Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Webová databáze konferencí

Petr Svoboda

Vedoucí práce: doc. Ing. Petr Fišer, Ph.D.

13. května 2015

Poděkování

Děkuji svému vedoucímu práce doc. Ing. Petru Fišerovi, Ph.D., oponentovi Ing. Janu Schmidtovi, Ph.D. a Bc. Janu Kubálkovi, autorovi práce, na kterou tato práce navazuje.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 13. května 2015 .....

Odkaz na tuto práci

Svoboda, Petr. *Webová databáze konferencí*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Cílem práce byla tvorba webové databáze konferencí podle vzoru již existující databáze konferencí vyvinuté v rámci závěrečných prací na FIT a FEL, ČVUT. Byl proveden sběr požadavků a návrh doménového modelu. Součástí práce je i analýza dalších možností řešení problému, ze které vyplynulo, že by bylo vhodnější rozšířit webovou databázi publikací, která byla vyvíjena v rámci diplomové práce na FIT, ČVUT. Provedli jsme analýzu a návrh rozšíření této aplikace. Změny jsme zanesli také do ER modelu rozšiřované aplikace. V implementační části jsme pak provedli rozsáhlý refaktoring aplikace a následnou implementaci nových funkcí. Implementovanou funkcionalitu jsme otestovali na úrovni uživatelských testů.

Klíčová slova konference,publikace,web,databáze

Abstract

The main objective of this thesis is to develop a web database of scientific conferences based on an existing solution that has been developed within several master's and bachelor's theses at FIT, CTU and FEL, CTU. At first, requirements analysis and domain model design were made. These analyses were followed by research of other possible solutions. The results showed that

the best way to solve the problem may be extending a web database of publicatoins, which was developed within a master's thesis at FIT, CTU. When the extension of the web publication database was analysed and designed, the ER model was changed appropriately. Within the implementation process, large refactoring was applied to the whole application and new features were implemented. Usability testing was done on newly implemented features.

Keywords conference,publication,web,database

Obsah

Seznam obrázků

Úvod

Po celém světě se koná velké množství konferencí a workshopů s IT tématikou. Na fakultě informatiky ČVUT byl vznesen požadavek na systém, který by shromažďoval informace o různých konferencích, umožnoval správcům doplňovat průběžně nové informace a zároveň by sloužil k informování studentů a pracovníků fakulty o konání konferencí. Takový systém byl v minulosti již vytvořen a vyvíjen. Předmětem této práce bylo jeho rozšíření. Jelikož fakultě slouží též systém pro správu publikací, který pracuje s podobnými informacemi, bylo jedním z hlavních cílů propojit oba systémy, aby nedocházelo k duplicitám a nekonzistencím. Propojením obou aplikací navíc vzniká cenný archiv konferencí a publikací.

Kapitola 1

Cíl práce

Cílem práce je analýza, návrh a implementace SW systému pro evidenci vědeckých a odborných konferencí. Po analýze starého systému však byla zvolena cesta alternativní - rozšíření paralelně vyvíjené aplikace pro správu publikací.

Hlavním pilířem práce bylo propojení stávající webové databáze konferencí s webovou databází publikací. Následovaly pak drobné úpravy ve formě změn databázového modelu, přidání a oprava některých stávajících funkcí.

Mezi hlavní požadavky na novou aplikaci patřily:

- Přidání určitých entit. Např. zavedení samostatné entity pro konferenci.
- Vylepšení správy uživatelů
- Implementace autentizace uživatelů přes fakultní LDAP a později Shibboleth.
- Oprava stránkování záznamů
- Zavedení informace o indexaci sborníků konferencí v publikačních databázích
- Zavedení entity pro workshop ročníku konference
- Propojení s webovou databází publikací

Kapitola 2

Analýza

2.1 Funkční požadavky

Jelikož v práci dále porovnáváme různé možnosti řešení, je třeba shrnout kompletní požadavky na výslednou aplikaci. Poněvadž by bodové vyjádření požadavků nebylo dostatečně detailní, použijeme pro definici požadavků souvislý text. Nejprve analyzujeme stávající webovou databázi konferencí vyvíjenou v rámci závěrečných prací[1][2][3] na FIT a FEL, ČVUT.

2.1.1 Funkcionalita stávající webové databáze konferencí

Aplikace slouží pro vedení údajů o konferencích. Ke konferencím evidujeme pouze jejich ročníky. O správu všech entit se stará pouze správce, běžnému uživateli jsou data dostupná pouze pro čtení.

K ročníku konference jsou vedeny různé atributy (viz obr. 2.1). Každý ročník je možné zařadit do kategorií dvou typů nezávisle na sobě - vlastních kategorií a ACM kategorií[4]. Každému ročníku konference může správce nastavit příznak archivace.

ACM kategorie jsou vedeny stromově (každá kategorie může mít podkategorie). Vlastní kategorie konferencí nejsou vedeny stromově.

Uživatelé se přihlašují přes interní přihlašovací údaje. Není dostupné obnovení přihlašovacích údajů např. zasláním na email. Uživatele může správce přidávat, mazat i editovat. Každému uživateli může správce přiřadit roli. Dále může správce uživatelům nastavit příslušnost do uživatelských skupin, které správce může též libovolně definovat. Uživatelé si mohou změnit heslo a zobrazit informace o svém účtu.

Uživatelským skupinám může správce přiřadit vlastní kategorie konferencí. Stejně tak může správce uživatelské skupiny vytvářet, editovat a mazat.

Uživatel si může nechat zobrazit seznam ročníků konferencí. Výsledky může omezit zadáním vlastních kategorií konferencí či ACM kategorií. Dále může výsledky omezit jen na své osobní konference (viz dále). Výsledky mo-

hou být omezeny pouze na konference, které patří do kategorií, které jsou přiřazeny skupinám uživatele. Ve výpisu ročníků konferencí jsou zvýrazněny blížící se termíny deadline, notifikace a finální verze. Uživatel si může nechat zobrazit jen ročníky konference s nastaveným příznakem archivování. Uživatel též může omezit výsledky podle názvu. Výsledky jsou stránkované a dají se setřídit podle libovolného sloupce.

Zobrazení detailu ročníku konference provede uživatel přechodem ze seznamu ročníků konferencí. Uživatel si zde může přidat konferenci mezi své osobní konference. V detailu ročníku konference vidí uživatel všechny vedené údaje o konferenci. V detailu ročníku konference může správce založit navazující ročník konference a provádět další operace s daným ročníkem konference - tedy editování, mazání, přiřazování kategorií.

2.1.2 Nové funkční požadavky

Ročníky konferencí již na sebe nebudou navazovat řetězově, nýbrž bude založena nová entita pro konferenci jako takovou, která bude nadřazena příslušným ročníkům konference. Tyto konference bude moci administrátor spravovat, ale uživatelům budou zobrazeny stále jen ročníky konferencí.

Ke každému ročníku konference lze určit v jaké veřejné databázi dokumentů (např. SCOPUS, IEEExplore) je její sborník indexován. Dostupné databáze dokumentů spravuje administrátor.

Ročníkům konferencí bude nastavován příznak archivace automaticky, a to jednou za den. Archivovány budou automaticky ročníky z uplynulých let.

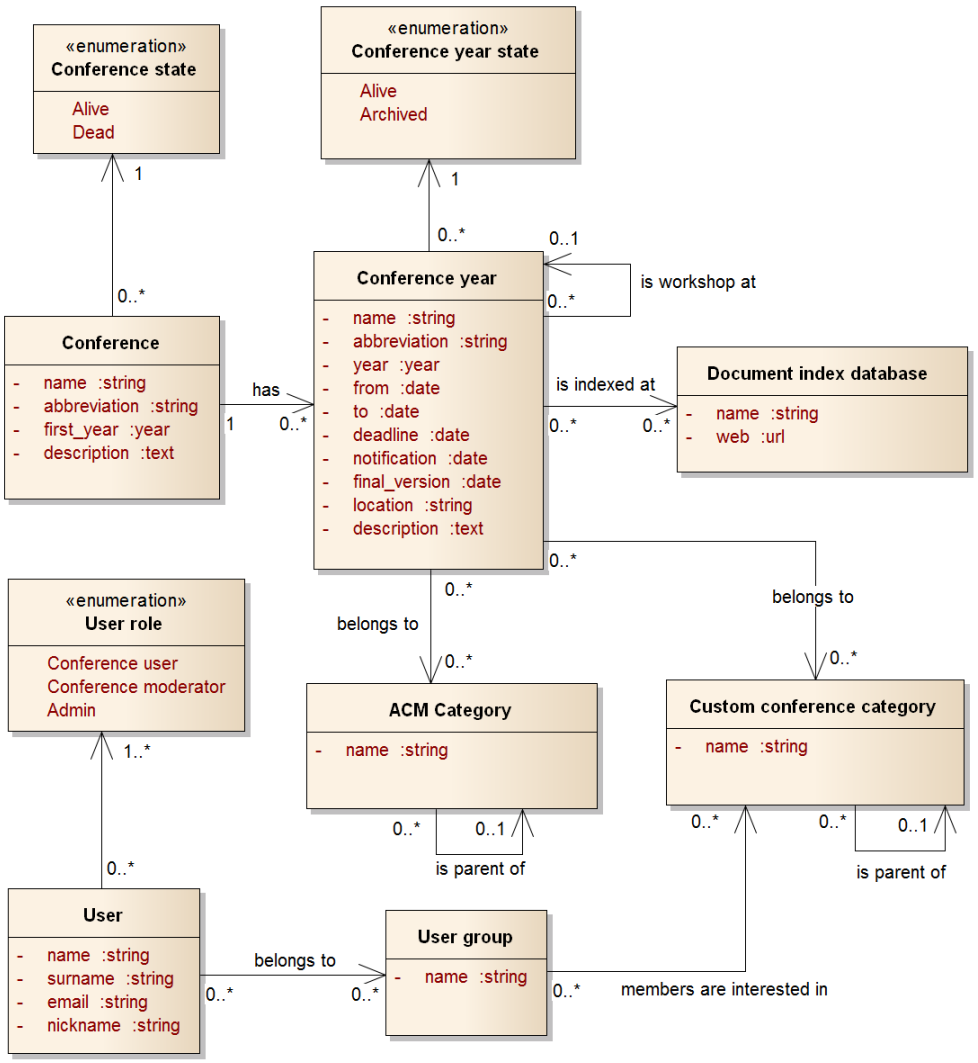
Uživatelé se budou moci přihlašovat přes fakultní autentikační mechanizmus. Přihlašování bude zajištěno přes LDAP a Shibboleth.

Ke každému ročníku konference bude možné přiřadit neomezené množství jiných ročníků konference. Takto přiřazené ročníky konferencí představují workshopy přidružené danému ročníku konference.

Konference a jejich ročníky budou sdílené s databází publikací, spravovat je však bude možné z obou aplikací současně, což představuje riziko zanesní nekonzistencí v datech.

2.2 Analýza problémové domény a vymezení pojmů

Nejprve bylo potřeba seznámit se s pojmy z problémové domény a definovat vztahy mezi jednotlivými entitami. K tomuto seznámení dobře posloužila stávající databáze konferencí, jelikož obsahovala většinu požadované funkcionality (viz 2.1.1).



Obrázek 2.1: Doménový model nové aplikace.

2.2.1 Doménový model

Pro lepší orientaci v problémové doméně zde uvádíme doménový model vytvořený v CASE nástroji Enterprise Architect od společnosti Sparx (obr. 2.1).

