TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Automatizace procesu zadávání studentských prací a projektů

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Daniel Maděra**
Vedoucí práce: Ing. Jana Vitvarová, Ph.D.

Liberec 2014

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Automation of the thesis and students projects assignment process

Bachelor thesis

Study programme:	B2646 – Information technology
Study branch:	1802R007 – Information technology
Author:	**Daniel Maděra**
Supervisor:	Ing. Jana Vitvarová, Ph.D.

Liberec 2014

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Daniel Maděra**
Osobní číslo:	**M11000103**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Automatizace procesu zadávání studentských prací a projektů**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Na základě rozhovorů s vedoucím práce, dále s tajemníkem a vedoucím ústavu MTI identifikujte stávající proces zadávaní studentských projektů a závěrečných prací.
2. Na základě dalších rozhovorů navrhněte a nechte si odsouhlasit nový proces s prvky automatizace, který stávající proces zjednoduší a zrychlí.
3. Vytvořte webovou aplikaci, která bude nový proces podporovat.
4. Požadavky na bezpečnost aplikace: na autentizaci uživatelů využijte přihlašovací systém Shibboleth.
5. Požadavky na rozšiřitelnost aplikace: aplikace musí být navržena tak, aby byla snadno rozšířitelná o další ústavy na fakultě.
6. Vytvořte uživatelskou příručku ve formě nápovědy.
7. Aplikaci otestujte.
8. Vytvořte dokumentaci k aplikaci a vhodně okomentujte zdrojový kód.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **30–40 stran**

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

Seznam odborné literatury:

[1] **Booch G., Rumbaugh J., Jacobson I.,: The Unified Modeling Language User Guide, Addison-Wesley, ISBN 0-321-26797-4**

[2] **Kendall S.: The Unified Process Explained. ISBN 0-201-74204-7**

[3] **Connolly T., Begg C., Strachan A.: Database systems, Addison-Wesley, 1999**

[4] **Přihlašovací system Shibboleth: http://shibboleth.net/**

Vedoucí bakalářské práce: **Ing. Jana Vitvarová, Ph.D.**
Ústav mechatroniky a technické informatiky

Datum zadání bakalářské práce: **10. října 2013**

Termín odevzdání bakalářské práce: **16. května 2014**

prof. Ing. Václav Kopecký, CSc.
děkan

Technická univerzita v Liberci - 14 - L.S.

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2013

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 16. 05. 2014

Podpis: 

Abstrakt

Tato práce dokumentuje automatizaci procesu zadávání a výběru studentských prací v podobě webové aplikace. Výstupem práce je funkční aplikace, která bude reálně používána studenty a vyučujícími na Fakultě mechatronicky, informatiky a mezioborových studií (FM) Technické univerzity v Liberci (TUL). Aplikace má za úkol zrychlit proces a usnadnit komunikaci při výběru bakalářských či diplomových prací a projektů. Výstupem aplikace je poskytnout administrátorům export oficiálních zadání, které budou následně zadávány do Informačního systému (IS) STAG. Pro studenty je výstupem vygenerované oficiální zadání. Dle tohoto zadání mohou začít pracovat na dané práci.

Vyučujícím aplikace umožňuje přehlednou správu témat prací, které jsou následně zobrazeny studentům k výběru. Studentům aplikace poskytuje přehledný soupis zadání s rozsáhlými možnostmi filtrování, kde si mohou vybrat práce dle požadovaných zaměření či oblíbených vyučujících. Studenti mají možnost výběru více témat najednou.

V rámci vývoje aplikace bylo nutné správně analyzovat stávající proces zadávání prací od zadavatelů. Tím byli vedoucí a tajemník ústavu Ústavu mechatroniky a technické informatiky (MTI). Následně navrhnout a nechat odsouhlasit nový proces, který bude zahrnovat automatizaci v podobě webové aplikace. Ta měla splňovat požadavky intuitivního grafického prostředí s dostatečně rozsáhlou nápovědou. Návrh aplikace by měl být proveden tak, aby byl snadno rozšiřitelný o další ústavy na fakultě, připravený pro různé jazykové verze a vhodně využíval autorizační části s pomocí autentizačního systému Shibboleth.

Aplikace je realizována v programovacím jazyku PHP s využitím MySQL databáze. Práce popisuje vývoj aplikace za použití frameworku CakePHP a vodopádové metodiky vývoje softwaru.

Klíčová slova

Webová aplikace, programování, návrh aplikace, PHP, MySQL, CakePHP framework

Abstract

This thesis describes an automation of the thesis and student's projects assignment process in form of a web application. The output of this thesis is the application that will be used by students and teachers at the Faculty of Mechatronics, Informatics and Interdisciplinary Studies (FM) Technical University of Liberec (TUL). A main goal of the application is to accelerate the current process and facilitate a communication in the selection of bachelor and master degree theses or student's projects. Administrators use the application to export the official theses. These theses will be entered to the Information system STAG. Students can export the official theses and start working on them immediately.

Teachers can easily manage topics of each thesis. These topics are listed to students and afterwards selected by them. The application provides for students a structured list of topics with extensive options of filtering such as required specializations or favourite teachers. Students can select more than one topic.

At the beginning of the development was necessary to properly analyze the current assignment process at the Institute of Mechatronics and Computer Engineering (MTI). After that a new process was designed with supported automatization in form of the web application. Requirements for user interface are simplicity and intuitiveness with a sufficiently extensive help. The design of application should be easily extended for more institutes on faculty and prepared for internationalization. Authorization service Shibboleth should be part of an authentication system.

Application was written in PHP programming language with usage of MySQL database. The thesis describes software development using CakePHP framework and waterfall software development methodology.

Keywords

Web application, programming, application design, PHP, MySQL, CakePHP framework

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

Seznam zkratek

ACL Access control list.

AJAX Asynchronous JavaScript and XML.

API Application programming interface.

BP Bakalářská práce.

CSS Cascading Style Sheets.

CSV Comma-separated values.

DP Diplomová práce.

FM Fakulta mechatroniky, informatiky a mezioborových studií.

GPL General Public License.

HTML HyperText Markup Language.

HTTP Hypertext Transfer Protocol.

IS Informační systém.

IT Informační technologie.

MTI Ústav mechatroniky a technické informatiky.

MVC Model-View-Controller.

ORM Objektově relační mapování.

PDF Portable Data Format.

URL Uniform Resource Locator.

XML Extensible Markup Language.

1 Úvod

Tato práce vznikla z důvodu zbytečně komplikovaného a časově náročného aktuálního procesu zpracování témat studentských prací a projektů. Cílem práce je zjednodušit a zrychlit zpracování témat. Kompletní analýzou tohoto procesu se zabývá kapitola 2. Pro přehlednost byl proces převeden do vývojových diagramů, které přehledně znázorňují zbytečné úkony, které proces provází.

Automatizace procesu popisuje kapitola 3.1. Pro porovnání s neautomatizovaným procesem, slouží znovu vývojové diagramy, které znázorňují, kde došlo k nahrazení uživatelských úkonů aplikací. Vývojové diagramy naznačují další užitečné funkce, které aplikace přináší pro studenty i pedagogy.

Pro programování aplikace byly zadavatelem dané dostupné technologie. Bylo nutné řešit problematiky výběru frameworků, které popisuje kapitola 3.2. Aplikace byla realizována s frameworkem CakePHP. Tato kapitola dále popisuje všechny další použité externí knihovny.

Kapitola 4 popisuje způsob realizace aplikace v podobě rozdělení na menší problematiky. Realizace aplikace probíhala na základě vodopádové metodiky vývoje softwaru. Jaké výhody a úskalí přinesla tato metodika do vývoje popisuje kapitola 4.1. Dalšími podkapitoly bylo řešení požadavků na rozšiřitelnost aplikace, mimo jiné příprava pro překlad uživatelského rozhraní. Návrh grafického rozhraní je založen na UseCase diagramu z kapitoly 4.2 popisující uživatelské rozhraní.

Bezpečnostní požadavky na aplikaci shrnuje kapitola 4.10. Tato kapitola popisuje návrh systémových rolí aplikace a použití autentizačního systému Shibboleth. Dále popisuje způsob autorizace pro jednotlivé sekce aplikace a jejich rozdělení. Kapitola se dále zabývá problematikou a řešení autorizačních komponent ve frameworku CakePHP a také realizaci alternativního přidělení oprávnění.

Závěr kapitoly o realizaci aplikace tvoří mimo jiné problematika a řešení vhodné nápovědy. Aplikace by měla splňovat požadavek připravenosti pro reálný provoz na FM, tzn. vytvořit funkční a otestovaný kód. Kompletním testováním aplikace se zabývá kapitola 4.13. Výstup aplikace je export oficiálních zadání. Následující kapitoly se zabývají způsoby exportu dat a jeho začlenění jako komponenty do frameworku CakePHP.

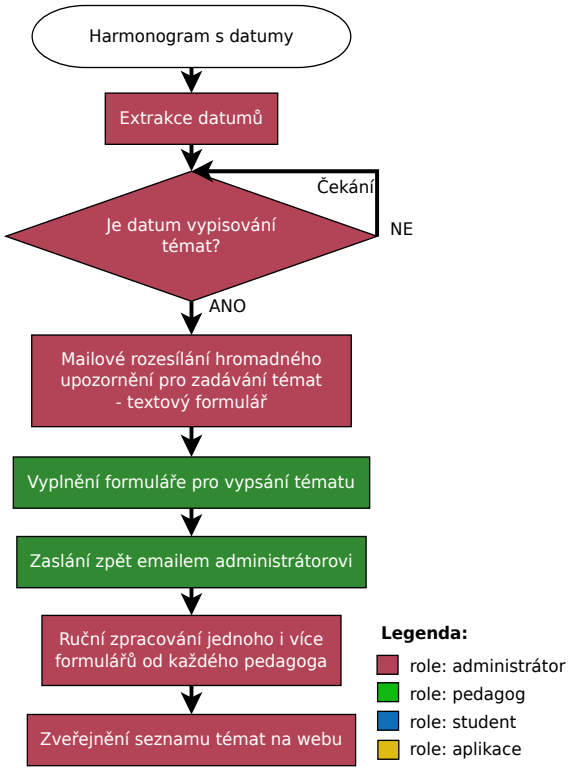
2 Analýza stávajícího stavu

K provedení automatizace zadávání prací a projektů, bylo nutné nejdříve proniknout do aktuálního procesu probíhajícího na MTI. Informace o aktuálním procesu mi poskytli tajemník, vedoucí ústavu MTI a vedoucí bakalářské práce. Tento neautomatizovaný proces byl rozdělen na 3 části:

- Zadávání témat
- Výběr témat
- Dokončení zadávání a export oficiálních zadání

Pro vizualizaci procesu byl použit vývojový diagram. Barevně jsou odlišeni aktéři, kteří danou akci provádějí. Červená označuje správce (administrátora), který má za úkol extrahovat od vyučujících jednotlivá témata. Procesy označené zelenou barvou jsou ty, které provádí pedagogové a modrá označuje akce studentů, jak již naznačuje legenda.

