jsou zde 3, každá z nich je odlišná a specifická pro každý typ publikace. Počítá se s tím, že šablona by měla být napsána tak, že bude obsahovat 3 sekce rozdělené podmínkami jako „if $pub_type==1 else if $pub_type==2" atd. Uvnitř výkonných bloků pro jednotlivé podmínky už budou rozmístěné zde specifikované proměnné, formátované tak, jak předepisuje norma.

Typ kniha:

$pub_type	Typ publikace, pro knihu vždy 1
$language	Jazyk publikace (1 = česky, 2 = anglicky, 3 = německy)
$title	Název publikace
$city	Město
$country	Země
$state	Stát (použitelné pro USA a Kanadu)
$authors	Seznam autorů
$publisher	Vydavatel
$isbn	ISBN
$year	Rok vydání
$pp	Počet stran
$files	Html odkaz na přiložené soubory

Typ časopis:

$pub_type	Typ publikace, pro časopis vždy 2
$language	Jazyk publikace (1 = česky, 2 = anglicky, 3 = německy)
$title	Název publikace
$city	Město
$country	Země
$state	Stát (použitelné pro USA a Kanadu)
$authors	Seznam autorů
$publisher	Vydavatel
$issn	ISSN
$year	Rok vydání
$issue_date	Datum vydání (měsíc, rok)
$pp	Seznam stránek od – do
$vol	Číslo svazku
$no	Číslo
$files	Html odkaz na přiložené soubory

Typ konference:

$pub_type	Typ publikace, pro konferenci vždy 3
$language	Jazyk publikace (1 = česky, 2 = anglicky, 3 = německy)
$title	Název publikace
$workshop_name	Název konference
$city	Město
$country	Země
$state	Stát (použitelné pro USA a Kanadu)
$authors	Seznam autorů
$publisher	Vydavatel
$isbn	ISBN
$ixxn	Obsahuje řetězce „ISBN“ nebo „ISSN“ v závislosti na typu výtisku konference
$year	Rok vydání
$w_from	Datum zahájení konference
$w_to	Datum ukončení konference
$pp	Seznam stránek od – do
$files	Html odkaz na přiložené soubory

11.2 Uživatelská/instalační příručka

11.2.1 Instalace

Abyste mohli nainstalovat aplikaci na nějaký server, měl by splňovat následující požadavky:

- existence webového serveru, nejlépe Apache 1.3 a vyšší
- podpora PHP 5
- existence databázového stroje MySQL ve verzi 4.1 nebo 5 (PHP musí mít nastavena příslušná extensions mysqli nebo mysql)..

Napřed je nutné zkopírovat obsah adresáře implementation na pevný disk a změnit nastavení konfiguračních souborů. V první řadě se podívejte na obsah souboru /common/db_fns.inc.php. Naleznete zde funkci db_connect a v ní řádek:

```
$dsn = 'mysqli://user:passwd@host/database_name';
```

Změňte tedy DSN podle svých požadavků. Místo mysqli patří mysql, pokud používáte MySQL 5. Za database_name dosaďte název databáze, které nastavte default collation na utf8_czech_ci. Dále si všimněte souboru /install/db.sql. Tento soubor je SQL skript, který obsahuje kompletní strukturu databáze, včetně testovacích dat. Importujte tento skript (např. PhpMyAdminem, nebo použijte jiný vámi preferovaný nástroj).

Otevřete soubor /common/common.inc.php a editujte seznam povolených souborových přípon, login a heslo do administračního rozhraní. Pozor na to, že login není libovolné, musí být zaregistrováno v databázové tabulce submitter, proto raději pro začátek neměňte nastavení řetězce login, ale změňte pouze heslo.

Nyní můžete nakopírovat celý upravený obsah složky implementation do adresáře, kam má přístup webový server (bez adresáře install) a nastavit přístupová práva pro identitu

uživatele se kterou vystupuje webový server, pro zápis do souborů a složek, kam je to zapotřebí. Na systémech typu UNIX nastavte příkazem chmod práva 0777 pro adresáře /log, /storage a /templates_c. Ostatní adresáře by měly mít práva nastavena na 0755. Všechny soubory by měly mít nastavená práva na 0644. Zkontrolujte, zda jsou hodnoty přiřazeny správně.

Instalace je tím kompletní, můžete zkontrolovat funkčnost webovým prohlížečem. Přihlaste se do administračního rozhraní, přepněte na záložku „schválené publikace“, otevřete libovolnou publikaci a zkuste k ní přidat nějaký soubor. Neměla by se zobrazit žádná chybová hláška. Takto nastavená aplikace by měla být zcela funkční.