2.2.2 Textový popis entit

Konference je hlavním pojmem celé aplikace. Ke konferenci je třeba vést následující údaje:

- Název
- Zkratka
- Popis
- První rok konání konference

Pokud je již konference neaktivní, lze jí nastavit příslušný příznak. Pak ji uvidí v seznamu konferencí jen správce, v administraci bude mít možnost filtrovat neaktivní konference. Běznému uživateli nebudou neaktivní konference zobrazeny. Konference lze zařadit do ACM kategorií[4], toto zařazení určuje správce. Též lze konferenci nastavit příslušnost k interním kategoriím. Každý uživatel si může konferenci přidat do svého seznamu oblíbených konferencí. Poté si uživatel může nechat zobrazit jen ročníky svých oblíbených konferencí. Zároveň je uživateli poskytnuta možnost zobrazení doporučených ročníků konferencí, viz. pojem uživatelské skupiny.

Ročník konference je další důležitý pojem. Ke každé konferenci lze vést neomezené množství jejích ročníků. K ročníku konference evidujeme tyto údaje:

- Název. Lze jej oproti názvům konferencí přetěžovat.
- Zkratka. Lze ji oproti zkratkám konferencí též přetěžovat.
- Ročník
- Datum začátku a konce konání konference
- Datum deadline pro zasílání příspěvků
- Datum notifikace o přijetí příspěvků
- Datum pro zaslání finální verze příspěvků
- Místo konání ročníku konference
- Indexace ve veřejných databázích dokumentů

Již proběhlé ročníky konference se automaticky přesunují do archivu. Uživatel si pak může zvolit, zda se mu mají zobrazit archivované či aktuální konference. Při volbě zobrazení archivovaných ročníků konferencí navíc může uživatel zvolit kvůli lepší přehlednosti zobrazení jen posledních archivovaných ročníků od každé konference. Navíc v problémové doméně figuruje i pojem "Workshop". Workshop lze interně reprezentovat jako ročník konference, u kterého je vedeno, jakému jinému ročníku, jakožto worskhop, přísluší. Ke každému ročníku konference bude možné v aplikaci zvolit i více přidružených workshopů.

Entita uživatel je rozšířením již existující entity v databázi publikací. Evidujeme k ní tyto údaje:

- Jméno
- Příjmení
- Email
- Nickname. Vyplní se přihlašovacím jménem pokud není zadáno jinak.

Uživatele je navíc potřeba autorizovat. Autorizace je řešena přes uživatelské role. Uživatel může mít přiřazených více rolí. Aplikace bude obsahovat tyto role:

- *Administrator*. Uživatel s touto rolí může provádět libovolné operace.
- *Conference moderator*. Uživatel s touto rolí může spravovat konference a jejich ročníky.
- *Conference user*. Uživatel s touto rolí má read-only přístup do databáze konferencí.

Uživatel může patřit do uživatelských skupin definovaných administrátorem. Do skupin zařazuje uživatele správce, uživatelům není poskytnuta informace do kterých skupin patří. Skupin může mít přiřazených uživatel i více. Autentikace uživatelů je poskytována přes interní přístupové údaje či přes fakultní login. Uživatel nemůže využívat více autentizačních mechanizmů najednou.

Uživatelské skupiny spravuje administrátor. Lze vytvořit libovolné množství uživatelských skupin, bez možnosti dědičnosti. Jednotlivé uživatele pak do uživatelských skupin může zařadit administrátor. Hlavní význam uživatelských skupin spočívá v možnosti zobrazení doporučených konferencí pro jednotlivé uživatele. Pro každou skupinu lze určit seznam doporučených interních kategorií konferencí. Uživateli pak můžou být zobrazeny jen doporučené konference na základě jeho zařazení do skupin.

ACM kategorie. V systému lze spravovat ACM kategorie konferencí a řadit je do stromové struktury. Ke každé konferenci je možné vybrat zařazení do libovolného množství ACM kategorií. Uživatel při zobrazení ročníků konferencí může zvolit filtr na určité ACM kategorie.

Interní kategorie konferencí. Význam této entity je totožný s entitou ACM kategorie. Oproti ní lze ale tyto interní kategorie určit jako doporučené pro uživatelské skupiny.

Veřejná databáze dokumentů. Administrátor může v systému spravovat dostupné veřejné databáze dokumentů. Ke každému ročníku konference pak může administrátor určit v jakých databázích je sborník ročníku konference indexován.

2.3 Přehled existujících řešení

V této části jsou uvedena nalezená existující řešení, která by mohla pokrýt požadovanou funkcionalitu.

2.3.1 Conference & Association Management Software

Tento komerční software od společnosti X-CD[5] je určený pro komplexní správu konferencí a veškerých souvisejících náležitostí.

Jednotlivé funkce jsou distribuovány v modulech. Zde je seznam funkcionalit několika z nich:

- Registrace účastníků včetně možnosti online platby
- Registrace a správa vystupujících subjektů
- Tvorba a správa časového plánu konference
- Booking hotelů v místě konání konference
- Nahrávání publikací
- Nativní mobilní aplikace

Společnost X-CD navíc nabízí i další služby spojené s tématikou konferencí, např. pořizování zvukových či obrazových záznamů.

Tento software cílí hlavně na organizátory konferencí. Pro účel fakulty by sice mohl být použitelný, ale jakékoliv úpravy, pokud by byly možné, by byly nákladné. Tento software je pro jednoduchou evidenci konferencí a publikací příliš robustní, navíc je vyvíjen k jiným účelům, než potřebuje fakulta.

2.3.2 Webová databáze konferencí vyvinutá v rámci BP pro ČVUT

Tento systém byl vyvíjen v rámci závěrečných prací od roku 2006[1][2][3]. Procházel řadou úprav, v konečném stavu obsáhl velkou řadu funkcí. Byl nasazen v produkčním prostředí na http://ddd.fit.cvut.cz/Konference/.

Jelikož byl vyvíjen na míru, disponuje všemi funkcemi důležitými pro fakultu. Požadavky na systém ale přibyly, implementace nových funkcí by tedy znamenala rozšíření tohoto systému. Tato cesta rozšíření se jevila jako nejpřijatelnější řešení.

2.4 Možnosti řešení

2.4.1 Rozšíření již existující databáze konferencí

Jako nejlepší volbou by se mohlo jevit rozšíření stávající webové databáze konferencí, na které naposledy pracoval v rámci své bakalářské práce Vít Zdrubecký[1]. Aplikace již disponuje většinou požadovaných funkcí, jednalo by se poté o drobná rozšíření a úpravy.

2.4.1.1 Analýza DB

Uvádíme zde ER model původní databáze konferencí (obr. 2.2).

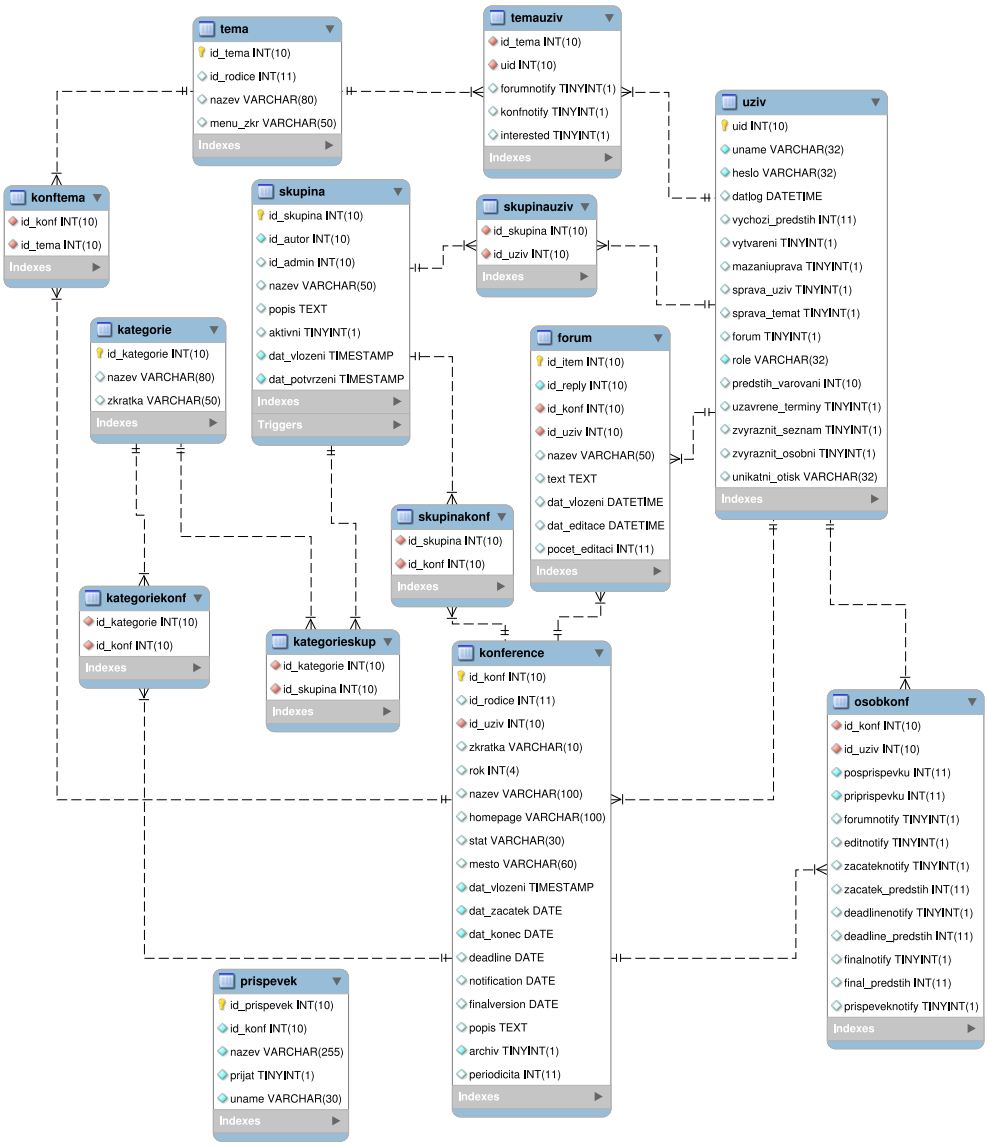
Databázový model byl vhodně navržen. V rámci úprav databázového modelu navrhujeme provést normalizaci. Vyčleníme údaje spojené s oprávněními, autentizací a uživatelským nastavením z tabulky `uziv` do samostatných tabulek. Např. při implementaci LDAP autentikace jsou totiž pole `uname` a `heslo` zbytečná.

Dále by bylo potřeba založit novou tabulku pro entitu konference jako takové. K té by pak existoval vztah 1:N k patřičným ročníkům konference.

Jelikož by byla zavedena i informace o indexaci sborníků ročníků konferencí ve veřejných databázích dokumentů, bylo by třeba založit další dvě tabulky - jednu pro entitu reprezentující databázi veřejných dokumentů a druhou tabulku pro určení vztahů s ročníky konferencí, poněvadž se jedná o vztah M:N.

2.4.1.2 Analýza kódu

Na úvod uvádím ukázku kódu (Listing 2.1) ze souboru `sprava.php` (17 KiB), který se stará o výpis a úpravy konferencí.



Obrázek 2.2: ER model původní databáze konferencí.