2.1 Zadávání témat

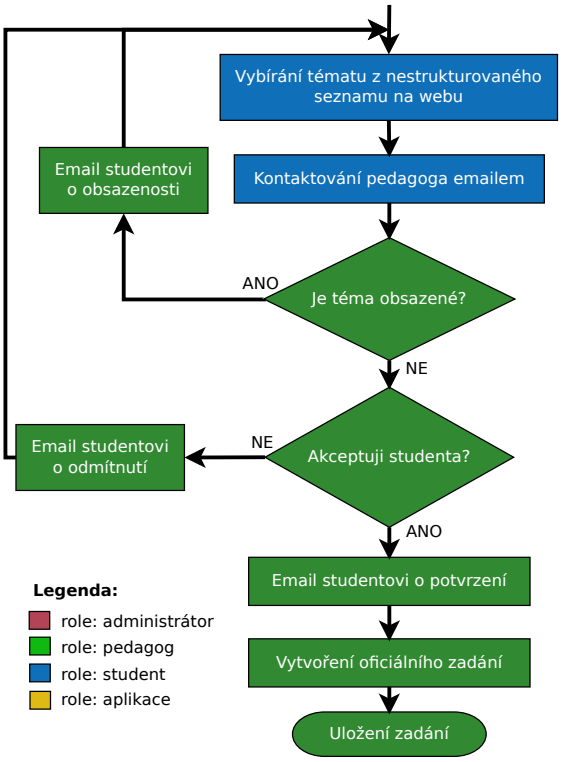


Obrázek 1: Aktuální stav zadávání témat

Z obrázku 1 je patrné, že správci je předán harmonogram s datumy pro následující rok,

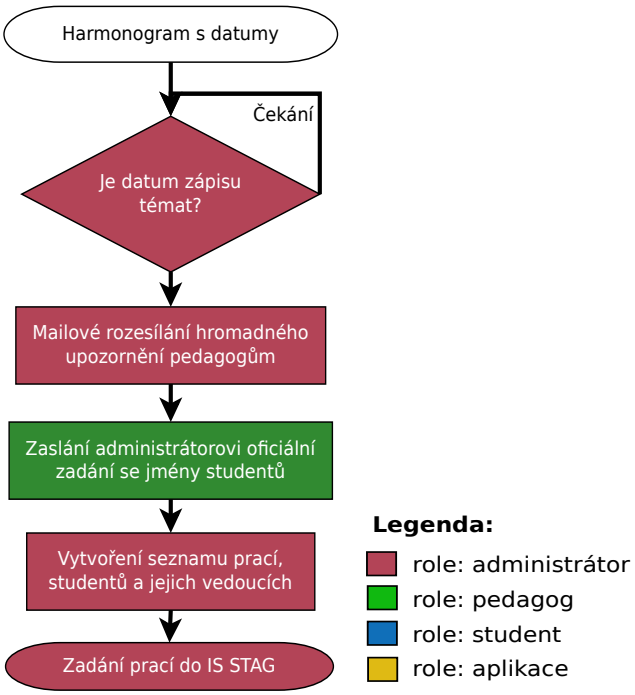
který obsahuje například konečný datum vypisování témat či odevzdání prací. Administrátor musí zvolit čas, který poskytne pedagogům na vytvoření témat. Správce tedy rozešle hromadně všem pedagogům na MTI email s formulářem pro vypsání tématu a termínem odevzdání témat bakalářských a diplomových prací. Pedagog si vytvoří daný počet kopií formuláře, podle toho kolik zadání požaduje vypsat. Formuláře vyplní a zašle zpět správci, který zpracuje jednotlivá témata a metodou kopírování textů zveřejní na webu MTI.

2.2 Výběr témat



Obrázek 2: Aktuální stav vybírání témat

Obrázek 2 znázorňuje proces vybírání tématu. Studentovi je k dispozici na webu MTI seznam témat. Tento list není strukturovaný ani v žádné podobě aktualizovaný. Nelze tedy z něho identifikovat stav tématu. Zda o toto téma není zažádáno, nebo již není obsazené. Po vybrání tématu student kontaktuje emailem pedagoga, který odpovídá se stavem tématu. Pokud je téma volné a student splňuje požadavky, potvrzuje vyučující emailem studentovi přiřazení. Z tématu si pedagog vytvoří oficiální zadání pomocí formuláře, který poskytl správce v úvodním hromadném emailovém upozornění.



Obrázek 3: Aktuální stav dokončení zadávání

2.3 Dokončení zadávání a export oficiálních zadání

Poslední část (viz obrázek 3) vývojového diagramu popisuje předání zadání do IS STAG. Správce je nucen znovu rozeslat hromadné emailové upozornění pedagogům, aby mu zaslali formuláře s vyplněnými daty o oficiálních zadání se jmény studentů. Z těchto formulářů jsou zadání metodou kopírování vkládány do IS STAG.

2.4 Shrnutí

Aktuální proces zadávání není příliš vhodný. Dochází k velkému množství zasílání zbytečných emailových upozornění. Pokud téma obsadí student, není tato informace v seznamu prací obsažena. To má za následek, že další student, který má zájem o stejné zadání, bude emailem o dané téma žádat. Pedagog musí na žádost reagovat, že téma je obsazené. Další obrovskou nevýhodou tohoto řešení je, že všechny úpravy témat mohou být provedeny pouze správcem.

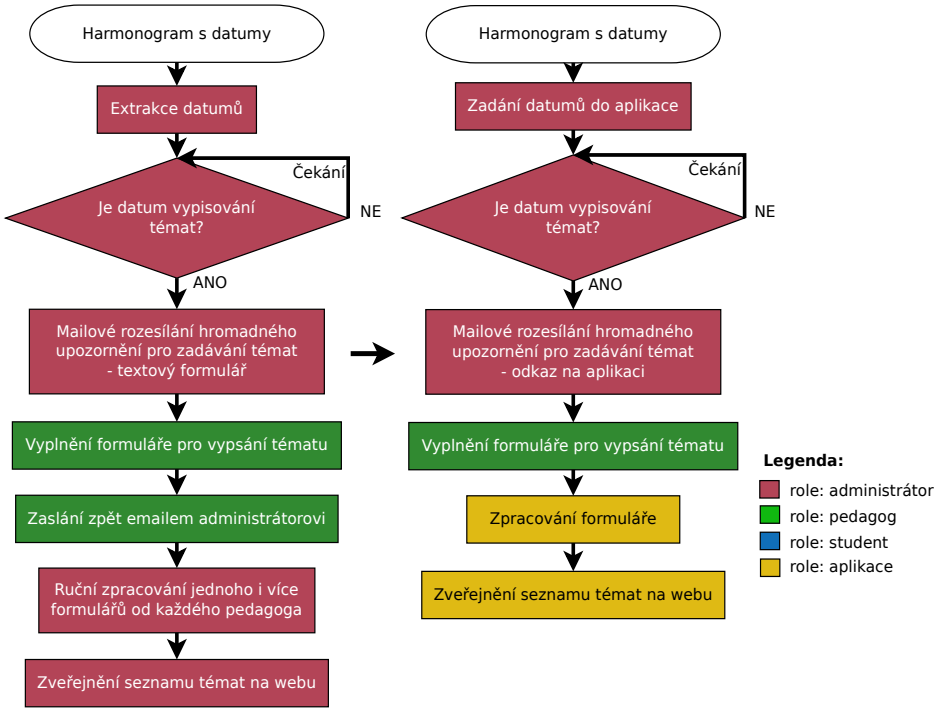
Z pohledu studenta je nepříjemné, že nemá zadání ze všech ústavů na fakultě centralizované, nýbrž musí procházet webové stránky každého ústavu a hledat seznamy témat. Provedl jsem krátký průzkum mezi studenty na téma - jak si vybírali zadání bakalářské práce a na jaké problémy při výběru narazili. Největší potíž měli studenti s tím, z jakého ústavu si práci mohou vybrat.

3 Návrh na řešení

3.1 Proces automatizace

V předchozích kapitolách byl představen stávající proces zadávání a výběru bakalářských, diplomových prací i projektů ve vývojovém diagramu. Následující obrázky znázorňují v jakých částech nahradila aplikace práci aktérů v procesu. V levé části diagramu je vyobrazen stávající proces a pravá část ukazuje částečně automatizovaný proces. Žlutou barvou jsou vyobrazeny procesy, které vykonává aplikace.

3.1.1 Zadávání témat



Obrázek 4: Automatizované zadávání témat

Z obrázku 4 je zřejmé, že v první části procesu zadávání došlo k minimální automatizaci. V první verzi návrhu aplikace bylo zakomponováno automatické zaslání upozornění vyučujícím na daný blížící se koncový datum. Administrátor by měl za úkol pouze zadat datumy do systému a aplikace by se postarala o upozornění. Tento návrh nebyl podpořen zadavatelem. Požadoval vlastní odesílání upozornění, kde čas i četnost odesílání emailů je dostatečně variabilní. Dalším problémem automatického upozorňování je databáze pedagogů. Administrátor by musel vyzvat nejdříve všechny pedagogy, aby se do aplikace přihlásili a tím zajistili uložení emailové informace do databáze - tato funkce je detailně.

V následujícím roce by byla již vytvořena databáze pro rozeslání upozornění, ale administrátor by musel do aplikace zadat pedagogy, kteří v minulém roce například ještě nebyli součástí zaměstnanců univerzity. Možností by bylo aplikaci provázat s IS STAG, kde tento systém poskytoval nějaké Application programming interface (API) přes které by bylo možné tento seznam pedagogů aktualizovat. Takové API IS STAG bohužel neposkytuje.