11.2.2 Rozhraní pro uživatele

Uživatelské rozhraní umožňuje vyhledávat a zobrazovat publikace, které jsou uloženy v databázi. V pravé horní části s nachází menu, přes které se dostanete do ostatních částí aplikace. V levé dolní části se zobrazují citace, včetně odkazů na příslušející souborové přílohy. V pravé dolní liště se nachází: vstupní pole pro vložení hledané fráze(a), rozbalovací seznam kategorií(b), výběr autorů(c) a menu výběru limitu zobrazení(d). Formát zobrazení citací zvolíte prvkem (e). Váš prohlížeč musí mít povoleny cookies. V případě, že jsou cookies vypnuté, po refreshi stránky se vám formát citací nastaví zpátky na výchozí šablonu.

Obrázek 6 - uživatelské rozhraní

a) Do vstupního pole vložte frázi, kterou chcete hledat. Hledání se aplikuje na názvy publikací. Nezáleží na velikosti písmen, diakritika se však rozlišuje. Necháte-li pole frází prázdné, hledat se bude bez ohledu na název publikace.
b) Seznam kategorií můžete kliknutím rozbalit – v tuto chvíli máte před sebou zaškrtávací pole. Můžete tak zvolit, ve kterých kategoriích se má hledat. Jestli nevyberete žádnou z kategorií a kliknete na tlačítko „ok“, hledání se aplikuje na publikace spadající do všech kategorií, dokonce i na ty, které nejsou do žádných kategorií zařazeny.
c) V rozbalovacím menu zvolte hledat publikace od určitého autora. Výsledkem hledání bude opět seznam publikací, kterých byl vybraný člověk autorem.
d) Poslední možností je, že si můžete zvolit počet zobrazených výsledků hledání. Standardně je nastaven na hodnotu 10 výsledků.

Po stisku tlačítka „ok“ se zobrazí nalezené publikace. Když je limit zobrazení nastaven na nějakou hodnotu a počet nalezených položek je vyšší, pro tento případ se zobrazí dole pod seznamem ještě jednoduché navigační menu, pomocí kterého lze publikace stránkovat. U každé publikace je zobrazeno, do kterých kategorií patří. Pokud je k publikaci přiložen soubor budete moci kliknout na odkaz, který na něj vede.

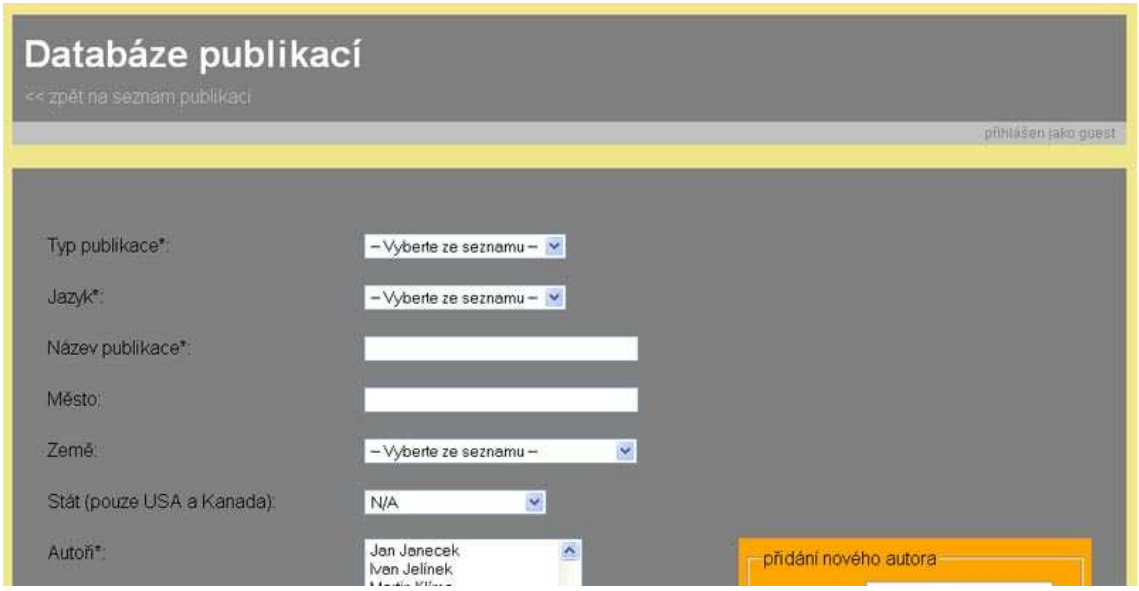
11.2.3 Rozhraní pro zadavatele

Do této části aplikace se dostanete z hlavního menu uživatelského rozhraní. Aby šlo zadat publikaci, musí váš prohlížeč mít povolené cookies. V případě, že nejsou cookies povoleny, publikace nepůjde vůbec vložit, neboť nepůjde identifikovat vaší session. K dispozici máte celkem 3 formuláře, ve kterých budete dotázáni na detaily publikace. Mezi jednotlivými formuláři se lze pohybovat pomocí tlačítek předchozí/další. 1. formulář požaduje všeobecné informace, 2. formulář doplňující informace. Ve 3. formuláři se zobrazí kontrolní zhodnocení všech zadaných informací a je vám nabídnut vstupní prvek pro vložení souborové přílohy.