Listing 2.1: Ukázka kódu z původní databáze konferencí.

```
else if (isset($_POST["sort_hidden"])){
        $sort = $_POST["sort_hidden"];
        $sipka1 = "&#9650; ";
 }
 //vychozi razeni je podle jmena
 else {
        $sort = 1;
        $sipka1 = "&#9650; ";
 }

 $sort_hidden = '<div><input type="hidden"
        name="sort_hidden" value="' . $sort . '" /></div>';
 $ob = $trideniPole[abs($sort)-1] . ($sort < 0 ? " DESC" : "");

 //strankovani
 $ofs  = (isset($_GET["ofs"])   && is_numeric($_GET["ofs"])
        ? $_GET["ofs"] : 0);
 $rows = (isset($_GET["rows"])  && is_numeric($_GET["rows"])
        ? $_GET["rows"] : 100);
 $rows = (isset($_POST["rows"]) && is_numeric($_POST["rows"])
        ? $_POST["rows"] : $rows);
 $ofs = ($ofs%$rows!=0 ? 0 : $ofs);
 if (mysql_num_rows($dotaz=query("SELECT count(1)
                                  FROM uziv $where")) > 0){
        $radek = mysql_fetch_array($dotaz);
```

Z analýzy zdrojového kódu zde lze formulovat jeho následující vlastnosti:

- Aplikace je psaná v čistém PHP, bez frameworku
- Aplikace je monolitická
- Zdrojový kód je nepřehledný
- Chybí jakákoliv dokumentace, komentáře jsou občas nicneříkající
- Systém sice používá Smarty, ale často generuje HTML kód přímo

2.4.1.3 Zhodnocení

Z analýzy zdrojového kódu vyplývá, že rozšiřování této aplikace by bylo extrémně náročné, jelikož by bylo potřeba pečlivě prostudovat celou aplikaci. Některé úpravy by vyžadovaly změny na mnoha místech, což by znamenalo velkou časovou investici, navíc by se významně zvýšilo riziko zanesení nových chyb do systému.

Výsledkem by pak byla aplikace, která by byla na rozšíření a různé úpravy nebo opravy také velmi náročná.

2.4.2 Tvorba nové aplikace

Další možností řešení by bylo vytvoření zcela nové aplikace od začátku. Tuto možnost navázání na původní aplikaci zmiňuje také její poslední autor, Vít Zdrubecký, v závěru své bakalářské práce[1].

Hlavní přínos spatřuji v tom, že by finální produkt byl dobře navržen od základu a dobře rozšiřitelný. Znamenalo by to okopírování funkcionality původní aplikace a implementaci nových funkcí.

Pro tvorbu tohoto systému bych zvolil z webových technologii PHP v kombinaci s Nette frameworkem, jelikož je dobře navržen a podporuje čistý kód. Také disponuje širokou komunitou uživatelů v České republice. Mocným nástrojem tohoto frameworku je systém komponent, pomocí kterého lze tvořit znovupoužitelné prvky uživatelského rozhraní. Tento framework poskytuje i další nástroje pro práci s odkazy, formuláři, autorizací, šablonami a mnoho dalšího. Veškeré prvky frameworku jsou navrženy tak, aby programátor mohl použít Depency Injection pattern, který zajišťuje nízkou provázanost objektů a usnadňuje testování. Získávání závislostí většinou pak programátor navíc nemusí řešit díky implementaci autowiringu přímo ve frameworku.

Jako RDBMS bych ponechal MySQL, jelikož je dostatečně univerzální, široce používaná, navíc je k dispozici zdarma.

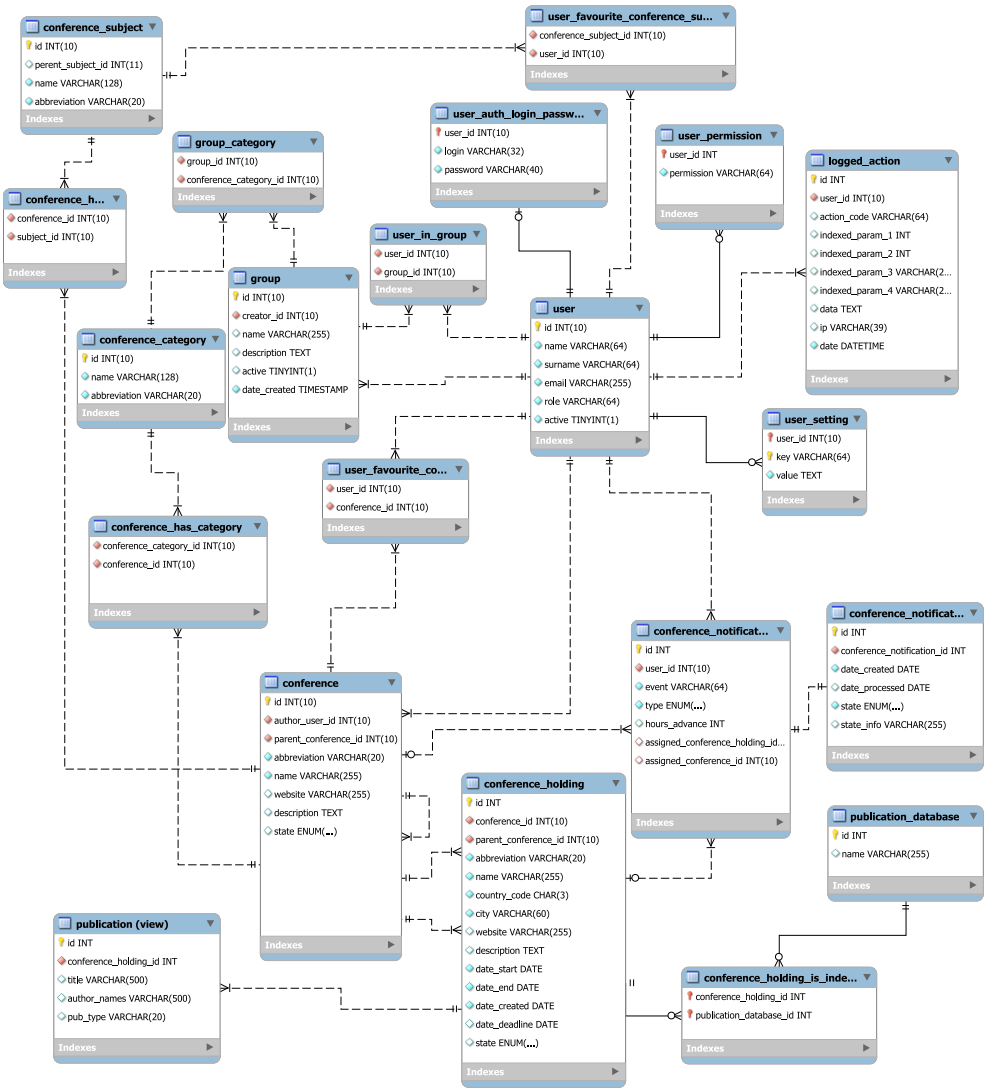
Uvádím zde návrh ER modelu pro novou aplikaci (obr. 2.3). V ER modelu jsou již zohledněny databázové pohledy pro propojení s databází publikací, jak uvedu dále.

Aplikace by měla být propojena s nově vyvíjenou databází publikací. Předmětem sdílení jsou entity konference a ročník konference. Tento problém by mohl být řešen různými způsoby.

První způsob spočívá v zavedení API pro výměnu informací ohledně sdílených entit. To by vyžadovalo definování protokolu a implementaci tohoto API do již vyvíjené aplikace. Toto řešení shledávám však příliš robustním a náročným na implementaci. Navíc jeden z požadavků na aplikaci určuje, že by bylo potřeba provádět změny entit oboustranně. Vznikl by tak problém se zachováním integrity sdílených dat.

Druhým způsobem řešení bylo sdílení částí databází. Tím by byla určena pro databázový systém technologie MySQL, nad kterou je databáze publikací postavena. Přístup ke sdíleným informacím by byl zajištěn pomocí databázových pohledů, pro něž lze definovat oprávnění a lze přes ně provádět i změny dat. Tyto pohledy jsou naznačeny ve schématu uvedeném výše. Tento přístup by byl jednoduchý na implementaci, ve vyvíjené databázi publikací by bylo třeba učinit jen minimum změn. Stále zde ale přetrvává problém zachování integrity dat.

Při volbě vytvoření kompletně nové aplikace bych volil druhý přístup sdílení informací s databází publikací. Zachování integrity dat by však bylo složité, změny integritních omezení by se musely vždy implementovat v obou aplikacích, což by zvyšovalo riziko chyb.



Obrázek 2.3: ER model nové aplikace.

2.4.3 Rozšíření nově vyvíjené databáze publikací

Třetí možností řešení bylo rozšířit paralelně vyvíjenou webovou databázi publikací. Tuto aplikaci vyvíjel v rámcí své diplomové práce Bc. Jan Kubálek[6]. Tato aplikace již byla v době této analýzy z větší míry hotova. Výhodou tohoto řešení je unifikovaný vzhled obou aplikací, možnost centrální správy uživatelů, možnost sdílení prvků UI a logiky, např. vykreslování a zpracovávání formulářů pro manipulaci s různými sdílenými entitami.

Webová databáze publikací využívá technologie PHP, Nette, MySQL a Twitter Bootstrap.

2.4.3.1 Analýza DB

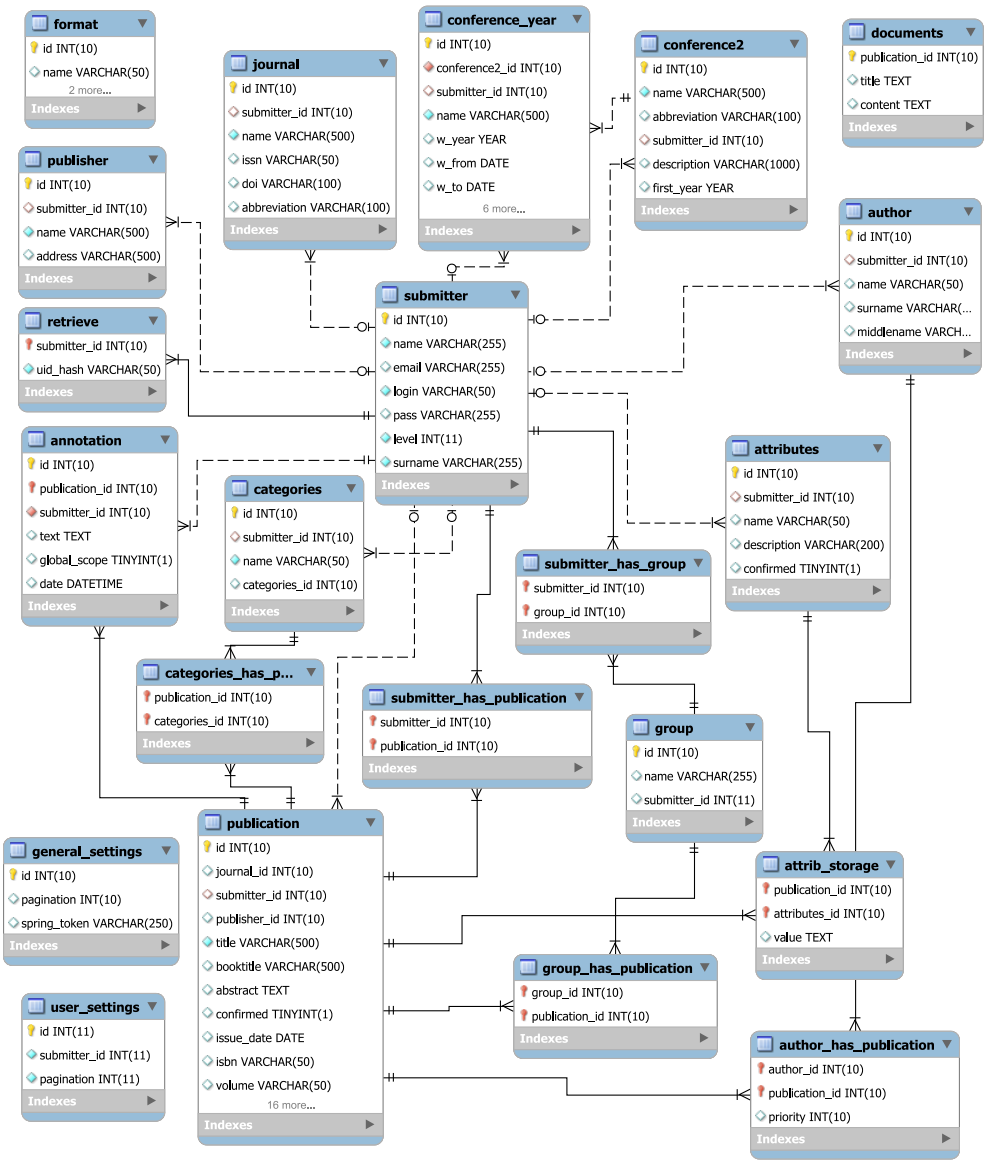
ER model databáze publikací je zachycen na obrázku 2.4. Databáze je vhodně navržená. Rozšíření by znamenalo přidání některých nových tabulek a zařazení nových sloupců do tabulek již existujících. Konkrétní změny jsou popsány v sekci 3.2.1.