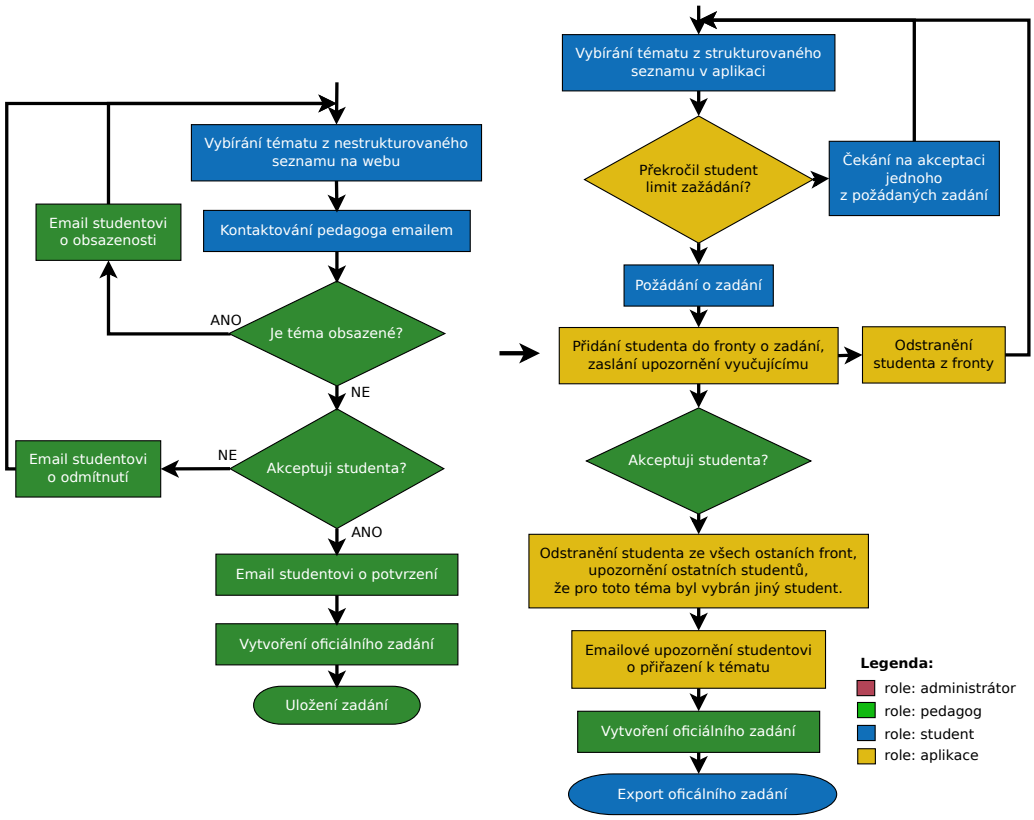
Rozdíl od stávajícího zadávání prací je v procesu mailového rozesílání upozornění. Nyní se v emailu rozesílá pouze odkaz na aplikace, kde pedagogové po přihlášení vyplní formulář pro vytvoření tématu. Aplikace provede validaci formuláře, data uloží do databáze a téma je zveřejněno v nyní již strukturovaném seznamu témat.

Doplňující funkcí pro vyučující je možnost zkopírování předchozího zadání a následně dané zadání editovat. Témata, která například svou rozsáhlostí mohou být zpracována ve více řešitelích, musí být zadána do systému tolikrát, kolik bude předpokládaných řešitelů problému. Z tohoto důvodu byla zavedena možnost zkopírování předchozího zadání. Pedagogové mohou označit zadání jako neaktivní. Toto zadání se nezobrazí studentu k výběru.

3.1.2 Výběr témat

V části výběru témat (viz obrázek 5) došlo k vícenásobným změnám v automatizaci. Do procesu byla zakomponována funkce burzy, kde více studentů může zažádat o jedno téma a pedagog si následně vybírá z přihlášených studentů. Po vybrání zadání student pokračuje požádáním o téma. Aplikace poté zkontroluje, zda student nepřekračuje limit zažádaných prací. Pokud ano, student musí vyčkat na potvrzení či odstranění z fronty vyučujícím u jiného zadání. V opačném případě proces výběru pokračuje. Zadavatel daného tématu (pedagog) je upozorněn na tuto skutečnost emailovým upozorněním. Pokud student splňuje všechny požadavky, pedagog ho akceptuje a student je přiřazen k danému tématu a tím odstraněn ze všech ostatních front stejného typu práce, o které v průběhu vybírání témat žádal. Ostatní studenti, kteří požadovali stejné téma jako vybraný student, jsou upozorněni, že téma obsadil někdo jiný. Pokud mají vyčerpaný počet vybraných témat, uvolní se jim jedna pozice pro další téma. Pokud pedagog studenta neakceptuje, dochází k odstranění studenta z fronty, na tuto událost je student upozorněn emailovou zprávou. V opačném případě proces pokračuje. Pedagog je přesměrován na vytvoření oficiálního zadání.

Jedním z požadavků zadavatele bylo rozdělit zadání na 2 části - téma práce a oficiální zadání. Téma je pouze obecný popis problému, ze kterého studenti vybírají. Vedoucí práce díky tomu nemusí téma detailně specifikovat už při vytváření. K úplné specifikaci dochází



Obrázek 5: Automatizované vybírání témat

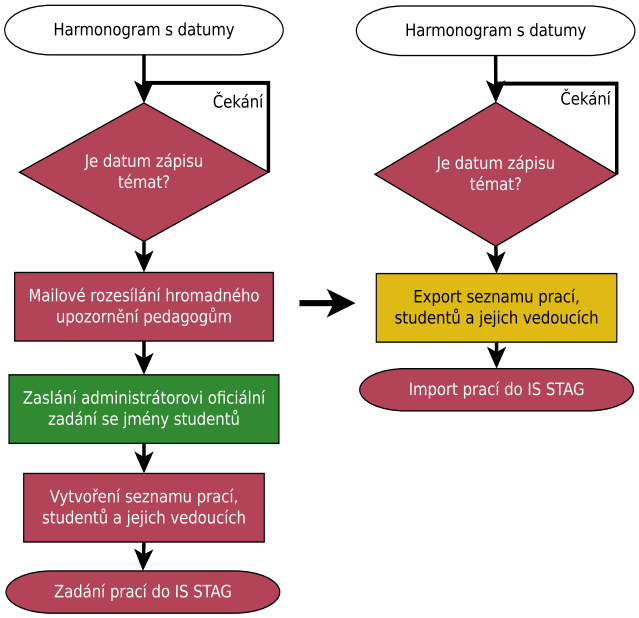
až v části oficiální zadání. Student je v této fázi k tématu přiřazen. Přiřazení studenta může probíhat u bakalářské práce už například rok dopředu, pedagog má proto možnost oficiální zadání pouze rozpracovat a zatím neposílat administrátoru k zpracování.

3.1.3 Dokončení zadávání a export oficiálních zadání

V poslední části diagramu (obrázek 6) došlo k automatizaci v komunikaci mezi administrátorem a pedagogy. Administrátor čeká na datum zápisu zadání do IS STAG. Z aplikace vyexportuje kompletní oficiální zadání se jmény studentů. Po tomto exportu aplikace uzavře možnost zadávání a editaci zadání. Proces pokračuje exportem do IS STAG. Z aplikace lze vyexportovat 3 typy souborů - CSV, XML nebo TXT.

3.1.4 Shrnutí

K automatizaci došlo pouze v jisté míře. Původní návrh zahrnoval automatizaci i rozesílání upozornění pedagogům a studentům, ale tento návrh neprošel u zadavatele, který požadoval vlastní správu hromadného upozornění. Dalším problémem by byl zdroj dat.



Obrázek 6: Automatizované dokončení zadávání

Jelikož není k dispozici žádné navázání komunikace s IS STAG například v podobě API, tudíž nelze získat databázi vyučujících či studentů. Administrátor by byl nucen tyto databáze spravovat bez jakékoliv automatizace, rychle by ztratil přehled, který pedagog nebo student ještě je člen univerzity. Data v systému by byla nekonzistentní.

Funkce burzy zavádí do aplikace jeden problém. Jelikož na změny stavu tématu jsou vyučující upozorněni emailem, vzniká prodleva mezi požádáním o téma a přiřazením studenta k práci. Pokud student zažádá o tři zadání v jeden čas, rozešlou se emaily všem vedoucím daných témat. Email obsahuje odkaz na aplikaci, kde by měl studenta k práci přiřadit. V případě že jiný vyučující potvrdí stejného studenta dříve, odkaz na aplikaci s potvrzením bude už neplatný. Na tuto možnost je upozorněn pedagog v emailu.

Dalším problémem byl import vyhotovených oficiálních zadání do IS STAG. Jak už bylo zmíněno, IS neposkytuje API, přes které by bylo možné komunikovat s aplikací. Export z aplikace se proto prování v textovém režimu, aby bylo přehledné a snadné přenést data metodou kopírování do aplikace. Pro následný import je vytvořen export do XML souboru, který obsahuje tyto data ve strukturované formě.

Výhodou pro studenty v automatizovaném řešení je možnost exportu oficiálního zadání ihned potom, co pedagog potvrdí zadání k dalšímu zpracování. Studenti tudíž nemusí čekat dokud nedostanou podepsané oficiální zdání.

3.2 Výběr technologií

K vývoji aplikace mi byl přiřazen adresář na serveru sirius2.tul.cz. Na tomto serveru běží HTTP server s PHP modulem a umožňuje tedy programovaní webových aplikací v PHP v5.4. Dále mi byl poskytnut autentizační systém Shibbolleth a MySQL databáze. Z frontendových frameworků byly použity Bootstrap v2.3 a jQuery v2.1. V následujících kapitolách jsou představeny jednotlivé frameworky a odůvodněn jejich výběr.