Položky, které jsou povinné jsou označeny hvězdičkou. Oranžově zvýrazněné prvky jsou určené ke vložení autorů a kategorií (v 1. formuláři) nebo vydavatelů a rozšířených atributů (ve 2. formuláři). Výběr více autorů najednou se provádí myší a současným stiskem klávesy ctrl.

Ve 2. formuláři máte k dispozici možnost vaší publikaci přiřadit další atributy a to buď napsáním textu do vstupního pole již existujícího atributu, nebo přidáním vlastního.

3. formulář vám zobrazí souhrn všech zadaných údajů. K publikaci můžete vložit soubor. Když má prohlížeč povolen javascript, pouhým kliknutím na vstupní pole se zobrazí další pole, takže přiložených souborů můžete odeslat libovolně mnoho.



Obrázek 7 - rozhraní pro zadavatele

11.2.4 Administrační rozhraní

Administrační rozhraní se skládá ze 7 přepínatelných záložek, ve kterých jako administrátor vybíráte zvolenou funkcionalitu. Musí být zapnuté cookies. V případě vypnutých cookies, nebude úplně korektně fungovat formulář pro editaci publikace v záložce 1 a 2.

Zde je uveden význam jednotlivých záložek:

1. schvalování, editace a mazání neschválených publikací
2. zneplatnění, editace a mazání schválených publikací
3. editace, mazání a přidávání rozšířených atributů, editace jejich viditelnosti (veřejný/neveřejný)
4. editace, mazání, schvalování a přidávání kategorií
5. editace, mazání, schvalování a přidávání autorů
6. editace, mazání, schvalování a přidávání vydavatelů
7. editace šablon, které určují formát citace

V první záložce je okamžitě vidět seznam publikací, které byly přidány zadavatelem. Po kliknutí na konkrétní položku se vám zobrazí stručný náhled na vlastnosti publikace. Jsou zobrazeny i odkazy na přiložené soubory s možností přidat další a nebo vymazat všechny. Jestliže zadavatel přiložil k publikaci rozšířený atribut, budete vidět i ten. Vy máte dále na výběr několik možností co s publikací provést.

Obrázek 8 - administrační rozhraní

Upravit publikaci:

Této volby využijete v případě, kdy chcete publikaci poopravit nebo nějak doplnit. Zpřístupní se vám kliknutím na tlačítko „upravit“. Zobrazíte tím formulář připomínající zadavatelské rozhraní. Pracuje se s ním velice podobně (viz 11.2.3) s tím rozdílem, že není dělen do separátních částí, ale zobrazuje přehledně všechny vlastnosti najednou a pole, která nemají být vyplněna (což je pevně dáno typem publikace) jsou chráněna proti zápisu. Změnou typu publikace se opět okamžitě zneplatní položky, které typu publikace nepřísluší. Je nutno podotknout, že aby tato funkce fungovala musí být zapnut javascript. V opačném případě budou přístupná všechna políčka, což ale není žádná tragédie, pouze se z formuláře vytratí jeho elegance. O krok zpět na předchozí obrazovku se můžete vrátit kliknutím na znak „<<“ v levém horním rohu záložky.

Schválit publikaci:

Jste-li přesvědčen o tom, že můžete publikaci schválit, klikněte na stejnojmenné tlačítko. Publikace se přesune mezi schválené, o čemž se okamžitě můžete přesvědčit. Poznamenejme, že si nemusíte dělat starosti s tím, jestli jsou autoři, kategorie a vydavatelé (nepatří sem rozšířené atributy) všichni schváleni. Aplikace rozpozná neschválené záznamy asociované k publikaci a spolu s ní je také schválí. Jak bylo řečeno, neplatí to pro rozšířené atributy, ty budou schváleny jen se zaškrtnutou direktivou „public“.

Smazat publikaci:

Kliknutím na tlačítko „smazat“, úplně odstraníte publikaci ze systému včetně přiložených souborů a záznamů, které byly společně k publikaci přidány. Takže například neschválení autoři, kategorie, rozšířené atributy a vydavatelé, na které publikace odkazuje budou smazáni (jen tak na okraj - nebudou smazáni, odkazuje-li na ně více než 1 publikace).

Záložky pro rozšířené atributy, kategorie a autory fungují všechny vcelku na podobném principu. Zobrazují veškeré záznamy, např. pátá záložka vypisuje jména autorů. Ti se dají řadit kliknutím na horní modrý popisek. Řadí vzestupně podle kritéria, které vyberete. Položky je možné editovat stisknutím tlačítka změnit.