2.4.3.2 Analýza kódu

Autor využívá techniky OOP a další techniky typické pro Nette framework. Aplikace vužívá architekturu MVP[7]. Datová vrstva je implementována s pomocí knihovny Nette Database. Vykreslování šablon je realizováno standardně přes Latte template engine. Aplikace hojně využívá AJAX pro vykreslování modálních dialogů a formulářů. Pro autorizaci je využita třída z Nette `Nette\Security\Permission`. V presenterech je předáván celý kontext a není využitý princip Depency Injection[8].

Zároveň lze formulovat i nevýhodné vlastnosti kódu:

- Formuláře nejsou vykreslovány centrálně, ale celé kusy šablon vykreslující daný formulář jsou rozkopírované na všech místech, kde se daný formulář vyskytuje. Změny je pak tedy potřeba provést na více místech.
- Pokud nějaký formulář reaguje na změny provedené v jiném formuláři, stará se o tuto logiku formulář, ve kterém jsou prováděny tyto změny a ne formulář nadřazený.
- Všechny presentery dědí od třídy `BasePresenter`, jejíž kód má velikost 117 KiB. Samotné presentery pak neobsahují skoro žádnou nesdílenou logiku. Díky dědičnosti navíc obsahují nežádoucí metody.
- V aplikaci prakticky nejsou použity komponenty, důsledkem je výskyt opakujícího se kódu (např. řazení záznamů v tabulce, filtr záznamů podle počátečních písmen).
- Kvůli duplicitnímu kódu jsou šablony obrovské a nepřehledné. Největší z nich, `Publication\addnew.latte`, má velikost 184 KiB.



Obrázek 2.4: ER model vyvíjené databáze publikací.

2.4.3.3 Zhodnocení

Řešení rozšíření databáze publikací je vhodné, jelikož případné budoucí úpravy budou jednoduché a bude zajištěna integrita dat. Administrátor navíc bude sdílené entity spravovat z jednoho místa, UI bude jednotné. Pokud má být ale výsledná aplikace dále rozšiřitelná a kód přehledný, bude potřeba provést rozsáhlý refaktoring celého kódu. Na potřebu refaktoringu ukazuje přítomnost velkých tříd, obsáhlých metod a duplicitního kódu, jak uvádí ve své knize o refaktoringu Martin Fowler[9].

2.4.4 Zvolené řešení

Jako konečné řešení jsem zvolil rozšíření nově vyvíjené databáze publikací. Výhoda jednoduchého zachování integrity dat je značná, stejně jako jednolitost UI a centrální přístup do obou aplikací. Tyto výhody dominují nad nevýhodou nutnosti rozsáhlého refaktoringu.

Kapitola 3

Analýza a návrh rozšíření aplikace pro správu publikací

3.1 Use cases

Zde jsou uvedeny nejdůležitější případy užití. Seznam navazuje na výčet případů užití uvedený v diplomové práci autora rozšiřované aplikace[6].

3.1.1 UC1

Popis: Přihlášení do systému

Aktéři: nepřihlášený uživatel, systém

Hlavní scénář:

1. Use case začíná, když nepřihlášený uživatel navštíví webovou aplikaci.
2. Nepřihlášený uživatel zadá přihlašovací údaje a potvrdí odeslání formuláře.
3. Pokud přihlášení proběhne korektně, uživatel je přesměrován na homepage.
4. V opačném případě systém uživateli zobrazí příslušnou chybovou zprávu.

Alternativní scénář:

3. Nepřihlášený uživatel se chce přihlásit přes Shibboleth.
4. Uživatel klikne na tlačítko "Sign in using SSO".
5. Uživatel zadá své fakultní přihlašovací údaje a volbu potvrdí.
6. Pokud přihlášení proběhne korektně, uživatel je přesměrován na homepage.

3.1.2 UC2

Popis: Zobrazení ročníků konferencí

Aktéři: přihlášený uživatel, systém

Hlavní scénář:

1. Use case začíná, když si chce přihlášený uživatel nechat zobrazit ročníky konferencí.
2. Uživatel vybere z menu položku "Conferences".
3. Systém zobrazí veškeré ročníky konference. Výsledky jsou stránkované.
4. Uživatel může zvolit řazení záznamů podle různých sloupců.

3.1.3 UC3

Popis: Přidání konference do oblíbených konferencí

Aktéři: přihlášený uživatel, systém

Podmínky pro spuštění: uživatel se musí nacházet na výpisu ročníků konferencí nebo na detailu ročníku konference

Hlavní scénář:

1. Use case začíná, když si chce přihlášený uživatel přidat konferenci do oblíbených konferencí.
2. Uživatel u příslušného ročníku konference klikne na tlačítko s ikonou hvězdičky.
3. Systém buďto přidá nebo odebere danou konferenci ze seznamu oblíbených konferencí uživatele.

3.1.4 UC4

Popis: Zobrazení oblíbených konferencí uživatele

Aktéři: přihlášený uživatel, systém

Podmínky pro spuštění: uživatel se musí nacházet na výpisu ročníků konferencí

Hlavní scénář:

1. Use case začíná, když si chce přihlášený uživatel nechat zobrazit jen ročníky svých oblíbených konferencí.

2. Uživatel vybere ze sady přepínacích tlačítek možnost "My" s ikonou hvězdy.

3. Systém zobrazí jen ročníky oblíbených konferencí uživatele. Ostatní parametry zobrazení zůstanou zachovány.

3.1.5 UC5

Popis: Zobrazení doporučených konferencí uživatele

Aktéři: přihlášený uživatel, systém

Podmínky pro spuštění: uživatel se musí nacházet na výpisu ročníků konferencí

Hlavní scénář:

1. Use case začíná, když si chce přihlášený uživatel nechat zobrazit jen ročníky doporučených konferencí na základě zařazení uživatele do skupin.

2. Uživatel vybere ze sady přepínacích tlačítek možnost "Suggested".

3. Systém zobrazí jen ročníky doporučených konferencí pro daného uživatele. Ostatní parametry zobrazení zůstanou zachovány.

3.1.6 UC6

Popis: Zobrazení posledních archivovaných ročníků konferencí

Aktéři: přihlášený uživatel, systém

Podmínky pro spuštění: uživatel se musí nacházet na výpisu ročníků konferencí

Hlavní scénář:

1. Use case začíná, když si chce přihlášený uživatel nechat zobrazit jen poslední archivované ročníky konferencí.

2. Uživatel vybere ze sady přepínacích tlačítek možnost "Archived">"Archived - last conference years only".

3. Systém zobrazí jen poslední archivované ročníky konferencí, u kterých není veden žádný aktivní ročník. Ostatní parametry zobrazení zůstanou zachovány.

3.1.7 UC7

Popis: Zobrazení ročníků konferencí podle kategorií konferencí

Aktéři: přihlášený uživatel, systém

Podmínky pro spuštění: uživatel se musí nacházet na výpisu ročníků konferencí

Hlavní scénář:

1. Use case začíná, když chce uživatel zobrazit jen ročníky konferencí, kde konference patří do určitých kategorií.
2. Uživatel klikne na tlačítko "Other filters".
3. Systém zobrazí zaškrtávací seznam ACM kategorií a interních kategorií konferencí.
4. Uživatel vybere které kategorie chce zobrazit a potvrdí volbu.
5. Systém zobrazí požadované ročníky konferencí. Pokud uživatel vybral více kategorií, systém zobrazí ročníky takových konferencí, které patří alespoň do jedné z vybraných kategorií. Ostatní parametry zobrazení zůstanou zachovány.

3.1.8 UC8

Popis: Zobrazení detailu ročníku konference

Aktéři: přihlášený uživatel, systém

Podmínky pro spuštění: uživatel se musí nacházet na výpisu ročníků konferencí

Hlavní scénář:

1. Use case začíná, když si chce uživatel nechat zobrazit detail ročníku konference.
2. Uživatel klikne na název ročníku konference.
3. Systém přesměruje uživatele na stránku detailu konání konference.
4. Uživatel vidí detail ročníku konference, může si nechat zobrazit ostatní ročníky konference, přidružené publikace a workshopy.

3.1.9 UC9

Popis: Zařazení konference do skupin.

Aktéři: administrátor nebo moderátor, systém

Podmínky pro spuštění: správce se nachází v sekci "Administration">"All conferences"

Hlavní scénář:

1. Use case začíná, když správce zakládá nebo edituje konferenci.
2. Správce vybere požadované příslušné ACM a interní kategorie ze zaškrtávacího seznamu.
3. Správce potvrdí volbu.
4. Systém zařadí danou konferenci do zadaných kategorií.

Alternativní scénář:

4. Administrátor zavře modální okno s formulářem pro editaci nebo přidání konference.
5. Systém provedené změny neuloží.

3.1.10 UC10

Popis: Správa přiřazení workshopů k ročníku konference

Aktéři: administrátor nebo moderátor, systém

Podmínky pro spuštění: správce se nachází v sekci "Administration">"All conferences"

Hlavní scénář:

1. Use case začíná, když správce edituje ročníky konference.
2. Správce u požadovaného ročníku klikne na ikonku "W".
3. Systém zobrazí v modálním okně správci seznam přidružených workshopů.
4. V případě akce odpojení ročníku workshopu od ročníku konference
   a) U příslušného workshopu klikne správce na ikonku křížku.
   b) Systém od daného ročníku konference odpojí příslušný workshop.

5. V případě akce přiřazení ročníku workshopu k ročníku konference
   a) Správce zadá název ročníku workshopu do textového pole
   b) Systém správci formou našeptávání zobrazuje výsledky vyhledávání
   c) Správce klikem na požadovaný záznam přiřadí ročník workshopu k danému ročníku konference.

3.1.11 UC11

Popis: Správa uživatelských skupin.

Aktéři: administrátor, systém

Podmínky pro spuštění: správce se nachází v sekci "Administration">"All Conference User Groups"

Hlavní scénář:

1. Use case začíná, když správce potřebuje spravovat uživatelské skupiny.
2. Správce klikne na požadovanou akci (Create/Update/Delete).
3. Systém nechá správci zobrazit modální okno:
   a) s formulářem pro operace v případě zlaožení či editace záznamu.
   b) s potvrzením smazání záznamu.
4. Systém provede požadované změny.

Alternativní scénář:

4. Administrátor zavře modální okno s formulářem pro editaci nebo přidání uživatelské skupiny.
5. Systém provedené změny neuloží.

3.1.12 UC12

Popis: Zařazení uživatele do skupin

Aktéři: administrátor, systém

Podmínky pro spuštění: správce se nachází v sekci "Administration">"All users"

Hlavní scénář:

1. Use case začíná, když správce edituje či zakládá nového uživatele.

2. Správce vybere ze seznamu příslušné uživatelské skupiny
3. Systém provede požadované změny.

Alternativní scénář:

4. Administrátor zavře modální okno s formulářem pro editaci nebo přidání uživatele.
5. Systém provedené změny neuloží.

3.1.13 UC13

Popis: Správa ACM kategorí

Aktéři: administrátor, systém

Podmínky pro spuštění: správce se nachází v sekci "Administration">"All ACM Categories"

Hlavní scénář:

1. Use case začíná, když správce potřebuje spravovat ACM kategorie.
2. Správce klikne na požadovanou akci (Create/Update/Delete).
3. Systém nechá správci zobrazit modální okno:
   a) s formulářem pro operace v případě založení či editace záznamu.
   b) s potvrzením smazání záznamu.
4. Systém provede požadované změny.