3.2.1 Výběr aplikačního frameworku

V začátku plánování realizace prvním problémem bylo vybrat správný framework pro programování aplikace v PHP. Dostupné technologie zadavatelem nikterak neomezili výběr, jelikož většina z dostupných PHP frameworků má podporu MySQL databáze a implementace vlastního způsobu autentizace. Výhoda oproti minulosti je, že na trhu je zdarma dostupných hned několik desítek frameworků. Po detailnějším zaměření jsem výběr zúžil na tři hlavní - CakePHP, Zend a Nette. Jejíchž srovnání je popsáno v následující kapitole. Při práci s frameworky se z velké části programování věnuje studium dokumentace. Pokud daný framework nemá dostatečně rozsáhlou dokumentaci a komunitu lidí, kteří s ním pracují, práce s frameworkem trvá mnohem déle. Programátor má při řešení daných úskalí mnohem menší pravděpodobnost nalezení řešení. Při výběru jsem upřednostňoval tři hlavní vlastnosti frameworku – rychlost zpracování požadavku s přístupem do databáze, flexibilita, čistota a stručnost kódu. Jedním z hlavních požadavků je možnost rozšíření aplikace pro všechny ústavy na FM, bylo nutné klást důraz na rychlost zpracování, aby nedocházelo k latenci serveru při vysokém zatížení. Některé frameworky obsahují nepřeberné množství implementovaných funkcí, ale na úkor rychlosti a hardwarových požadavků. Aplikace by měla být navržena tak, aby nebyl problém přizpůsobit se změnám zadavatele. Proto byl kladen důraz na vysokou flexibilitu. Čistota a stručnost kódu je nedílnou součástí všech typů projektů, jelikož orientace a porozumění kódu je prvním krokem k úspěšnému týmovému rozšiřování aplikace v budoucích letech.

3.2.2 CakePHP

CakePHP je open source aplikační webový framework vyvíjený společností Cake Development Corporation od roku 2007. Je uvolněný pod licencí MIT (X11) a z velké části založen na známém frameworku Ruby On Rails [1]. Výhodou tohoto frameworku je, že programátora nutí k dodržování Model-View-Controller (MVC) návrhového vzoru. CakePHP také podporuje Objektově relační mapování (ORM). Nedochází přímo k vytvoření datových

objektů s obsahem jednotlivých položek v databázi jako je to například u Java Hibernate, nýbrž k poskytnutí dat v asociativním poli, podrobněji popsáno v kapitole 4.3. Asociativní pole je seznam klíčů typu řetězec a k nim hodnoty libovolného datového typu. CakePHP dále obsahuje velice užitečnou konzoli s nástroji ke správě kódu a adresářové struktury. Poskytuje mimo jiné generování modelových tříd pouze na základě nastavení připojení k databázi. Z možností konzole jsem ještě využil Access control list (ACL) komponentu, která slouží ke správě rolí a jejich oprávnění. Použití komponenty je detailněji popsáno v kapitole 4.10.

3.2.3 Porovnání CakePHP s Nette a Zend

S Nette frameworkem [9] jsem již pracoval na předchozích projektech. Pro české programátory je výhodou dostatečně rozšířená česká komunita i dokumentace. Nette obsahuje například užitečný šablonový zápis, tím dodává čistotu, přehlednost a snadnou orientaci v kódu, kde se renderuje HTML. CakePHP disponuje standardní sadou funkcí jako Nette – směrování, validace, základní ověřování, ACL i například použití komponent a helperů. Z této dvojice jsem vybral CakePHP. Od použití Nette bylo ustoupeno z důvodu nekompatibility. Po aktualizaci na novější verzi Nette bylo nutné přeprogramovat již vytvořené šablony. Dalším důvodem byla touha po zkoumání nového frameworku [7]. U Zend 2 frameworku docházelo ke srovnání pouze podle dokumentace. Zend disponuje standardními funkcemi jako Nette nebo CakePHP, navíc obsahuje například různé nadstandardní validační funkce. Nevýhodou Zend 2 frameworku je nepříjemná rozsáhlost kódů. Tato problematika je vyjádřena v následujících tabulkách 1 a 2 [8].

Tabulka 1: Srovnání kódové rozsáhlosti - přihlašovací formulář

Framework	**Počet řádků**
`CakePHP`	12
`Nette`	8
`Zend 2`	27

Tabulka 2: Srovnání kódové rozsáhlosti - zpracování formuláře

Framework	**Počet řádků**
`CakePHP`	14
`Nette`	14
`Zend 2`	39

3.2.4 Twitter Bootstrap

K programování aplikace byl využit frontendový framework Twitter Bootstrap verze 2.3. Tento framework urychluje práci s HTML, CSS i JavaScriptem. Pojeví se na moderním vzhledu aplikace a rychlosti vývoje [2]. Bootstrap lze vyexportovat s jistou konfigurací, není nutné při použití pouze menšiny nástrojů využívat kompletní zdrojový kód frameworku.

Vybíral jsem pouze ze dvou hlavních frontendových frameworků a tím byly již zmíněný Bootstrap a Foundation. Výhodou Foundation je, že poskytuje pouze kostru pro programátora. Je velice flexibilní, lze v něm prvky lépe upravovat dle vlastních požadavků. V Bootstrapu je tomu naopak, pokud požadujeme pouze úpravu určitých vlastností tlačítka, jednoduší je napsat celé tlačítko pomocí CSS znovu. V aplikaci nebylo nutné měnit výchozí styl, protože tento u Bootstrapu je dostatečně propracovaný i rozsáhlý a zadavatel nezasahoval nikterak do vývoje grafického rozhraní, proto byl zvolen Bootstrap. Dalším kritériem byla podpora Bootstrapu 2.3 v CakePHP v podobě pluginu, který je detailněji popsán v kapitole 4.9. Z tohoto důvodu jsem použil starší verzi Bootstrap 2.3, i když je dostupná nová verze 3.0.

3.2.5 JQuery

Nedílnou součástí Bootstrap frameworku je knihovna JQuery [3]. V aplikaci je použita ve verzi 2.1. Tato knihovna usnadňuje tvorbu animací, událostí a práci s AJAX (Asynchronous JavaScript and XML). V aplikaci byla využita pro moderní, interaktivní uživatelské rozhraní. Hlavní využití JQuery v aplikaci mimo jiné je v provázející nápovědě (více v kapitole 4.12) a v sekci vybírání témat studenty, kde k filtrování je využit AJAX pro asynchroní komunikaci se serverem.

4 Realizace aplikace

4.1 Metodika vývoje aplikace

Vývoj aplikace lze přirovnat k vodopádové metodice. Při vývoji nedocházelo k častým změnám požadavků, a proto vodopádovou metodiku vývoje softwaru bylo možné použít [5]. Obrovskou výhodou je, že zadavatelé jakožto tajemník ústavu MTI a vedoucí mé bakalářské práce jsou z oboru IT a tudíž obeznámeny s problematikou zadávání softwaru. Požadavky na aplikaci, které mi byly poskytnuty, byly dostatečně rozsáhlé i ucelené a zahrnovaly věcné poznámky k aplikaci. Nedocházelo tedy k problematice, kde zadavatel kvalifikovaně neposkytuje dostatečnou specifikaci požadavků a tudíž nedocházelo k výrazným změnám v návrhu aplikace.

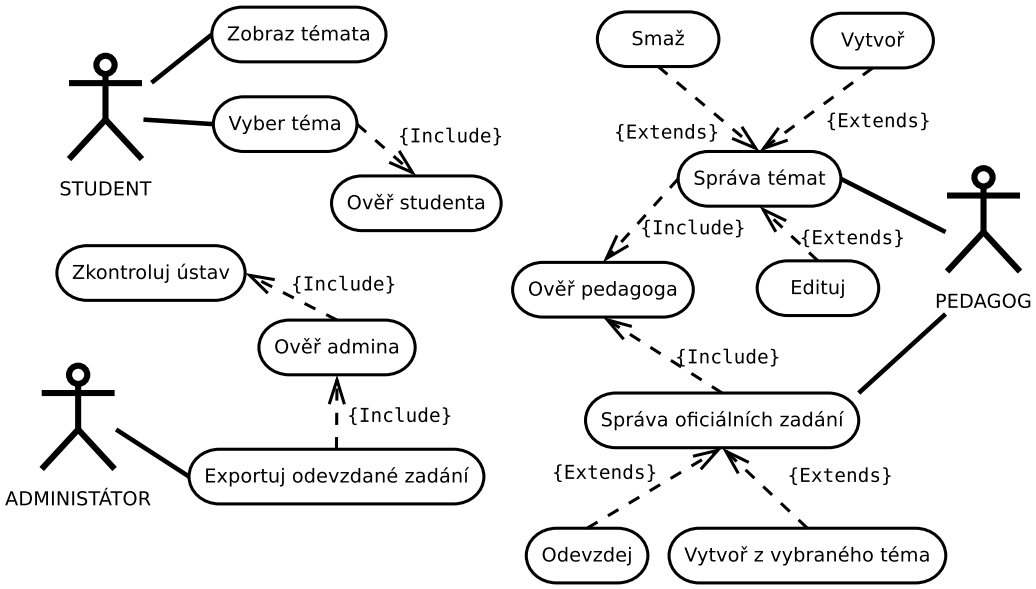
Na základě požadavků bylo vytvořeno uživatelské rozhraní aplikace v HTML. V podobě prezentace bylo toto uživatelské rozhraní předvedeno tajemníkovi a vedoucímu ústavu MTI. Po odsouhlasení byl vytvořen návrh aplikace a databázového modelu (více v kapitolách 4.3 a 4.5). Tento návrh byl implementován a následně otestován. V následujících kapitolách jsou jednotlivé kroky vývoje detailněji popsány.

4.2 Uživatelské rozhraní

Dle požadavků byla aplikace rozčleněna na jednotlivé pohledy, které představovaly hlavní součásti aplikace. Tyto pohledy byly následně rozděleny do oblastí, kde každá oblast představovala akci daného pohledu. V této fázi ještě nebyl vytvořen databázový diagram, proto vznikl ještě mezikrok v podobě jednoduchého UseCase diagramu (viz obrázek 7), který usnadnil vytváření pohledů a daných oblastí představující v uživatelském rozhraní hlavní menu.

Pohledy v diagramu jsou všechny aktivity, které jsou spojeny přímo s aktérem, například „Správa témat“ u aktéra označený „PEDAGOG“. Oblast znázorňuje aktivitu v daném pohledu, například aktivita „Vytvoř“ v pohledu „Správa témat“. Po definování pohledů a oblastí následovalo rozdělení, které budou typu landmark[1] a které budou možné zobrazit pouze danému uživateli. To představuje v diagramu aktivitu, ke které je vkládána akce začínající slovem „validuj“. Na základě pohledů a oblastí došlo k vytvoření struktury aplikace pouze v HTML podobě bez implementovaných funkcí. Všechny typy uživatelů sdílejí stejný design pohledů. Nebylo nutné design rozlišovat, pouze dochází ke změně přístupných položek v hlavním menu.