Kliknutím na tlačítko „smazat“, položku trvale odstraníte z databáze a všechny vazby mezi touto položkou a publikacemi, které se na ní odkazovaly budou zrušeny. Tato operace je svým způsobem nevratná, takže je zapotřebí se rozmyslet, zda je opravdu důvod položku mazat (většinou lze vystačit pouze se samotnou editací). O tom, zda můžete položky bez obav mazat nebo ne, vás informuje varovná značka. Toto varování je pro vás důležité hlavně u autorů. Smazáním autora u publikace, která má přiřazena jen jednoho byste mohli způsobit, že publikace ztratí jeden ze svých nejdůležitějších atributů.

Obrázek 9 - správa autorů

Editace šablon:

Konečně poslední možností, kterou jako administrátor disponujete, je editace šablon. Tato funkce je vám přístupná v sedmé záložce pod názvem „šablony“. Tlačítkem uložit po předchozí editaci šablon, změníte šablony, do kterých systém dosazuje texty citací v uživatelském rozhraní. Ke správnému pochopení syntaxe šablon prostudujte

dokument [14] a prohlédněte si tabulku proměnných, které můžete do šablon vložit (viz příloha 11.1).

Práce s rozšířenými atributy:

Ve 3. záložce pod názvem „atributy“ můžete manipulovat s rozšířenými atributy. Dle vašeho uvážení můžete přidávat publikacím další vlastnosti. Když pak budete chtít, aby se tyto atributy zobrazovaly v citacích, upravte šablony v 7. záložce (například prostým přidáním elementu {$_mujatribut}).

11.3 Obsah přiloženého CD

/doc adresář s dokumentací včetně tohoto textu
/implementation zdrojové skripty

Obsah adresáře /implementation:

./admin/ soubory pro administračního rozhraní
./common/ sdílené soubory, knihovny funkcí, tříd atd.
./common/pear soubory Pear frameworku
./configs/ konfigurační soubory pro Smarty
./graphics/ obrázky a ikony
./install/ obsahuje soubory potřebné k instalaci
./libs/ soubory knihovny Smarty
./log/ soubory logů
./load/ soubory rozhraní pro zadávání nových publikací
./storage/ adresář souborového úložiště
./templates/ šablony
./templates_c/ kompilované šablony, automaticky generované třídou Smarty

Popis souorů projektu v adresáři /implementation:

./index.php vstupní bod uživatelské části
./user.inc.php součástí je třída User
./index.css soubor stylů pro vstupní část aplikace
./admin/Admin.inc.php abstraktní třída Admin
./admin/AdminT12.inc.php finální třída AdminT12
./admin/AdminT3.inc.php finální třída AdminT3
./admin/AdminT4.inc.php finální třída AdminT4
./admin/AdminT5.inc.php finální třída AdminT5
./admin/AdminT6.inc.php finální třída AdminT6
./admin/AdminT7.inc.php finální třída AdminT7
./admin/log_info.php zpráva o odhlášení
./admin/logout.php odhlašovací skript
./admin/admin.css soubor stylů pro administrační rozhraní
./admin/index.php vstupní bod administračního rozhraní
./common/common.inc.php soubor globálních definic a konstant
./common/db_fns.inc.php databázové funkce
./common/misc_fns.inc.php specializované funkce
./common/Parser.inc.php třída Parser
./common/SException.inc.php třída SException
./common/SAException.inc.php třída SAException

./load/index.php vstupní bod části aplikace určené pro zadávání publikací
./load/load.css soubor stylů pro zadavatelské rozhraní
./load/Load.inc.php třída Load
./templates/admin_header.tpl záhlaví zadavetelské části
./templates/admin_header.tpl záhlaví administrační části
./templates/admin_tab12.tpl šablona 1. a 2. záložky administračního rozhraní
./templates/admin_tab3.tpl šablona 3. záložky administračního rozhraní
./templates/admin_tab4.tpl šablona 4. záložky administračního rozhraní
./templates/admin_tab5.tpl šablona 5. záložky administračního rozhraní
./templates/admin_tab6.tpl šablona 6. záložky administračního rozhraní
./templates/admin_tab7.tpl šablona 7. záložky administračního rozhraní
./templates/err_401.tpl šablona chybové stránky 401
./templates/load_st1.tpl šablona 1. formuláře zadavatelského rozhraní
./templates/load_st2.tpl šablona 2. formuláře zadavatelského rozhraní
./templates/load_st3.tpl šablona 3. formuláře zadavatelského rozhraní
./templates/load_stFinish.tpl šablona konečného formuláře zadavatelského rozhraní
./templates/user.tpl šablona uživatelského rozhraní