Alternativní scénář:

4. Administrátor zavře modální okno s formulářem pro editaci nebo přidání ACM kategorie.
5. Systém provedené změny neuloží.

3.1.14 UC14

Popis: Správa interních kategorí konferencí

Aktéři: administrátor, systém

Podmínky pro spuštění: správce se nachází v sekci "Administration">"All Conference Catgories"

Hlavní scénář:

1. Use case začíná, když správce potřebuje spravovat interní kategorie konferencí.
2. Správce klikne na požadovanou akci (Create/Update/Delete)
3. Systém nechá správci zobrazit modální okno:
   a) s formulářem pro operace v případě založení či editace záznamu.
   b) s potvrzením smazání záznamu.
4. Systém provede požadované změny.

Alternativní scénář:

4. Administrátor zavře modální okno s formulářem pro editaci nebo přidání interní kategorie konferencí.
5. Systém provedené změny neuloží.

3.1.15 UC15

Popis: Správa dostupných veřejných databází dokumentů

Aktéři: administrátor, systém

Podmínky pro spuštění: správce se nachází v sekci "Administration">"All Document Index Databases"

Hlavní scénář:

1. Use case začíná, když správce potřebuje spravovat veřejné databáze dokumentů dostupné v systému.
2. Správce klikne na požadovanou akci (Create/Update/Delete)
3. Systém nechá správci zobrazit modální okno:
   a) s formulářem pro operace v případě založení či editace záznamu.
   b) s potvrzením smazání záznamu.
4. Systém provede požadované změny.

Alternativní scénář:

4. Administrátor zavře modální okno s formulářem pro editaci nebo přidání záznamu.
5. Systém provedené změny neuloží.

3.1.16 UC16

Popis: Určení indexace sborníku ročníku konference ve veřejných databázích dokumentů.

Aktéři: administrátor, systém

Podmínky pro spuštění: správce se nachází v sekci "Administration">"All Conferences"

Hlavní scénář:

1. Use case začíná, když správce zakládá či upravuje ročník konference.
2. Správce vybere požadované databáze publikací, kde má být sborník ročníku konference indexován.
3. Správce potvrdí volbu.
4. Systém provede požadované změny a zavře modální okno.

Alternativní scénář:

4. Administrátor zavře modální okno s formulářem pro editaci nebo přidání záznamu.
5. Systém provedené změny neuloží.

3.2 Návrh DB

Jelikož se věnuji rozšíření jiné aplikace, musí být databáze původní aplikace zachována. Popíši zde změny v původní databázi. Tabulky zastupující sdílené entity jsou v původní databázi již zastoupeny, bude se tedy jednat hlavně o rozšíření těchto tabulek a přidání tabulek nových. Dále bylo potřeba vyřešit problém synchronizace databáze publikací a staré databáze konferencí z pohledu obsahu.

3.2.1 Změny ve struktuře DB

ER model databáze publikací je dostupný na diagramu 2.4.

Jak je vidět, tabulka pro entitu *ročník konference* je v databázi již zastoupena. Tabulku je třeba rozšířit o následující sloupce:

- `abbreviation`. Zastupuje zkratku ročníku konference.
- `parent_id`. Odkazuje na ročník workshopu přidružený konferenci.
- `deadline`. Určuje datum deadline pro zaslání příspěvků.

- `notification`. Určuje datum notifikace o přijetí příspěvků.
- `final_version`. Určuje datum zaslání finální verze příspěvků.
- `state`. Označuje stav konání konference z hlediska archivace. Může nabývat hodnot `'alive'` a `'archived'`.

Tabulka pro entitu *konference* je v databázi publikací též již založena. Nese však název `'conference2'` a tak ji přejmenujeme na `'conference'`. Doplníme navíc sloupec `'state'` určující stav konference z hlediska aktivity. Může nabývat hodnot `'alive'` (konference je aktivní) a `'dead'` (konference je neaktivní).

Tabulka pro zastoupení *uživatelů* již v databázi publikací existuje. Nese název `submitter`. Byly u ní provedeny následující úpravy:

- Vyčlenění autentizačních údajů do zvláštní tabulky `auth_login_password`. Hlavním důvodem této úpravy byla přítomnost více autentizačních mechanizmů a tak by sloupce `login` a `pass` u některých uživatelů nedávaly smysl.
- Vyčlenění autorizačních údajů (sloupce `level`) do zvláštní tabulky `user_role`. Uživatel může mít více přiřazených rolí a tak byla tato úprava nezbytná.
- Vložení sloupce `nickname`. Tento údaj slouží k uchování uživatelského jména (většinou login vzhledem ke zvolenému autentizačnímu mechanizmu).

Byly založeny tabulky pro různé *autentizační mechanizmy*. Konkrétně `auth_login_password`, `auth_ldap` a `auth_shibboleth`.

Také byly nově založeny tabulky pro reprezentaci *ACM kategorií* a *interních kategorií* konferencí. Jedná se o tabulky `acm_category` a `conference_category`. Jelikož mají kategorie vztah ke konferencím M:N, byly založeny i tabulky definující příslušné vztahy - `conference_has_acm_category` a `conference_has_category`. Struktura tabulek reprezentujících ACM a interní kategorie je stejná. Obsahují sloupce `id`, `name` a `parent_id`. Sloupec `parent_id` označuje rodičovskou kategorii, jelikož lze kategorie uspořádat do stromové struktury.

Nově vzniklé tabulky `document_index` a `conference_year_is_indexed` slouží k evidenci *indexace sborníků ročníků konferencí*.

Jelikož v rozšiřované aplikaci již figurují uživatelské skupiny, bylo třeba založit novou entitu *uživatelských skupin - z hlediska konferencí*. Proto byla založena tabulka `cu_groups` (podle Conference User Groups). Kvůli vztahu M:N k interním kategoriím konferencí byla založena i tabulka `cu_group_has_conference_category`. Poněvadž jsou uživatelské skupiny ve vztahu M:N i k uživatelům, byla založena tabulka `submitter_has_cu_group`.

Pro vedení *oblíbených konferencí* uživatelů pak byla založena tabulka `submitter_favourite_conference`.

3.2.2 Přenos dat z původní databáze konferencí

Jelikož ve staré databázi konferencí není zavedena entita pro konferenci jako takovou, ale příslušnost jednotlivých ročníků k dané konferenci je řešena přes jednosměrný spojový seznam ročníků konference, je potřeba určit jakým způsobem budou ročníky roztříděny k jednotlivým konferencím. Nejprve je potřeba analyzovat data ve staré databázi. To učiníme v několika následujících krocích.

3.2.2.1 Zjištění zda jsou u ročníku konferencí vedené zkratky

Vykonáme nad databází následující SQL dotaz:

Listing 3.1: SQL dotaz pro zjištění dostupnosti zkratek u konferencí.

```
SELECT count(*) FROM `konference` WHERE
        zkratka = '' OR zkratka IS NULL;
```

Výsledkem je, že všechny ročníky konferencí mají řádně vyplněnou zkratku.

3.2.2.2 Zjištění nekonzistencí názvů konferencí

Dále je vhodné zjistit, zda k jednotlivým zkratkám existuje více názvů konferencí. Může se jednat o náhodné překlepy, chybějící slova apod. Zkratky, u kterých je evidováno více názvů a příslušné názvy získáme vykonáním následujícího SQL dotazu:

Listing 3.2: SQL dotaz pro zjištění nekonzistence zkratek.

```
SELECT nazev, zkratka FROM konference WHERE zkratka IN (
  SELECT zkratka FROM konference GROUP BY zkratka HAVING
        count(distinct nazev) >= 2
) GROUP BY nazev,zkratka ORDER BY zkratka;
```

Výsledkem je seznam zkratek a názvů, které je potřeba ručně revidovat. Bylo nalezeno 24 záznamů, které je potřeba ručně projít a případně opravit.

3.2.2.3 Zjištění nekonzistencí zkratek konferencí

Jedná se o obměnu předchozího testu. Tento test má naopak za úkol odhalit, zda k jednotlivým názvům existuje více zkratek. Výsledek ukázal, že kromě jednoho záznamu nebyly nalezeny žádné další problematické záznamy.

3.2.2.4 Zjištění stavu provázanosti přes spojový seznam - sloupec `id_rodice` a nalezení duplicitních zkratek

Tento test má za úkol zjistit, jestli jsou poctivě vedeny záznamy spojového seznamu. Zároveň odhalí míru duplicity zkratek u různých konferencí.

Listing 3.3: SQL dotaz pro zjištění stavu provázanosti přes spojový seznam.

```
SELECT zkratka, count(*) as p FROM `konference`
WHERE id_rodice = 0 GROUP BY zkratka HAVING p >= 2;
```

Z výsledku testu lze určit, že pod jednou zkratkou se může skrývat i více různých konferencí a tak nelze uvažovat zkratku jako unikátní identifikátor konference. Záznamy spojového seznamu (sloupec `id_rodice`) jsou však všude vyplněny.

3.2.2.5 Ověření rozřazení ročníků konferencí podle zkratek oproti rozřazení ročníků konferencí podle záznamů spojového seznamu

Výsledkem tohoto testu získame informaci, zda se shoduje rozdělení ročníků konferencí na konference podle zkratek a podle záznamů spojového seznamu. Zároveň ve výsledku budou zastoupeny různé konference s duplicitní zkratkou. Nejprve je potřeba spustit přípravný SQL skript dostupný na přiloženém CD (v cestě `src/database/t-conference-make-groups.sql`). Ten přidá do tabulky pomocný sloupec `id_praotce`, který představuje pro každý ročník konference ID prvního záznamu v řetězovém seznamu. Poté spustíme následující SQL dotaz:

Listing 3.4: SQL dotaz pro zjištění rozdílů rozdělení ročníků konferencí na konference podle zkratek a podle záznamů spojového seznamu.

```
SELECT id_praotce, zkratka FROM `konference`
WHERE zkratka IN (
  SELECT zkratka FROM konference GROUP BY zkratka
  HAVING COUNT(DISTINCT id_praotce) >= 2
)
GROUP BY id_praotce,zkratka ORDER BY zkratka ASC;
```

Výsledek značí, že rozdělení podle zkratek a záznamů spojového seznamu je totožné s výjimkou různých konferencí se stejnou zkratkou. Záznamy jsou tedy z pohledu těchto informací v pořádku a lze je použít pro import.

3.2.2.6 Závěr

V původní databázi bylo potřeba opravit především nekonzistence, které odhalil test 3.2.2.2. Kontrola a opravy byly zajištěny doc. Ing. Petrem Fišerem, Ph.D.

U přenosu dat začněme zkopírováním interních kategorií konferencí a ACM kategorií. Tuto operaci zajistí jednoduchá posloupnost SQL dotazů:

Listing 3.5: SQL dotaz pro přenos ACM kategorií a interních kategorií konferencí

```
INSERT INTO newdb.conference_category (id, name)
 (SELECT id_kategorie AS id, REPLACE(nazev, '&amp;', '&')
 AS name FROM olddb.kategorie);

SET foreign_key_checks = 0;
  INSERT INTO baka.acm_category (id, name, parent_id)
    (SELECT id_tema AS id, nazev AS name,
    id_rodice AS parent_id FROM `tema`);
SET foreign_key_checks = 1;
```

Vypnutí kontroly integritních omezení je zde nezbytné, jelikož jsou v tabulce uloženy hierarchické záznamy a tak při vytváření potomka ještě nemusí existovat rodič. Jelikož tato integritní omezení existovala i v původní databázi, lze očekávat bezchybné záznamy.