[1] Oblast či pohled, který je přístupný ze všech uživatelů.



Obrázek 7: Use Case diagram funkcí aplikace

4.2.1 Responzivní design

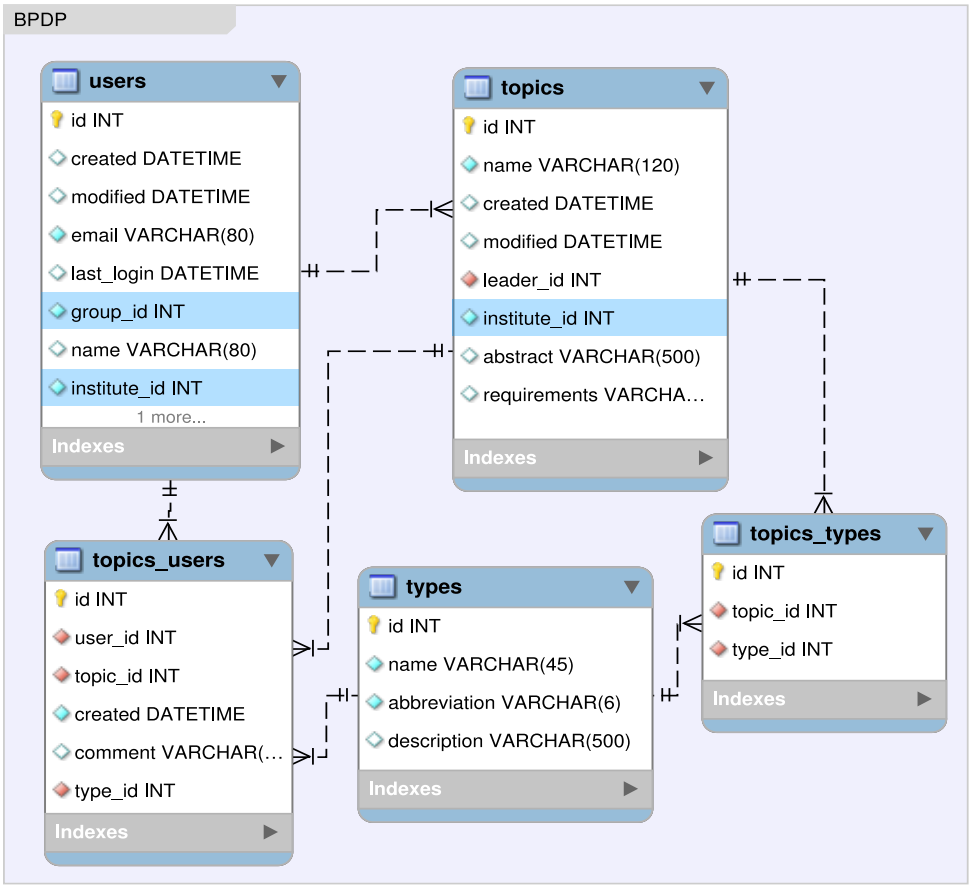
Framework Bootstrap (popsaný v kapitole 3.2.4) poskytuje podporu pro responzivní design. Tento design pomocí CSS optimalizuje zobrazení aplikace dle zobrazovacího média (tablet, desktop pc či mobilní zařízení). Využívá se flexibility velikostí objektů pomocí procentuálních hodnot. Dále se využívá „media queries“, které mění styl dokumentu na základě šířky zobrazovacího média. Bootstrap má již připravené styly k HTML elementů pro responzivní design. V aplikaci byl tento design využit mimo jiné například v hlavním menu v podobě HTML elementu pro list.

4.3 Databázový model

Databázový model byl základ pro tvorbu struktury aplikace. Při jejímž návrhu se vycházelo z vývojového diagramu, který je popsán v kapitole 3.1 a také z UseCase diagramu 7 z kapitoly 4.2. Pro vývoj databázového modelu bylo použito vývojové prostředí MySQL Workbench. Obrázek 8 znázorňuje pouze část databázového modelu, která realizuje funkci burzy při vybírání témat.

Na obrázku tabulka `topics` představuje témata studentských prací. S tabulkou `users` sdílí několik vazeb. První je typu `1:N`. Tato vazba přestavuje cizí klíč v tabulce `topics` s názvem `leader_id`. Další vazbou je `M:N`, která je realizována tabulkou `users_topics`. Tato tabulka slouží k zaznamenání zažádání tématu. Studenti si tedy mouhou žádat více zadání. S touto vazební tabulkou je také svázána tabulka `types`, která upřesňuje o jaký typ

práce student žádal (BP, DP atd.). Poslední tabulka `topics_types` představuje vazbu `M:N` mezi `topics` a `types`. Nese záznamy o tom, jaké typy prací lze u daného tématu zažádat. U všech cizích klíčů v této ukázce je nastavena vlastnost `cascade` při aktualizaci záznamu a vlastnost `restrict` při pokusu o smazání záznamu.



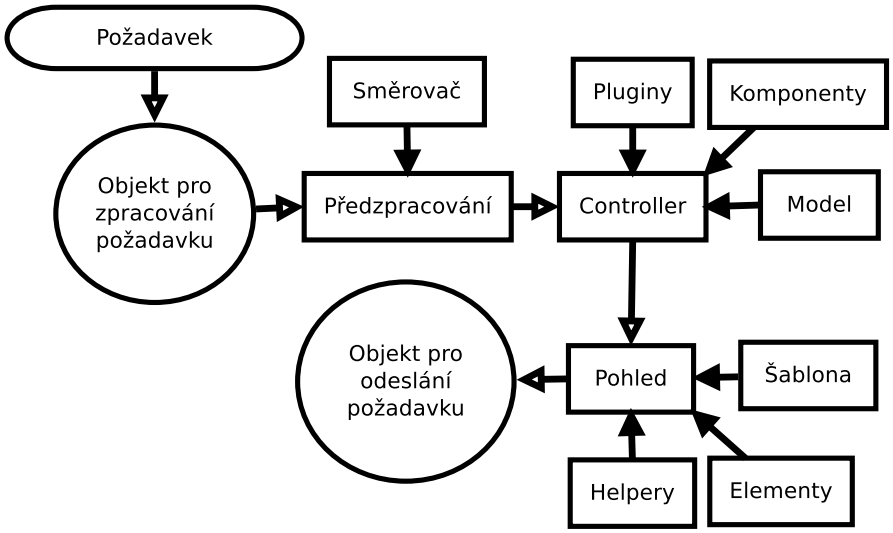
Obrázek 8: Ukázka části databázového modelu

V databázovém modelu je použit jeden trigger u tabulky `departments`, která přestavuje stromovou strukturu fakulty. Tato tabulka je svázána vazbou `M:N` k tabulce `topics`, udává pro jaké obory je téma určené. Trigger zajišťuje při smazání jedné úrovně ve struktuře fakulty (například odebrání jednoho oboru) přiřazení tématu do úrovně vyšší než byl smazaný záznam. V aplikaci nelze smazat záznam nejvyšší úrovně, proto téma bude přiřazeno vždy v minimálně jednom oboru nebo v určité skupině oborů.

4.4 Aplikační framework

4.4.1 Zpracování požadavku

Zpracování požadavku frameworkem CakePHP znázorňuje následující obrázek 9. Po přijetí požadavku HTTP se vytvoří objekt typu `RequestObject`. Následně dochází ke předzpracování tohoto objektu pomocí směrovače, který slouží k propojení daného URL na konkrétní controller. Ve směrovači lze uvést i alternativní pravidla pro směrování, například určité skupiny URL adres, specifikované regulárním výrazem, budou přesměrovány na jinou URL adresu. Dále se požadavek předá řídícím logikám aplikace. Díky směrovači je zavolán správný controller společně s jeho akcí, ve které dochází k manipulaci dat v podobě modelových tříd. V controlleru se používají také pluginy a komponenty, kterým se věnují kapitoly 4.7.2 a 4.9 Obsah html stránky se generuje za pomocí helperů, elementů a předdefinovaných šablon pohledů a předaných dat z controlleru [1].



Obrázek 9: Zpracování požadavku

4.4.2 Konfigurace CakePHP

CakePHP pracuje s přepisováním URL (URL rewriting), který slouží k přesměrování na jinou URL nebo podstrčení URL uživateli. Na serveru provozující aplikaci bylo nutné uvést do provozu modul Apache `mod_rewrite`. Všechny konfigurace přepisování URL se provádějí pomocí souboru `.htaccess`. Pro validní běh frameworku bylo důležité nastavit parametr `RewriteBase` u všech `.htaccess` souborů v aplikaci, protože kořenový adresář virtuálního serveru se nenachází v kořenovém aresáři aplikace, ta se nacházela v adresáři `mti.tul.cz/home/vitvarova/bpdp/login/`. Kořenový adresář virtuálního serveru je

`mti.tul.cz/home/vitvarova`, parametr `RewriteBase` musí být nastaven na `/bpdp/login`.

4.4.3 Konvence a názvosloví

Od počátku programování s CakePHP bylo vhodné dodržovat konvence názvů. CakePHP má konvence na všechny prvky aplikace – třídy, funkce, proměnné, tabulky v databázi i jejich sloupce atd. Díky dodržení těchto konvecí framework poskytoval mnoho užitečných funkcí bez mapování či detailních nastavení. Pro tyto potřeby je v CakePHP připravená statická třída `Inflector`, díky níž je jednoduché převádět například mezi názvoslovím řídících logik a modelů, kde v modelech se využívá jednotného čísla s velkými písmeny a v řízení se využívá množného čísla s malými písmeny [1].