Přenos uživatelských skupin ani uživatelů neprovedeme, poněvadž bude k dispozici fakultní autentikační mechanizmus a uživatelské skupiny nebyly ve starém systému využívány.

O přenos samotných ročníků konferencí se stará PHP skript dostupný na přiloženém CD (v cestě `impl/bin/import-conferences-from-old-db.php`). Načte ročníky konferencí a vhodně je rozřadí do konferencí. Poté sloučí tyto záznamy s aktuální databází. Navíc přenese i příslušnost konferencí ke kategoriím.

Kapitola 4

Realizace

4.1 Refaktoring kódu

Z analýzy plyne, že pokud má být výsledná aplikace dále rozšiřitelná a přehledná, musí být u ní proveden rozsáhlý refaktoring kódu. V této sekci uvedeme hlavní úpravy, které byly v kódu provedeny. Všechny úpravy zachovávají původní funkcionalitu.

4.1.1 Zavedení komponent pro CRUD operace nad entitami

CRUD operace jsou operace vytváření, editace, načítání a mazání záznamů. Tato zkratka představuje čtveřici slov: create, read, update, delete. V původní aplikakci veškerou logiku ohledně operací nad entitami obstarávala hlavní třída `BasePresenter`. Mezi tyto operace patří hlavně přidávání a editace entit, dále mazání a často pak zobrazení dalších přidružených entit. Vykreslování příslušných tlačítek a formulářů je pak realizováno přímo v šabloně. Tam navíc není použito žádných bloků pro dědičnost či vkládání kódu, takže příslušné prvky UI jsou rozkopírované do všech šablon, kde jsou použité. Ve výsledku v šablonách nacházíme velké množství duplicitního kódu. Od třídy `BasePresenter` navíc dědily všechny ostatní presentery a tak obsahovaly velké množství nadbytečných metod.

Jako řešení tohoto problému jsme zavedli v systému speciální typ komponenty pro operace nad entitami (umístěné v adresáři `App\CrudComponents`). Pro každou entitu, nad kterými lze v aplikace činit základní CRUD operace na více místech, jsme vytvořili samostatnou komponentu. Tyto komponenty obsahují veškerou funkcionalitu potřebnou k provádění operací uživatelem. Mezi tuto funkcionalitu patří zpracovávání a vykreslování formulářů a tlačítek pro jednotlivé operace.

Uvedeme zde klíčové obecné vlastnosti CRUD komponent. Většina z nich je implementována ve třídě `BaseCrudComponent`, od které dědí ostatní CRUD komponenty.

- Ke každé operaci je možné přidat neomezené množství callbacků, které budou zavolány při provedení dané operace.
- Lze zakázat, případně povolit provádění jednotlivých operací, pouze však na obecné úrovni, nikoliv na úrovni jednotlivých záznamů.
- Závislosti jsou předávány v konstruktoru, je zde použitý princip Depency Injection.
- V komponentách lze zjistit, do kterých částí aplikace má uživatel přístup. Lze tedy snadno zamezit zobrazení informací a příslušných formulářových polí podle toho, zda má uživatel přístup k údajům o konferencích či publikacích.
- Komponent stejného typu může být na stránce více, navíc mohou být do sebe vnořené.
- Šablony pro vykreslování jsou zpravidla tři. Jedna pro vykreslování modálních oken s formuláři (`modals.latte`), druhá pro vykreslování ovládacích prvků spojených s konkrétním záznamem (`controls.latte`) a třetí pro vykreslení tlačítka pro přidání záznamu (`add.latte`).

V presenterech jsou pak tyto komponenty použity jednoduchým způsobem. Kromě továrny, která zajistí jejich vytvoření, je potřeba komponenty vykreslit. Na začátek šablony jednoduše přidáme `{control JMENO_KOMPONENTY}`. Tam, kde potřebujeme tlačítko pro přidání, vložíme `{control JMENO_KOMPONENTY-addButton}`. Kdekoliv, kde potřebujeme provádět operace nad konkrétním záznamem, vložíme `{control JMENO_KOMPONENTY-controls-ID_ZAZNAMU}`. Reakce presenteru na operace lze realizovat registrací potřebných callbacků při vytváření komponenty.

Uvedeme zde výčet zavedených CRUD komponent. Všechny poskytují možnost vytváření, editace a mazání příslušných entit. V následujícím výčtu jsou tedy uvedené jen další funkce.

- `AuthorCrud`. Tato komponenta obstarává manipulaci s autory publikací. Umožňuje zobrazit publikace daných autorů.
- `AttributeCrud`. Komponenta pro manipulaci s vlastními atributy publikací. Umožňuje zobrazit seznam publikací, u kterých je příslušný atribut veden.
- `AnnotationCrud`. Předmětem této komponenty jsou uživatelské poznámky, které jsou zavedeny v původní databázi publikací[6].

- `ConferenceYearCrud`. Tato komponenta implementuje operace nad ročníky konferencí. Pokud má být přístupná možnost založení nového ročníku, je třeba komponentě předat v konstruktoru ID příslušné konference. Komponenta umožňuje též konkrétním ročníkům nastavit/zrušit příznak archivace. Zároveň poskytuje možnost zobrazení publikací spojených s daným ročníkem. Komponenta také umí zobrazit přidružené workshopy a zajišťuje i jejich správu. Všechny operace je možné u komponenty zakázat či povolit, lze navíc definovat zvlášť oprávnění pro zobrazení a správu přidružených workshopů.

- `ConferenceCrud`. Tato komponenta zajišťuje manipulaci s konferencemi. Umožňuje uživatelům zobrazit ročníky konkrétní konference a provádět nad nimi operace, k tomu využívá komponentu `ConferenceYearCrud`. Komponenta umožňuje též uživateli zobrazit přidružené publikace rozdělené podle svých ročníků. Zároveň poskytuje možnost označení konference jako aktivní/neaktivní.

- `CuGroupCrud`. Komponenta pojmenovaná zkratkou místo ConferenceUserCrud. Tato komponenta zajišťuje manipulaci s uživatelskými skupinami z hlediska konferencí.

- `DocumentIndexCrud`. Komponenta, která zaštiťuje operace spojené s dostupnými veřejnými databázemi dokumentů.

- `FormatCrud`. Účelem této komponenty je správa šablon formátů citací pro publikace. Entita pro *formát citace* byla zavedena v původní databázi publikací[6].

- `GroupCrud`. Tato komponenta zajišťuje manipulaci s uživatelskými skupinami z hlediska publikací.

- `Journal`. Komponenta pro správu časopisů. Umožňuje zobrazení souvisejících publikací.

- `Publisher`. Předmětem této komponenty je správa vydavatelů. Umožňuje zobrazit přidružené publikace.

- `User`. Tato komponenta poskytuje operace nad uživatelskými účty. Umožňuje zobrazit publikace konkrétního uživatele.

- `CategoryCurd`. Jedná se o abstraktní komponentu, která nemůže být instanciována. Poskytuje základní funkcionalitu pro operace nad stromovými kategoriemi. Umožňuje navíc operaci pro přidání podkategorie. Poskytuje i základní šablony, které mohou její potomci přepsat či rozšířit.

- `ConferenceCategoryCrud`. Tato třída implementuje operace nad vlastními kategoriemi konferencí. Dědí od třídy `CategoryCrud`.

- `AcmCategoryCrud`. Tato třída implementuje operace nad ACM kategoriemi konferencí. Dědí od třídy `CategoryCrud`.
- `PublicationCategoryCrud`. Tato třída implementuje operace nad kategoriemi publikací. Dědí od třídy `CategoryCrud`.

4.1.2 Přesun logiky ohledně závislosti formulářů

Původně byla veškerá logika zpracování formulářů implementována v třídě `BasePresenter`. Pokud byl formulář použitý na více místech, o reakcích na jeho odesílání na základě jeho umístění rozhodoval jeho vlastní handler. Tedy formulář nutně musel vědět o všech místech, kde byl použitý a zajišťovat patřičné reakce. Uvedeme zde ukázku původního kódu. Kód je zkrácen a oproštěn od nepodstatných detailů.

Listing 4.1: Ukázka kódu pro zpracování formuláře z původní aplikace pro správu publikací.

```
class BasePresenter {
public function conferenceYearFormSucceeded($form) {
  ...
    // $this->name - Name of the presenter
  if ($this->name == "Publication") {
          ...
        $this["publicationAddNewForm"]["conference_year"]->...;
          ...
  } elseif ($this->name == "Conference") {
          ...
    $this->template->conferencesYears = $this->confYears;
          ...
  }
  ...
}
}
```

Lepším řešením by bylo umožnit zadat formuláři callback, který by byl zavolán po úspěšném zpracování formuláře. Pak by formulář mohl být kompletně nezávislý. Přesně takto jsme postupovali při refaktoringu. Refaktorovaný kód z ukázky výše by pak vypadal takto:

Listing 4.2: Ukázka kódu pro zpracování formuláře po refaktoringu původní aplikace pro správu publikací.

```
class ConferenceYearCrud {
  public function formSucceeded($form) {
    ...
    $this->onAdd($someData);
  }
}

class ConferencePresenter {
  public function createComponentCrud(){
    $c = $this->crudFactory->create();
    $c->onAdd[] = function($someData) {
      $this->template->conferencesYears = $this->confYears;
      ...
    }
  }
}

class PublicationPresenter {
  public function createComponentCrud(){
    $c = $this->crudFactory->create();
    $c->onAdd[] = function($someData) {
      $this["publicationAddNewForm"]["conference_year"]->...;
      ...
    }
  }
}
```

4.1.3 Redukce třídy `BasePresenter`

Od třídy `BasePresenter` dědí všechny třídy reprezentující presentery. Samotné presentery však obsahovaly minimum logiky, často jen volaly metody definované v třídě `BasePresenter`. Proto měla třída `BasePresenter` velikost 117 KiB a čítala 2900 řádků. Tuto třídu bylo potřeba výrazně eliminovat a funkcionalitu přesunout do jiných tříd. Jelikož se většina metod týkala CRUD operací, bylo vhodné funkcionalitu přesunout do CRUD komponent (viz 4.1.1). Tím jsme třídu BasePresenter eliminovali jen na opravdu sdílenou množinu metod a atributů.

4.1.4 Zavedení komponenty pro řazení záznamů

V aplikaci bylo vyřešeno řazení příspěvků poněkud těžkopádně. Nejlépe nám poslouží ukázka kódu:

Listing 4.3: Ukázka kódu pro řazení z původní aplikace pro správu publikací.

```
<th>
{if $sort == 'surname' && $order == 'ASC'}
  <a n:href="Author:showall (expand) $params, 'surname', 'DESC'">
                                                  Surname &#9650;</a>
{elseif $sort == 'surname' && $order == 'DESC'}
  <a n:href="Author:showall (expand) $params, 'surname', 'ASC'">
                                                  Surname &#9660;</a>
{else}
  <a n:href="Author:showall (expand) $params, 'surname', 'ASC'">
                                                  Surname</a>
{/if}
</th>
```

Podobný kód se vyskytuje v každé šabloně, kde je potřeba řadit záznamy v tabulce. Jak je vidět, např. změna názvu sloupce, či změna symbolu pro sestupné a vzestupné řazení, by znamenala změnu na více místech a tudíž riziko zanesení chyby. Navíc je kód zbytečně dlouhý a nepřehledný, zvlášť pokud tabulka obsahuje sloupců více.

Proto jsme se rozhodli zavést komponentu pro řazení výsledků. Použili jsme již existující komponentu NasExt/SortingControl[10]. Po změně ten samý kód pro řazení záznamů vypadá takto:

Listing 4.4: Ukázka kódu pro řazení po refaktoringu aplikace pro správu publikací.

```
<th>
        {control sorting, "surname", "Surname"}
</th>
```

Samozřejmě je nutné definovat povolená pole pro řazení a výchozí řazení. Tak činíme při vytváření komponenty pro řazení.

4.1.5 Odstranění statických validátorů

U některých validátorů formulářových polí nám nevadí jejich statická definice. Např. u validátoru číselné hodnoty, maximální délky textu apod. Pokud se ale jedná o validátory, které komunikují s modelovou vrstvou, není jejich statická definice žádoucí. Jelikož CRUD komponenty s formuláři využívají princip depency injection, měl by být u takových validátorů použit také. Nepředpokládá

se, že by formuláře v CRUD komponentách byly využity mimo tyto komponenty. Navíc potřebné validátory většinou nezaberou více než jeden řádek kódu, proto byly validátory přesunuty přímo k příslušným formulářům. Formulář tak vyžaduje v konstruktoru případné závislosti potřebné pro validaci.

4.1.6 Zavedení komponenty pro zobrazení seznamu kategorií

Ve výsledné aplikaci budou k dispozici tři různé druhy kategorií. Vlastní kategorie konferencí, ACM kategorie konferencí a vlastní kategorie publikací. Bylo by tedy vhodné zavést abstraktní komponentu představující stromový seznam kategorií, stejně jako jsme zavedli abstraktní CRUD komponentu `CategoryCrud`.

V původní aplikaci byl seznam kategorií publikací řešený poněkud těžkopádným způsobem. Pro zobrazení seznamu byl využit jQuery plugin jqxTree[11]. Jeho inicializace však probíhala v každé šabloně, kde byl seznam využit, zvlášť. Tím vznikal duplicitní kód.

Vytvořením komponenty pro zobrazení seznamu jsme tento problém vyřešili. Navíc se jedná o abstraktní komponentu, takže jí můžeme použít pro všechny tři druhy kategorií. Komponenta umožňuje nastavit zda má seznam disponovat ovládacími prvky, v tom případě komponenta využije `CategoryCrud` komponentu. Dále lze nastavit zda mají být u položek seznamu přítomny zaškrtávací pole, lze nastavit šířku a výšku seznamu a další parametry seznamu. Od svých potomků komponenta vyžaduje implementaci metod pro získávání záznamů ve standardizovaném formátu a vytváření příslušného potomka `CategoryCrud` komponenty.

Pokud potom chceme seznam se zaškrtávacími políčky použít jako formulářový prvek, vytvoříme daný formulářový prvek jako textové pole a využijeme v šabloně následující konstrukce:

Listing 4.5: Ukázka kódu pro použití seznamu kategorií jako formulářového prvku.

```
<div id="acmCategoryList">
  {control acmCategoryList}
</div>
...
<input id="acmCategoryInputElement" ...>
...
<script>
  replaceInputWithFancytree(
    $('#acmCategoryList'),
    $('#acmCategoryInputElement')
  );
</script>
```

Implementaci funkce `replaceInputWithFancytree` nalezneme v souboru `www/js/main.js`.

Komponenta pro zobrazení seznamu kategorií původně využívala, stejně jako původní řešení, plugin jqxTree[11]. Ukázalo se však, že tento plugin nepodporuje nativně tzv. lazy rendering, tedy při vytvoření seznamu vytvoří zároveň všechny potřebné HTML elementy. Jelikož v systému evidujeme kolem 1400 ACM kategorií, samotné načtení seznamu trvalo v prohlížeči Google Chrome kolem deseti sekund, což bylo nepřijatelné. Proto bylo nakonec potřeba využít jiného pluginu pro zobrazení stromového seznamu. Vybrali jsme plugin Fancytree[12], který zvládá zobrazení větších stromových seznamů bez problémů.

4.1.7 Vytvoření speciálního formulářového prvku pro datum

V aplikaci je potřeba zadávat datumy v různých formulářích. V původní aplikaci byla pro zadávání datumů využita klasická textová pole s připojeným validátorem pro ověření formátu data. V souboru `www/js/main.js` pak byla zajištěna inicializace pluginu pro zobrazení kalendáře k formulářovým polím. Jednotlivá formulářová pole zde však byla vyjmenována přes ID HTML prvku. Jelikož tato ID generuje framework automaticky, stačí formulář přesunout či použít na jiném místě a zobrazení kalendáře u patřičných polí nebude fungovat.

Proto jsme zavedli speciální typ pole pro určení data, `App\Forms\Controls\DateInput`. Tato třída zajistí automatické použití validátoru a nastaví HTML elementu třídu, podle které se později inicializuje plugin pro zobrazení kalendáře u daného prvku. Dále jsme do třídy `App\Form\BaseForm` zavedli metodu `addDate`, která zajistí přidání prvku pro zadávání datumů do formuláře.

Zároveň jsme v systému zavedli i podobné třídy pro další varianty zadávání časových údajů. Jedná se o třídy `DateTimeInput`, `MonthInput` a `YearInput`. Zároveň jsme obohatili i třídu `BaseForm` o příslušné metody pro přidávání formulářových polí do formuláře.

4.1.8 Přidání nových filtrů pro šablony

Ve všech šablonách byly datumy formátovány explicitním uvedením formátu. Pokud bychom chtěli formát změnit, museli bychom ho změnit ve všech šablonách. Proto jsme zavedli do šablon filtr `ldate`. Místo vykreslování datumů přes konstrukci `{$someDate|date:'%Y-%m-%d'}` můžeme pak použít jednoduše konstrukci `{$someDate|ldate}`.

Zároveň jsme zavedli filtr pro zobrazení jmen autorů. Lze je pak jednoduše vykreslit konstrukcí `{$authorObject|authorName}`.

4.2 Implementace nových funkcí

4.2.1 Zavedení komponenty pro sadu přepínacích tlačítek

Jelikož jsme v aplikaci potřebovali využít přepínací tlačítka, a to na více místech, bylo žádoucí vytvoření komponenty, která by práci s tlačítky zjednodušila. Proto jsme vytvořili komponentu `ButtonGroupComponent`. Komponenta sama validuje vstup, lze jí zadat výchozí tlačítko. Samozřejmostí je podpora AJAXu a možnost přidat callback volaný při změně volby. U tlačítek můžeme definovat text, interní název (id tlačítka), ikonu a styl. Navíc lze s touto komponentou vytvářet i rozbalovací tlačítka, tedy že při kliknutí na tlačítko se zobrazí další možnosti.

4.2.2 Oddělení informací ohledně publikací a konferencí

Každý uživatel může mít přístup buďto k informacím o konferencích, k informacím o publikacích, nebo k informacím o publikacích a konferencích zároveň. Přístup k informacím o publikacích uživatel má, pokud má nastavenou roli `reader`, `submitter` nebo `admin`. Tyto role jsou již dostupné v rozšiřované aplikaci. Přístup k informacím o konferencích uživatel má, pokud má nastavenou roli `conference-user`, `conference-moderator` nebo `admin`. Pokud patří do skupiny `admin`, má tedy přístup k informacím o publikacích i konferencích zároveň. Přidělení oprávnění k rolím je definováno ve třídě `\App\Model\ACL`.

V aplikaci jsou do šablon presenterů i CRUD komponent předávány proměnné `$isPU` a `$isCU`, které jsou zkratkami termínů isPublicationUser, resp. isConferenceUser. Proměnné určují, ke kterým informacím má přihlášený uživatel přístup. V šablonách lze tedy omezit vykreslování informací podle oprávnění uživatele.

Zároveň má každý presenter i CRUD komponenta k dispozici podobné metody `isPU` a `isCU`, které vrací příslušnou hodnotu typu boolean. Lze tak omezit formulářová pole rovnou při vytváření formulářů.

4.2.3 Automatické archivování ročníků konferencí

Jelikož pro zvýšení přehlednosti vedeme příznak archivace u ročníků konferencí, je potřeba ročníkům nastavovat tento příznak automaticky, pokud již nejsou aktuální. Proto jsme vytvořili skript `bin\cron-archive-conference-years.php`. Na serveru je třeba nastavit automatické spouštění tohoto skriptu, např. s využitím služby `crond` pokud používáme unix-based systém. Skript nastaví příznak archivace u takových ročníků konferencí, kde je jejich rok menší než aktuální. Nejlepší je nastavit automatické spouštění skriptu s frekvencí jednou za den. Stačilo by sice nastavit spouštění skriptu jednou za rok, ale v době spuštění by server mohl být offline, proto je lepší nechat skript spouštět častěji.

4.2.4 Příprava systému pro více autentizačních mechanizmů

Jelikož jedním z požadavků bylo zavedení autentizace uživatele fakultním jménem a heslem, bylo třeba systém upravit, aby podporoval více autentizačních mechanizmů. Proto jsme zavedli třídu `BaseAuthenticator`, která implementuje rozhraní `IAuthenticator`. Třída definuje dostupné autentizační mechanizmy a umožňuje nastavit a zjistit aktuální autentikační mechanizmus uživatele. Dále třída deleguje přihlašování jménem a heslem na konkrétní autentikátor.

4.2.5 Přidání autentizačního mechanizmu LDAP

Jedním z požadavkem na aplikaci bylo zavedení přihlašování fakultním jménem a heslem. K tomu byl využit protokol LDAP. Přihlašování přes LDAP jsme implementovali s využitím doplňku pro Nette od Martina Míky[13]. Konfigurace se provádí v souboru `app\config\config.local.neon`.

4.2.6 Přidání autentizačního mechanizmu Shibboleth

Jelikož po implementaci LDAP přihlašování přišla nepříznivá zpráva, tedy že provoz fakultního LDAP serveru bude ukončen v květnu roku 2015, bylo potřeba nahradit tento mechanizmus jiným. Zvolena byla shibboleth autentikace. Proto jsme zavedli třídu `ShibbolethAuthenticator`, která autentizaci zaštiťuje. Při použití Shibboleth autentizace je nezbytná přítomnost doplňku pro apache `mod_shib`. Zároveň je třeba tento doplněk nakonfigurovat a nastavit zabezpečení shibboletem pro lokaci `WWW_ROOT/sign/shibboleth`.

4.2.7 Zavedení informací o indexaci sborníků konferencí ve veřejných databázích publikací

Tato funkce byla implementována zavedením CRUD komponenty `DocumentIndexCrud` a presenteru `DocumentIndexPresenter`. Zároveň byl upraven formulář pro přidání a úpravu ročníku konference, kde se dá indexace sborníků v daných databázích určit.

4.2.8 Možnost přiřazení workshopů ročníkům konferencí

Jedním z požadavků na systém bylo přidání možnosti libovolnému ročníku konference přiřadit ročník workshopu. Workshop jako takový není v systému reprezentován samostatnou entitou, nýbrž jako ročník konference. Možnost správy přidružených workshopů jsme realizovali úpravou CRUD komponenty `ConferenceYearCrud`. Zároveň jsme přidali zobrazování informací o přidružených workshopech do třídy `ConferencePresenter` a příslušných šablon.

Kapitola 5

Testování

Testování je nepostradatelným prvkem procesu vývoje softwaru. Aplikace byla testována pouze velmi základním způsobem, po implementaci nové funkce či refaktoringu části kódu byla funkčnost otestována přímo vývojářem. Nebyly však stanoveny všechny okrajové podmínky. Vymezení všech těchto podmínek by vyžadovalo hlubší analýzu. Systém je však připraven na testování jednotlivých komponent, lze využít falešných, tzv. mock objektů pro dosazení potřebných závislostí. V aplikaci nebyly využity unit testy ani integrační testy. Uživatelské testy probíhaly v průběhu vývoje, kdy zadavatel testoval funkcionalitu, navrhoval změny a upozorňoval na nedostatky.