4.5 MVC a adresářová struktura

Framework CakePHP přímo vybízí k používání MVC. Díky adresářové struktuře jsou jednotlivé vrstvy přehledně odděleny, nevzniká tedy část kódu, který by obsahoval práci s databází, ovládací prvky a výpis na obrazovku. Adresářovou strukturu popisuje následující obrázek 10.

```
app
├── Controller
│   ├── Components ..... Adresář pro vlastní komponenty.
│   ├── AppController.php
│   ├── ThesesController.php
│   └── ...
├── Model
│   ├── AppModel.php
│   ├── Thesis.php
│   └── ...
├── View
│   ├── Elements ..... Strukturovaný adresář elementů.
│   ├── Layouts ..... Adresář pro ukládání šablon pohledů.
│   ├── Thesis
│   │   ├── add.ctp ..... Pohled pro přidání zadání.
│   │   ├── edit.ctp
│   │   └── ...
│   └── ...
└── ...
```

Obrázek 10: Adresářová struktura aplikace

4.6 Modelová vrstva

Modely jsou třídy, které pracují s daty aplikace. Všechny modelové třídy dědí od `AppModel` a jeden model reprezentuje tabulku v databázi. Pouze deklarací modelové třídy jsou implementovány metody pro ukládání, mazání a dotazování dat. Užitečnou funkcí modelů je automatické ukládání datumu a času vytvoření či změny provedené v daném záznamu. Stačí pouze vytvořit v tabulce atributy `created` a `modified` typu `DATETIME` [4].

4.6.1 Relační mapování

Relační mapování je nutné nastavit ručně v každém modelu. Vytváří se pomocích atributů třídy. Pro relační vztahy `1:N`, `N:1` a `1:1` jsou atributy `hasMany`, `belongsTo` a `hasOne`. Tyto proměnné obsahují asociativní pole, které definuje pravidla pro relační vztah. Klíč v poli reprezentoval jiný model, hodnota představovala další asociativní pole, kde se definovaly například podmínky spojení nebo způsob řazení podle toho, jaký vztah je využíván. Struktura pole je pevně definovaná dokumentací k CakePHP. V modelových třídách nebyl použit relační vztah `M:N`, jelikož CakePHP tento vztah výrazně zpomaloval. Všechny vztahy `M:N` byly realizovány dvěma vztahy `1:N`.

4.6.2 Validace dat

Validace dat v modelové vrstvě funguje podobně jako relační mapování. Je tvořena asociativním polem uloženým v atributu `validate`. Klíče v poli jsou názvy sloupců v databázové tabulce a hodnoty jsou další asociativní pole obsahující omezení. Ty jsou tvořeny klíči `rule`, `allowEmpty`, `message` a další. Na základě těchto validací CakePHP automaticky vygeneruje ošetření formuláře. Proto je nutné uvádět u každého omezení klíč `message`, aby se uživateli zobrazila zpráva o tom, jaké pole musí opravit. Tímto klíčem se jistým způsobem narušuje MVC struktura aplikace. Zprávy pro uživatele by se měly definovat na `View` vrstvě. Na modelové by mělo dojít pouze k výjimce při nedodržení daných omezení.

Následující ukázka kódu 1 představuje modelovou třídu `Thesis`. Tato třída slouží k správě záznamů o oficiálních zadání a obsahuje relační mapování i validaci dat, proto je ideální k demonstraci výše popsaných vlastností modelových tříd.

Ukázka kódu 1: Modelová třída pro oficiální zadání

```
1 class Thesis extends AppModel {
2 public $hasMany = array( // provazani relaci 1:N
3         'Requirement', 'Literature', ...
4 );
```

```
5  public $belongsTo = array(
6          'User', 'Type', ... // provazani relaci N:1
7  );
8  public $validate = array(
9    'name' => array( // validacni block pro atribut name
10     'required' => array( // pojmenovani validace
11       'rule' => 'notEmpty', // hodnota nesmi byt prazna
12       //zprava uzivateli pri nesplneni validace
13       'message' => 'Nazev oficialniho zadani musi byt vyplnen.'
14          ),
15     'size' => array(    // pojmenovani validace
16       'rule' => array('between', 10, 100),
17       ...
18     )));
19 }
```

4.7 Řídící logika

V řídící logice, představující v MVC `controller`, se nachází kompletní logika aplikace a práce s modely. Každá hlavní modelová třída, která v aplikaci představuje modul[2], má řídící logiku. Pro jeden model je jedna řídící logika v podobě třídy, která dědí od hlavní třídy `AppController` a je uložená v souboru `./app/Controller/NazevModeluController.php`. Ve třídě `AppController` lze definovat společné chování všech řídících tříd.

4.7.1 Akce

Veřejné metody `controlleru` jsou akce, ke kterým lze v aplikaci přistoupit na URL `./nazevControlleru/nazevAkce`. V této akci se nachází práce s modely a výstup je předáván zobrazovací vrstvě. Akce přebírá v zásadě pouze žádný nebo jeden parametr, který slouží k identifikaci aktuální položky, na které se má akce vykonat (např. `./theses/edit/33` provádí editaci záznamu s `ID` 33. V akcích se přistupuje také k `protected` proměnné `AppControlleru` typu `RequestObject`, která obsahuje všechna data ohledně požadavku - kompletní hlavičku a formulářová data.

[2]Modul aplikace je sekce, kde uživatel pracuje s databázovým modelem v podobě editace, mazání a vkládání záznamů

4.7.2 Komponenty

Řídící logika také používá komponenty. Jsou to části logiky, které mohou být sdílené všemi nebo jedním Controllerem. V aplikaci byly využity komponenty implementované již ve frameworku – `Session`, `Paginator`, `Sql` a další. Komponenta `Paginator` je využívána ke stránkování seznamu témat. Výpis je implementován ve funkci s názvem `view` a nachází se ve třídě `TopicsController`. Díky této komponentě je do seznamu automaticky zakomponováno stránkování, stačí již vytvořit odkazy na danou akci, které zahrnují číslo stránky, počet záznamů na stránku, směr a typ řazení. Komponenta tyto parametry typu GET zpracuje a vyselektuje záznamy v modelové třídě. Ve výpisu témat se používá komponenta `Paginator` společně s komponentou `RequestHandler`, díky níž lze využít AJAX a dosáhnout tedy asynchroního stránkování. CakePHP rozpozná stránkovací odkaz podle parametrů, které jsou jim předávány a vygeneruje k nim JavaScriptové metody, které zpracovávají asynchronní datový přenos a aktualizují daný HTML objekt.

4.8 Pohledy

Pohledy jsou části kódů, které renderují obsah HTML stránky. Data používaná pro zobrazení se předávají pohledům pomocí vlastní metody controllerů jménem `set(nazev, hodnota)`, která přebírá název proměnné, ke které lze v pohledu přistoupit. Hodnota je nejčastěji v podobě asociativního pole. Tvorbu HTML obsahu usnadňují šablony pohledů, helpery a elementy.

4.8.1 Šablony pohledů

Šablony pohledů (layouty) tvoří základ pohledu a sdílí se mezi všemi pohledy. V této šabloně je definovaná hlavička HTML dokumentu a veškerý statický obsah. Pohledy jsou do této šablony vkládány. V aplikaci jsou vytvořené tři šablony. První je nejvíce používaná, obsahuje hlavní stránku aplikace. Druhá šablona slouží pro AJAX komunikaci, zde není třeba tvořit kompletní hlavičku a tělo HTML stránky, nýbrž jen specifikovat, kam se má pohled vložit. Vkládání pohledů se realizuje funkcí `$this->fetch('content')`. Poslední pohled slouží jako šablona pro emailovou komunikaci.

4.8.2 Helpery

Díky helperům je kód renderující obsah stránky čisčí a přehlednější. V aplikace jsou využívány helpery `HtmlHelper`, `JSHelper`, `FormHelper`, `CsvHelper` a další. Helpery jsou

doplňující třídy frameworku, ale lze si napsat i vlastní. Každý helper je nutné před použitím inicializovat v controllerech, přidáním názvu helperu do pole v proměnné `helpers`.

Ukázka kódu 2: Použití helperů

```
1  $this->Js->get('#add-button');
2  // pridani udalosti na element s id "add-button"
3  $this->Js->event('click', 'add_button_clicked()');
4  // vystupem je <button class="btn .."">Pridat</button>
5  echo $this->Html->tag('button', __('Pridat'), array(
6      'class' => 'btn btn-success',
7      'id' => 'add-button',
8      'type' => 'button'
9  ));
10 echo $this->Form->input('', array(
11     'type' => 'hidden',
12     'id' => 'counter',
13     'value' => 0
14 ));
```

Ukázka 2 naznačuje, jak helpery zpřehlední kód. Funkcím helperům se předávají vlastnosti elemetů jako parametry asociativního pole. Tento kód vytvoří událost `click` na později vygenerované tlačítko s ID `add-button`. Poslední příkaz `echo` vygeneruje skryté formulářové pole. Pokud není před generováním polí formulářů dříve zavolaná funkce pro vytvoření formuláře, framework automaticky vytvoří hlavičku formuláře s výchozím nastavením[3].

4.8.3 Elementy

Elementy se využívají k eliminování duplicitního kódu v pohledech. Pro použití elemetu slouží funkce `element`, která přebírá název elementu a předávaná data. Element musí být umístěn v adresáři `Elements` a může být v hierarchii podadresářů uložen libovolně hluboko. Při předávání názvu elementu funkci, je nutné tuto cestu specifikovat. Předáváná data jsou znovu v podobě asociativního pole.

V aplikaci jsou jako elementy vytvořeny většiny formulářů. Bylo by zbytčné duplikovat kód, jelikož stejný formulář je využit pro vyváření i editaci záznamů. Elementy se využívají u většiny tlačítek. Jelikož kód různých tlačítek se liší pouze v pojmenování, stylu a akcí. Byl tento kód vložen do elementu a všechny vyjmenované parametry se předávají asociativním polem. V elementu toto asociativní pole vytvoří dle klíčů proměnné s daným obsahem.

[3]Například parametr action je nastaven na aktuální URL.

4.8.4 Sdílení pohledů

Užitečnou vlastností frameworku je sdílení pohledů. Tato vlastnost zamezuje duplicitu kódů a podporuje sdílení kódů mezi pohledy. Tvoří se vytvořením samostatného adresáře ve složce `View` a vložením souborů (pohledů), které sdílejí stejný obsah s ostatními pohledy. Zavoláním příkazu `$this->extend('/Common/edit');` se vloží příslušný kód sdíleného pohledu a lze následně tento kód dále rozšiřovat.

Následující ukázka kódu 3 naznačuje sdílený kód pro formulář k vytváření záznamů libovolného modelu. Nejdříve je nutné zjistit, na jakém modelu se tvorba provádí. K tomu slouží název controlleru a statická třída `Inflector`. Následně je vytvořena hlavička a vložen obsah formuláře daného modelu. V poslední řadě je vloženo potvrzovací tlačítko a ukončení formuláře.

Ukázka kódu 3: Sdílený kód pro editační formulář

```
1  // z hodnoty "theses" => "Thesis"
2  $model = Inflector::classify($this->params['controller']);
3  // vlozeni elementu pro vytvoreni formulare
4  echo $this->element('forms/create', array(
5      'model' => $model,
6  ));
7  // dle controlleru vlozi prislusny formular pro editaci
8  echo $this->element('forms/'.strtolower($model));
9  //Odeslani formulare
10 echo $this->element('forms/submit');
```

4.8.5 Jazykové variace

Všechny řetězce z elementů, šablon i pohledů musí procházet filtrovací funkcí například `echo __('Pridat')`. Tato funkce slouží k lokalizaci aplikace. Jedním z hlavním požadavků na aplikaci byla možnost rozšíření na jinou jazykovou verzi. Pro úplné dokončení tohoto požadavku musí být napsány překladové soubory. Tyto soubory obsahují klíč v podobě řetězce ve výchozím jazyku aplikace, v tomto případě českým a hodnota je překladový jazyk. Problematiku zavádí databázová data. Pro tu je ve frameworku CakePHP připravena komponenta. Jenž na základě jazykové lokalizace vybere tabulku, která obsahuje překlad výchozí tabulky. Je nutné tedy vytvořit z databázových tabulek, které mají být přeloženy kopie. Tyto tabulky obsahují pouze index a řetězcové hodnoty z výchozí tabulky, tzn. všechny číselné hodnoty se nepřekládají (např. cena nebo počet). Tyto tabulky jsou tedy svázány vztahem `1:1`. K vytvoření překladových tabulek poslouží užitečná konzole

frameworku.

4.9 Pluginy

Díky rozsáhlé komunitě frameworku CakePHP je k dispozici mnoho pluginů. Pluginy jsou umístěny ve zvláštní složce mimo aplikaci a nikterak nezasahují do jádra frameworku. Tyto pluginy obsahují identickou adresářovou strukturu aplikace s definovanými komponentami či helpery. Načtení pluginu se definuje v konfiguračním souboru `app/Config/bootstrap.php` příkazem `CakePlugin::loadAll();`, inicializace probíhá v rodičovském controlleru. Pokud má komponenta pluginu nahradit výchozí komponentu frameworku, je nunté toto nastavení specifikovat asociativním polem, kde klíč je výchozí komponenta a hodnota je `jmenoPluginu/jmenoComponenty`.

4.9.1 BoostCake

`BoostCake` je plugin, který byl vytvořen pro používání CakePHP frameworku s frontendovým Twitter Bootstrap frameworkem. Tento plugin obsahuje nové helpery, které dědí od výchozích helperů CakePHP. Rozšiřují základní HTML elementy o nastavení CSS tříd, tak aby výchozí HTML element využíval předdefinovaný vzhled frontendového frameworku.

4.9.2 Vlastní plugin

`BoostCake` obsahuje mnoho užitečných implementací, ale jeho vývoj pro Bootstrap 2.3 byl už zastaven a pokračuje pro verzi 3.0. Plugin pro tuto verzi je teprve v počátcích a mnohem méně implementovaný než pro verzi 2.3. Proto byla pro vývoj aplikace zvolena starší verze Bootstrapu. Plugin byl tedy použit pouze jako šablona pro vlastní plugin jménem `BoostCakeEx`. Zde je implementována podpora pro zobrazení upozornění uživateli. Upozornění mají čtyři úrovně - `success`, `warning`, `info` a `error`. Dále v `BoostCake` sice již byla vytvořena podpora vzhledu pro stránkování, ale bylo možné použít pouze kulatá tlačítka. Vzhled aplikace by nevypadal konzistentně, proto ve vlastním pluginu byl tento element znovu definován.

4.10 System rolí

Ze zjednodušeného UseCase diagramu na obrázku 7 je zřejmé, že v aplikaci figurují tři typy uživatelů - student, pedagog a administrátor.

Student má oprávnění listovat v seznamu témat. Následně požádat o určitý počet z nich. Nakonec vyexportovat dokončené oficiální zadání. Pedagog má oprávnění přiřazovat studenty ke svým tématům, mazat studenty z front o svá téma. Dále má možnost editovat svá témata, vytvářet, editovat a exportovat oficiální zadání. Administrátor má oprávnění vytvářet, editovat i mazat jakékoliv téma i oficiální zadání, dále mazat studenty z front, upravovat nastavení aplikace. Toto nastavení zahrnuje správu ústavů, typů prací, alternativních oprávnění, frontové limity a další. Administrátor má oprávnění exportovat vyhotovená oficiální zadání v jakémkoliv z dostupných formátů.

K realizaci zabezpečení přístupů do jednotlivých sekcí lze v CakePHP dosáhnout třemi způsoby - ACL, scaffolding a autorizační komponentou. ACL je nejsofistikovanější řešení, ale také nejvíce rozsáhlé. Vytvoří v databázi tabulky `acos` a `aros`, které představují dvourozměrné pole, kde jedna strana matice tvoří seznam všech funkcí aplikace a druhá strana tvoří všechny role. Pro každou funkci lze vybrat jednotlivé role, kterým je přístup udělen. Tato metoda je velice neflexibilní, pokud plánujeme často přidávat funkce do aplikace, jelikož se musí přidávat každá tato funkce do tabulky s `aros`. Pro potřeby aplikace byla tato metoda příliš komplexní a rozsáhlá. Další metodou je scaffolding. Tato metoda rozdělí aplikaci na sekce - například administrátor a běžný uživatel. Administrátor přistupuje ke svým funkcím prostřednictvím URL, kde před názvem controlleru je tzv. `scaffold` v tomto případě `admin` - `URL/admin/nazevControlleru/akce`. Nevýhodou je sdílení funkcí mezi rolemi, jelikož každá funkce, ke které má přístup pouze administrátor má prefix `admin_`. Posledním způsobem a v aplikaci využitým je autorizační komponenta.

4.10.1 Autorizační komponenta

S autorizační komponentou lze snadno dosáhnout sdílení funkcí mezi rolemi. Využívá se připravené komponenty frameworku zajišťující omezení přístupu k akcím podle dané role. Při inicializaci je nutné nadefinovat, že autorizační proces probíhá v controlleru, jelikož tuto komponentu lze využít i k jinému způsobu autorizace. V poslední řadě se nastavuje, co má uživateli zobrazit v případě, že autorizace selhala a kam se má aplikace přesměrovat po úspěšném přihlášení, to zajišťují parametry - `authError` a `logoutRedirect`.

Autorizace se provádí pomocí funkce `isAuthorized($user)`, která musí být implementována v každém controlleru. Lze tuto funkci vytvořit i v hlavním controlleru, který je následně rozšiřován. Zde se nastaví obecná autorizace, tzn. student a pedagog nemají přístup nikam, pouze administrátor má veškerá opránění. Tato funkce přebírá parametr `$user`. V této proměnné je uložena datová struktura databázové tabulky `users`, tudíž všechny údaje o uživateli včetně toho, jaká role danému uživateli náleží. Privátní pro-

měnná controlleru `$params` obsahuje údaje o dotazu. Na základě akce a role uživatele se tedy autorizuje (funkce vrátí `TRUE`) či ne.

4.10.2 Shibboleth

Díky systému Shibboleth lze využívat jedny přihlašovací údaje na všech webových aplikacích univerzity. Při přístupu na chráněný obsah je uživatel přesměrován na poskytovatele identity, pokud ho již poskytovatel služeb nezná. Ten zjistí, zda uživatel se prokázal správnou totožností a následně přesměruje zpět cílovému serveru a navíc předá data o uživateli. Data, která bude poskytovatel identity sdílet, jsou správcem definovány [6]. Díky tomu lze následně identifikovat uživatele - jeho email, zda je to student či vyučující, popřípadě z jaké fakulty a ústavu i další. Data jsou předávána v super-globálním poli `$_SERVER`.

Zjištění, zda je uživatel student nebo pedagog, probíhá na základě parametru `affiliation`. Kombinací hodnot `staff@tul.cz`, `faculty@tul.cz`, `member@tul.cz`, `employee@tul.cz` a `student@tul.cz` lze zjistit, zda je uživatel učitel, učící doktorand, příležitostný učitel bez smlouvy, ostatní personál, student nebo student na výměnném pobytu. Například učící doktorand má kombinaci - faculty, member, employee a student.

Pro potřeby aplikace stačí rozlišovat mezi studentem a pedagogem. Tedy sledování hodnoty student. V nastavení aplikace lze nadefinovat jaké kombinace jsou pro danou roli akceptující. Administrátor je autorizován samostatně pouze prostřednictvím emailu.

4.10.3 Altertnativní přidělení oprávnění

Administrátor má možnost udělit oprávnění manuálně. Toto přidělení oprávnění má vyšší prioritu. Nebere se tedy ohled na parametr `affiliation` v případě že v databázové tabulce `users_alter` se email přihlašujícího uživatele nachází. Data alternativním přidělování oprávnění se provádí v uživatelském rozhraní aplikace, kde administrátor předává seznam emailových adres a nastavuje jim selektovaným menu roli pro přístup.

4.11 Export dat

Export skupin oficiálních zadání se provádí do tří formátů - TXT, XML a CSV. Export TXT a CSV se ve frameworku CakePHP provádí obdobně jako klasický požadavek. S rozdílem, že v akci (například export) je nastaveno, že při serverové odpovědi má dojít ke stažení souboru příkazem `$this->response->download($fileName.$fileType)`. Pohled

exportů se moc neliší, pouze v nepoužívání HTML elementů, pužitím jiné šablony a specifikaci typu odpovědi příkazem `$this->response->type('Content-Type:text/csv')`. V exportu skupin oficiálních zadání lze vybrat, jaká data požadujeme vyexportovat (například pouze názvy a autory) a jejich pořadí.

K exportu XML se používá komponenta `RequestHandler`, u které se nastavují typy výstupu pomocí funkcí `respondAs` a `renderAs`. Jelikož framework předává data pomocí asociativního pole, který má stejnou strukturu jako exportovaný XML soubor. Tato struktura je popsána ve schématu na adrese http://www.mti.tul.cz/bpdp/theses-export-scheme.xsd, schéma uvádí všechny možnosti exportovaných hodnot.

Aplikace umožňuje pro vyučující i studenty exportovat vlastní práce do formátu PDF. K tomuto exportu se využívá třída `MPDF`, která převádí HTML kódované v UTF-8 do PDF souboru. Třída `MPDF` byla uvolněna pod licencí GPL. Pro používání třídy s frameworkem CakePHP byla napsána jednoduchá komponenta se stejným názvem, která je v aplikaci vyžívána.

4.12 Nápověda

Nápověda aplikace provází uživatele celým procesem výběru prací. Popisuje jaké funkce jsou v daném stavu procesu k dispozici a popřípadě, jak má pokračovat. Vytvářet komplexní uživatelskou příručku, která zahrnuje všechny funkce, je z mé zkušenosti zbytečné, protože žádný aktér tyto příručky nestuduje. Po uživatelském rozhraní se požaduje, aby bylo intuitivní a snadno ovladatelné.

Pro zobrazení nápovědy byl vytvořen element, který využívá Bootstrap komponentu `Modal`. Ten představuje jednoduchý informační dialog. Element vybírá nápovědu k zobrazení a přidává JavaScriptovou událost na tlačítko nápovědy, které je umístěné téměř v každém pohledu. V závislosti na URL element rozpozná daný controller i jeho akci a následkem toho, zobrazí příslušnou nápovědu s popisem funkcí a možných dalších kroků.

4.13 Testování aplikace

Testování aplikace probíhalo třemi úrovněmi. V první fázi testování se prováděly jednotkové testy na autorizačních funkcích a autentizační komponenty. V druhé fázi se provádělo částečné integrační testování. Poslední fázi vývoje se přešlo k akceptačnímu testování, kde docházelo k testování uživatelského rozhraní. Na samotné jádro frameworku CakePHP nebylo provedeno žádné testování.

4.13.1 Jednotkové testování

Jednotkové testování se provádělo pomocí frameworku PHPUnit. Ke specifikování parametrů testování posloužily konfigurační soubory XML. Tyto soubory zahrnovaly část možných kombinací vstupních parametrů. Výsledky testování se ukládaly do logovacích souborů, které jsou dostupné v adresáři `/app/tmp/`.

4.13.2 Integrační testování

Integrační testování zahrnovalo ověření správné komunikace mezi modelem a controllerem. Pro testování byl vytvořen jeden controller, který zahrnoval funkce pro otestování každého z použitých modelů, který obsahuje jiné nastavení než výchozí. V každé funkci byla vytvořena lokální prázdná databáze, do které pomocí testového buildu byla nahrána simulační data. Testovalo se především správnost nastavení modelů, tzn. mapování relačních vazeb a vlastní dotazové funkce.

4.13.3 Akceptační testování

Akceptační testování bylo prováděno dvěmi nezávislými osobami, které nikterak nesledovali průběh vývoje aplikace. V této úrovni se testovalo pouze uživatelské rozhraní. Testovatelé postupně procházeli všechny dostupné pohledy ze všech možných rolí. Dále akceptační testování zahrnovalo testování v čtyřech nejpoužívanějších prohlížečích pro desktop počítače - Firefox, Chrome (Chromium), Internet Explorer verze 8 a verze 9 a Opera. Pro všechny tyto prohlížeče je aplikace funkční. V poslední fázi akceptačního testování se provávděly testy na mobilních zařízeních. K dispozici byl mobilní telefon s operačním systémem Android a displayem 4,9"v prohlížeči Firefox Mobile a Chrome, dále mobilní telefon s operačním systémem iOS a displayem 4"v prohlížeči Safari. Nakonec se testovalo na tabletu s Android OS (display 10.1"). Na mobilních zařízeních se testovalo chování responzivního designu. Chyby aplikace, které projevilo akceptační testování byly opraveny.

5 Závěr

Stávající proces zadávání studentských prací a projektů byl analyzován na základě rozhovorů s tajemníkem a vedoucím ústavu MTI. Tato analýza pomohla identifikovat části procesu, které bylo nutné automatizovat tak, aby se nový proces výrazně zjednodušil a urychlil. Identifikace se prováděla na základě vývojových diagramů představujících stávající proces. Software nahradil části procesu, které zaváděly duplicitu práce a zbytečnou složitost. Nový proces s prvky automatizace v podobě wireframů byl prezentován a následně odsouhlasen zadavatelem.

V rámci práce byla vytvořena webová aplikace, která plně podporuje nový navržený proces automatizace. Aplikace splňuje požadavky na bezpečnost aplikace. K autentizaci je využit přihlašovací systém Shibboleth a systém rolí v aplikaci je vhodně navržen, tak aby splňoval požadavky všech skupin uživatelů - administrátorů, pedagogů a studentů. Aplikace zahrnuje i možnost alternativního rozdělení oprávnění administrátorem pro každého uživatele zvlášť.

Pro administrátory je připraveno rozhraní poskytující přehled o vytvořených či vybraných tématech a možný export všech zadání jednotlivých ústavu, který lze dle požadavků konfigurovat. Software umožňuje exportovat oficiální zadání do souboru PDF, XML, TXT a CSV. Pedagogové mají k dispozici přehlednou správu témat a oficiálních zadání. Studenti vybírají témata z přehledného, strukturovaného a centralizovaného soupisu témat. Tento list lze filtrovat dle různých parametrů, například dle oblíbeného vyučujícího, klíčových slov, oboru, pro které je zadání určeno a další.

Aplikace je navržena tak, aby splňovala požadavky na rozšiřitelnost aplikace. Přímo v uživatelském rozhraní administrátorů lze aplikaci rozšířit o další ústavy na fakultě. Aplikace také splňuje požadavek možnosti rozšíření na internacionalizovaný software (i18n). Pro zmezinárodnění jsou připraveny databázové tabulky. Všechny řetězce prochází filtrovací funkcí, ke kterým je nutné pro uskutečnění rozšíření doplnit překladové soubory.

Aplikace je dokončená a připravená pro testování v reálném provozu na adrese `http://www.mti.tul.cz/bpdp/`. V simulačním provozu bylo provedeno jednotkové, integrační a akceptační testování. Akceptační testování bylo prováděno nezávislými osobami. Na základě těchto testů byly odhaleny a následně opraveny chyby aplikace. Dále byla vytvořena nápověda k aplikaci, která provází uživatele v průběhu procesu a specifikuje další postup. Uživatelské rozhraní je jednoduché, nativní a obsahuje responzivitu, možnost aplikace přizpůsobit rozhraní zobrazovacímu médiu.

Použitá literatura

[1] CAKE DEVELOPMENT CORPORATION. *CakePHP: The rapid application development framework for PHP* [online]. [cit. 2014-04-05]. Dostupné z: `http://cakephp.org/`

[2] Bootstrap. EFRON, Bradley a Robert TIBSHIRANI. *A introduction to the bootstrap* [online]. Boca Raton: Chapman, c1994 [cit. 2014-04-26]. Dostupné z: `http://getbootstrap.com/2.3.2/`

[3] The jQuery Foundation. *jQuery: kompletní průvodce vývojáře* [online]. [cit. 2014-04-14]. Dostupné z: `http://getbootstrap.com/2.3.2/`

[4] GILMORE, Jason W. *Velká kniha PHP a MySQL 5: kompendium znalostí pro začátečníky i profesionály.* Vyd. 1. [i.e. 2. vyd.]. Brno: Zoner Press, 2007, 864 s. ISBN 80-868-1553-6.

[5] CONNOLLY, Thomas. *Database Systems: A Practical Approach to Design, Implementation, and Management* [online]. 4th ed. Harlow: Addison-Wesley, 2005 [cit. 2014-03-02]. ISBN 0-321-21025-5.

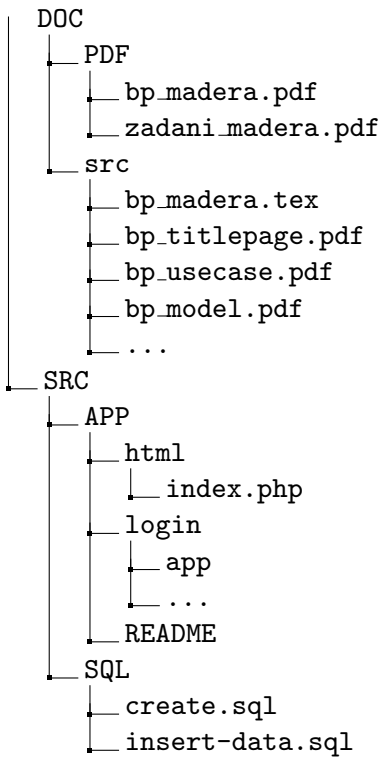
[6] SATRAPA, Pavel. Shibboleth: Identifikujte se jen jednou. [online]. 2005-12-08 [cit. 2014-04-12]. Dostupné z: `http://www.lupa.cz/clanky/shibboleth/`

[7] PHP Frameworks. BERGMANN, Sebastian a Stefan PRIEBSCH. PHP Frameworks [online]. Indianapolis, Ind.: Wiley, 2011 [cit. 2014-04-16]. Dostupné z: `http://www.phpframeworks.com/`

[8] JAKOUBĚ, Jaroslav. ORM test PHP frameworků. *Zdroják: Spouštíme letní ORM test PHP frameworků* [online]. [cit. 2014-05-01]. Dostupné z: `http://www.zdrojak.cz/clanky/letni-test-php-frameworku/`

[9] Nette Framework. NETTE FOUNDATION. *Nette Framework* [online]. 2008, 2014-01-07 [cit. 2014-05-03]. Dostupné z: `http://nette.org/cs/`

Obsah přiloženého DVD



- `DOC` – podadresář `PDF` obsahuje zadání a kompletní text BP ve formátu PDF, podadresář `src` pak LaTeXové „zdrojové kódy“ a obrázky použité v textu BP.
- `SRC` – podadresář `APP` obsahuje zdrojové kódy webové aplikce včetně souboru `README` popisující nutná nastavení před spuštěním aplikace. podadresář `SQL` obsahuje zdrojové kódy SQL dotazů pro vložení struktury databáze a výchozích dat.