Po dokončení vývoje byla aplikace předána vedoucímu práce k akceptačnímu testování. Později budou provedeny další uživatelské testy. Jelikož hlavní cílovou skupinou jsou studenti vyšších ročníků a pracovníci fakulty, bude třeba vybrat testery z těchto skupin. Zároveň je zde možnost využití laboratoře určené pro testování, která byla na fakultě nedávno otevřena. Jako podklad pro tvorbu testovacích scénářů poslouží mimo jiné případy užití uvedené v sekci 3.1.

Kapitola 6

Instalační příručka

Aplikace potřebuje ke svému běhu webový server s podporou PHP. Dále vyžaduje možnost připojení k MySQL databázi. Požadavky k jednotlivým službám uvedeme v následujících sekcích.

6.1 Konfigurace webového serveru

Uvedeme zde konfiguraci jen pro webový server Apache, jelikož se jedná o široce rozšířené řešení. Aplikace může být provozována i za použití jiných webových serverů.

Nejprve je potřeba zkopírovat zdrojové soubory na lokální disk. Zdrojové soubory jsou umístěny ve složce `src/impl` na přiloženém CD.

V konfiguračním souboru serveru Apache je třeba nastavit cestu k adresáři 'www', který se nachází v adresáři se zdrojovými soubory. To je možno zajistit klauzulí `DocumentRoot`. Pokud využíváme virtual hosting, je třeba uvést klauzuli v příslušném bloku `VirtualHost`. Další možností je využití direktivy `Alias`, k tomu je nutná přítomnost apache modulu `mod_alias`.

Aplikace vyžaduje přítomnost apache modulu `mod_rewrite` a možnost použít jeho direktivy v souborech `.htaccess`. Povolení či zakázání použití direktiv v souborech `.htaccess` lze určit direktivou `AllowOverride`.

Pokud má být dostupné přihlašování přes Shibboleth, je vyžadována přítomnost služby `shibd`, která musí být správně nakonfigurovaná. Dále je nutná dostupnost apache modulu `mod_shib`. Nakonec je potřeba zabezpečit URL `"%WWW_ROOT%/sign/shibboleth"`, na které je zajištěno přihlašování přes Shibboleth. Zabezpečení provedeme uvedením direktivy `AuthType shibboleth` a `ShibRequestSetting requireSession 1` (příklad viz Listing 6.1).

Listing 6.1: Zabezpečení URL pro přihlašování přes Shibboleth.

```
<Location /PubConf/sign/shibboleth>
        Allow from all
        AuthType shibboleth
        ShibRequestSetting requireSession 1
        require valid-user
        SSLOptions -StdEnvVars
</Location>
```

6.2 Nastavení PHP

Aplikace vyžaduje PHP verze 5.3.1 nebo vyšší. Pro správný běh aplikace musí být v PHP dostupná tato rozšíření:

- `PDO`
- `PDO_MYSQL`
- `Tokenizer`
- `iconv`
- `Reflection`
- `SPL`
- `PCRE`
- `SQLite3`
- `Fileinfo`
- `LDAP` (pokud má být dostupné přihlašování přes LDAP).

Zároveň je potřeba v konfiguračním souboru (obvykle `php.ini`) zkontrolovat direktivy `post_max_size` a `upload_max_filesize`. Tyto hodnoty je potřeba nastavit na maximální možnou velikost nahrávaného souboru. Současně je třeba zkontrolovat, zda jsou direktivy `register_globals` a `magic_quotes_gpc` nastaveny na `Off`.

6.3 Zřízení přístupu do MySQL

Aplikace vyžaduje možnost připojení k MySQL databázi. Nejvhodnějším řešením je umístění MySQL databáze na serveru, na kterém je aplikace nainstalována. Nejprve vytvoříme nový uživatelský přístup, přes který bude aplikace k databázi přistupovat. K novému účtu vytvoříme i databázi. Zvolíme uživatelské jméno (dále `%DB_USER%`), heslo (dále `%DB_PASS%`) a název databáze (dále `%DB_NAME%`).

Vytvoření uživatelského účtu zajistíme následujícím SQL dotazem:

Listing 6.2: Vytvoření přístupu do MySQL.

```
CREATE USER '%DB_USER%'@'localhost' IDENTIFIED BY '%DB_PASSWORD%';
```

Následující SQL dotaz pak zajistí vytvoření databáze:

Listing 6.3: Založení databáze v MySQL.

```
CREATE DATABASE `%DB_NAME%` DEFAULT CHARACTER SET = 'utf8'
                            DEFAULT COLLATE 'utf8_general_ci';
```

Dále je potřeba přiřadit oprávnění k nově vytvořenému uživatelskému účtu. Učiníme tak spuštěním následujícího SQL dotazu:

Listing 6.4: Nastavení oprávnění pro uživatele.

```
GRANT INSERT,SELECT,UPDATE,DELETE ON `%DB_NAME%`.* TO '%DB_USER%'@'localhost';
```

Jako poslední krok je třeba provést import struktury databáze a následně i dat. `%CD_PATH%` je cesta k souborům na přiloženém CD a `%ADMIN_USERNAME%` je uživatelské jméno použité k administraci MySQL, např. 'root'. Spustíme následující příkazy v terminálu:

Listing 6.5: Import dat.

```
mysql %DB_NAME% -p -u %ADMIN_USERNAME% < %CD_PATH%/src/database/create-schema.sql
mysql %DB_NAME% -p -u %ADMIN_USERNAME% < %CD_PATH%/data/database/import-data.sql
```

6.4 Konfigurace aplikace

Aplikace musí být nakonfigurována aby mohla správně fungovat. Nastavme aktuální cestu na adresář, kde se nacházejí zdrojové kódy aplikace. Konfigurační soubory pak najdeme v adresáři `app/config`. Zde upravíme soubor `config.local.neon`, kde nastavíme údaje pro připojení k databázi úpravou sekce `nette.database`. Dále provedeme úpravy v sekci `parameters`, kde lze změnit další nastavení, např. nastavení emailu, shibboleth autentizace a kontakt na administrátora.

Dále je potřeba povolit zápis do určitých cest pro uživatelský účet využívaný webserverem. To lze učinit např. příkazy `chown` a `chmod`, pokud používáme systém založený na UNIXu. Zápis je potřeba povolit pro následující cesty vzhledem k adresáři se zdrojovými soubory:

- `log`
- `temp`
- `temp/*`
- `storage`
- `vendor/jkuchar/multiplefileupload/MultipleFileUpload/Model/SQLite3`
- `vendor/jkuchar/multiplefileupload/MultipleFileUpload/Model/SQLite3/database.sqlite3`

Na závěr je třeba nastavit automatickou archivaci ročníků konferencí. Patřičný skript najdeme v cestě `bin/cron-archive-conference-years.php`. Nejlépe nastavíme spouštění tohoto skriptu jedenkrát za den. Pokud používáme UNIX-based systém, můžeme se přihlásit za uživatele používaného webserverem příkazem `su`. Poté spustíme příkaz `crontab -e` a vložíme následující řádek, kde je nutno zadat absolutní cestu ke skriptu.

Listing 6.6: Nastavení automatické archivace ročníků konferencí.

```
0 6 * * * /usr/bin/php ABSOLUTNI_CESTA_KE_SKRIPTU
```

Závěr

Cílem této práce bylo rozšíření již existující webové databáze konferencí. Provedli jsme analýzu, kde jsme zkoumali další možnosti řešení. Mezi tyto možnosti patřilo rozšíření paralelně vyvíjené webové databáze publikací a vývoj kompletně nové aplikace pro správu konferencí.

Nevhodnějším řešením problému se ukázalo být rozšíření webové databáze publikací, vyvíjené v rámci diplomové práce[6] na FIT, ČVUT.

Rozšiřovaná aplikace prošla významným refaktoringem, kde jsme zredukovali duplicitní kód a zlepšili rozšiřitelnost aplikace. Poté byly implementovány nové funkce, mezi které patří např. přihlašování přes Shibboleth, rozšíření a založení nových entit. Zároveň jsme rozšířili ER model a přidali nové tabulky a sloupce. Na závěr byla přenesna i data z původní databáze konferencí.

Testování probíhalo v průběhu vývoje a předpokládá se i další testování s využitím laboratoře pro testování na FIT, ČVUT.

Cíl práce byl splněn. Aplikace je nyní dostatečně rozšiřitelná, nebude tedy problém v budoucnu implementovat nové funkce. Aplikace je dostupná v produkční verzi na http://ddd.fit.cvut.cz/PubConf/. Využívat ji může každý, kdo má nějaký vztah k fakultě, jelikož aplikace podporuje přihlášení fakultním autentizačním mechanizmem. Věřím, že aplikace bude plnit svůj účel a bude sloužit ve prospěch svých uživatelů.

Literatura

[1] Zdrubecký, V.: *Nová verze databáze konferencí*. Bakalářská práce. Praha: České vysoké učení technickıe v Praze, Fakulta informačních technologií, 2013.

[2] Galík, P.: *Rozšíření databáze konferencí*. Bakalářská práce. Praha: České vysoké učení technickıe v Praze, Fakulta elektrotechnická, 2008.

[3] Kemr, J.: *Databáze konferencí*. Bakalářská práce. Praha: České vysoké učení technickıe v Praze, Fakulta elektrotechnická, 2006.

[4] Journal of Universal Computer Science: *ACM Categories*. [cit. 2015-05-04]. Dostupné z: `http://www.jucs.org/jucs_info/acm_categories`

[5] *X-CD Conference Management Software | A fully integrated and centralized conference management solution*. [cit. 2015-05-09]. Dostupné z: `http://www.x-cd.com/`

[6] Kubálek, J.: *Webová databáze referencí*. Diplomová práce. Praha: České vysoké učení technickıe v Praze, Fakulta informačních technologií, 2015.

[7] Potel, M.: MVP: Model-view-presenter: The Taligent programming model for C++ and Java. 1996, [cit. 2015-05-04]. Dostupné z: `http://www.wildcrest.com/Potel/Portfolio/mvp.pdf`

[8] Fowler, M.: *Inversion of control containers and the dependency injection pattern*. 2004, [cit. 2015-05-04]. Dostupné z: `https://blog.itu.dk/MMAD-F2013/files/2013/02/3-inversion-of-control-containers-and-the-dependency-injection-pattern.pdf`

[9] Fowler, M.: *Refaktoring: Zlepšení existujícího kódu*. Praha: Grada, 2003.

[10] Hudák, D.: *NasExt/SortingControl · GitHub*. 2015, [cit. 2015-05-05]. Dostupné z: `http://github.com/nasext/sortingcontrol`

[11] jQWidgets: *jQuery Tree*. [cit. 2015-05-05]. Dostupné z: http://www.jqwidgets.com/jquery-widgets-demo/demos/jqxtree/index.htm

[12] Wendt, M.: *mar10/fancytree · GitHub*. 2015, [cit. 2015-05-05]. Dostupné z: https://github.com/mar10/fancytree

[13] Míka, M.: *foglcz/ldap-authenticator · GitHub*. 2015, [cit. 2015-05-06]. Dostupné z: https://github.com/foglcz/ldap-authenticator/

Příloha A

Seznam použitých zkratek

SW Software

ER model Entity-relationship model

LDAP Lightweight Directory Access Protocol

DB Databáze

RDBMS Relational database management system

UI User interface

OOP Object-oriented programming

MVP Model-View-Presenter (architecture)

AJAX Asynchronous Javascript and XML

CRUD Create, Read, Update, Delete (operations)

SSO Single sign on

URL Uniform resource locator

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `data`
  - `database` ..... databázové skripty pro import dat
  - `storage` ..... nahrané dokumenty publikací
- `src`
  - `database` ..... databázové skripty
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS