Pedagogická fakulta
Faculty of Education
Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra informatiky

Back-End webové aplikace vědeckého časopisu Studia Kinanthropologica

Back-End of the web application for the scientific journal Studia Kinanthropologica

Bakalářská práce

Vypracoval: Lubomír Šimák

Vedoucí práce: PaedDr. Petr Pexa, Ph.D.

České Budějovice 2017

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 25. dubna 2017.

Lubomír Šimák

Poděkování

Rád bych poděkoval panu PaedDr. Petru Pexovi, Ph.D. za pomoc, cenné rady, ochotu a vedení této bakalářské práce. Taktéž si velice vážím podpory ze strany mé rodiny a přátel.

Abstrakt

Cílem bakalářské práce bude vytvoření serverové části (tzv. Back-end) webové aplikace vědeckého recenzovaného časopisu Studia Kinanthropologica, který je vydáván katedrou tělesné výchovy a sportu PF JU. S využitím webových technologií PHP, MySQL, AJAX, Javascript aj. bude vytvořen vlastní redakční systém, který bude sloužit nejen k administraci uživatelů, ale také pro správu, hodnocení a schvalování článků pro časopis Studia Kinanthropologica. Články budou moci psát registrovaní uživatelé s oprávněním autor, tyto články budou dále předávány k recenzi obsahu a formy vybraným uživatelům s oprávněním recenzent. Podle výsledků recenzního řízení autor článek upraví a ten bude odeslán administrátorům k finálnímu schválení.

Klíčová slova

HTML, PHP, AJAX, JavaScript, jQuery, SQL, webová aplikace.

Abstract

The aim of this bachelor thesis is to create a web application server component (ie. Back-end) of scientific journal Studia Kinanthropologica, which is published by the Department of Physical Education and Sports PF JU. Using Web technologies PHP, MySQL, AJAX, Javascript etc. will be created own content management system, which will serve not only to the administration of users, but also for the management, evaluation and approval of articles for the magazine Studio Kinanthropologica. Articles will be able to write registered users with the author authorization, these articles will be further transmitted for review the content and form to selected users with authorization reviewer. According to the results of the review process, author of article modifies this article and sends it for final approval to the administrators.

Keywords

HTML, PHP, AJAX, JavaScript, jQuery, SQL, web application.

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta pedagogická
Akademický rok: **2015/2016**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Lubomír ŠIMÁK**
Osobní číslo:	**P14230**
Studijní program:	**B7507 Specializace v pedagogice**
Studijní obor:	**Informační technologie a e-learning**
Název tématu:	**Back-End webové aplikace vědeckého časopisu Studia Kinanthropologica**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování:

Cílem bakalářské práce bude vytvoření serverové části (tzv. Back-end) webové aplikace vědeckého recenzovaného časopisu Studia Kinanthropologica, který je vydáván katedrou tělesné výchovy a sportu PF JU. S využitím webových technologií PHP, MySQL, AJAX, JavaScript aj. bude vytvořen vlastní redakční systém, který bude sloužit nejen k administraci uživatelů, ale také pro správu, hodnocení a schvalovaní článků pro časopis Studia Kinanthropologica. Články budou moci psát registrovaní uživatelé s oprávněním autor, tyto články budou dále předávány k recenzi obsahu a formy vybraným uživatelům s oprávněním recenzent. Podle výsledku recenzního řízení autor článek upraví a ten bude odeslán administrátorům k finálnímu schválení.

Rozsah grafických prací: **CD ROM**

Rozsah pracovní zprávy: **40**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. **Jak psát web [online]. [cit. 2016-04-06]. Dostupné z: http://www.jakpsatweb.cz/**
2. **W3Schools [online]. [cit. 2016-04-06]. Dostupné z: http://www.w3schools.com/**
3. **Tvorba webu [online]. [cit. 2016-04-06]. Dostupné z: http://www.tvorba-webu.cz/**
4. **PHP docs [online]. [cit. 2016-04-06]. Dostupné z: http://php.net/docs.php**
5. **Learn PHP 7, Find out What's New, and More [online]. [cit. 2016-04-06]. Dostupné z: http://www.sitepoint.com/learn-php-7-find-out-what**

Vedoucí bakalářské práce: **PaedDr. Petr Pexa, Ph.D.**
Katedra informatiky

Datum zadání bakalářské práce: **18. dubna 2016**

Termín odevzdání bakalářské práce: **30. dubna 2017**

Mgr. Michal Vančura, Ph.D.
děkan

L.S.

doc. PaedDr. Jiří Vaníček, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 18. dubna 2016

Obsah

1 Úvod

V dnešní rychlé době je již zcela nemyslitelné provádět tak komplexní operace, jako je například vedení recenzního řízení článku, například zasíláním elektronickou poštou či jinou metodou. Proto je potřeba vytvořit systémy, které toto řízení usnadní a značně urychlí. Dobře postavené webové systémy dokáží nejen urychlit práci, ale také z opravdu velké části jí zpřehlednit. Díky tomuto systému odpadá nepřehlednost ve vedení recenzního řízení a řízení článků, jako tomu je u posílání článků a posudků prostřednictvím elektronické pošty. Díky plánované administraci budou mít správci systému naprostou kontrolu nad recenzním řízením a správou jak článků, tak i uživatelů. Také budou mít k dispozici v systému informace o tom co se právě s článkem děje. Jestli je článek v recenzním řízení, čeká na schválení, či na přidělení recenzenta. Dále administrace bude sloužit i jako archiv všech článků a recenzí. Tím pádem bude mít správce k dispozici všechny potřebné informace, které se týkají recenzního řízení, a to vše na jednom místě bez nutnosti složité komunikace s ostatními členy tohoto řízení.

1.1 Cíle práce

Cílem práce je analýza požadavků katedry tělesné výchovy a sportu pedagogické fakulty Jihočeské university na vytvoření webového systému, jenž by napomáhal s recenzním řízením, schvalováním a správou napsaných článků. Na základě těchto požadavků následné vytvoření serverové části (tzv. backend) webové aplikace vědeckého časopisu Studia Kinanthropologica. Výsledný systém by měl především usnadnit a urychlit recenzní řízení na vložené články. Dále bude sloužit i pro správu uživatelů a přiřazování práv.

Při tvorbě projektu bude kladen velký důraz především na bezpečnost, rychlost a stabilitu systému. Taktéž bude systém tvořen s důrazem na jednoduchost a pochopitelnost pro všechny uživatele tohoto systému bez ohledu na věk uživatele. Z tohoto důvodu bude administrace omezena na základní funkce s možností jejich rozšíření.

1.2 Metody práce

V teoretické části práce se v první řadě zaměřím na zkoumání nejlepších metod tvorby webových aplikací, a to z hlediska bezpečnosti, ukládání dat a samotného běhu systému.

Dále se v teoretické části zaměřím na popsání vlastního počínání při analýze, návrhu, vývoji, nasazení a konečném testování systému jakožto celku, tak i jednotlivých částí systému. V praktické části práce se nejdříve zaměřím na specifikaci požadavků katedry tělesné výchovy a sportu na požadovanou webovou aplikaci. Po zpracování a ujasnění požadavků započnu analýzu pomocí myšlenkové mapy, případů užití (use case) a v případě potřeby detailnějšího popisu vnitřních procesů použiji i hlubší analytické metody, jako je například aktivity diagram.

Po dokončení analýzy, designu databáze a systému vyberu nejvhodnější technologie a postup pro vytvoření stabilní a bezpečné aplikace. Nejdříve práci započnu tvorbou navržené databáze, která se poté stane základem pro samotnou administraci. Nejdříve je důležité vytvoření databáze, jelikož celý administrační systém bude používat výhradně databázi pro ukládání dat a dále s těmito daty pracovat. Samostatné funkce administrace budu tvořit postupně a tvorbu budu prokládat testováním jednotlivých funkcí separátně na otestování funkčnosti a stabilnosti jednotlivé funkce. Po vytvoření a implementování samotného systému začnu testovat celý systém jakožto celek pro ověření funkčnosti jednotlivých částí systému ve spolupráci s ostatními částmi.

2 Specifikace požadavků

Kvůli specifikaci požadavků na plánovaný systém jsem se sešel na osobní konzultace s paní doktorkou Malátovou z katedry tělesné výchovy a sportu. Z požadavků paní doktorky vyplynulo, že požadovaný systém by měl býti dostupný ve dvou jazykových verzích, a to české a pro přístupnost uživatelům mluvícím jinými jazyky také anglické. Dále by měla být aplikace sama o sobě jednoduchá, pochopitelná a řízení průběhů v ní by mělo být přehledné i pro samotné správce celého systému.

2.1 Uživatelé

Administrace by měla dle požadavků obsahovat určité druhy uživatelských účtů, kterými jsou účty s označením Autor, administrátor a recenzent. Níže budu detailněji rozebírat a popisovat základní funkce a práva, která by měli mít tyto typy účtů k dispozici. Však s výhledem na budoucí rozšiřování systému, jsem se rozhodl udělat typy účtu ne příliš specifické k jejich určení a nikterak neomezovat možnost rozrůstání typů účtů. Proto budou práva rolím přidělována pomocí čtvrtého a jedinečného uživatele s označením superadministrátor.

2.1.1 Uživatel obecně

Každý uživatel, a to bez ohledu na typ účtu, který vlastní, by měl mít právo upravovat své osobní údaje, čímž je myšleno jméno, příjmení, pracoviště, obrázek profilu, heslo, kontakt na elektronickou poštu a samozřejmě i telefonní kontakt.

Každý neregistrovaný uživatel má právo se zaregistrovat, avšak musí sečkat na potvrzení účtu administrátorem. Díky tomuto potvrzení bude čerstvě přihlášenému uživateli předložen formulář pro odsouhlasení etického kodexu. Pokud uživatel toto prohlášení nepotvrdí nebude mu systém zpřístupněn, ale bude stále zobrazován tento formulář.

2.1.2 Autor

Uživatel s označením autor, by měl mít možnost přidávat články a před schválením článků je i editovat. Po přidání článků by měl autor mít možnost odeslat článek administrátorům s požadavkem o zahájení recenzního řízení. Jakmile recenzní řízení započne neměl by autor mít možnost manipulovat s tímto článkem. Po dokončení recenzního řízení by měl mít autor k dispozici posudky recenzentů a dle jejich znění upravit článek do vyhovující podoby. Když je článek upraven, je opět poslán do recenzního řízení a toto se opakuje dokud není článek plně schválen.

V případě, že je článek schválen, autor již nemůže s článkem manipulovat, a tedy jej mazat a upravovat. Jelikož článek je uložen do archivu pro potřeby dohledávání schválených článků, tudíž autor má právo jej pouze prohlížet, stahovat a zobrazit si jej ve svém profilu.

2.1.3 Administrátor

Účet administrátora hraje velkou roli při recenzním řízení článků. Administrátor od autorů přijímá požadavek o zahájení recenzního řízení na určitý článek a poté vybere vhodné recenzenty, kterým přidělí recenzní práci na těchto článcích. Administrátor bude vybírat pro daný článek recenzenty manuálně z důvodu zohlednění zaměření článku a odbornosti recenzenta. Dále přijímá recenze od recenzentů a dle uvážení je zasílá autorům pro potřeby úprav článků. Tento postup se opakuje do té doby, než článek odpovídá potřebám tisku. Jestliže některá z recenzí není v pořádku či je její obsah nevhodný, má administrátor právo recenzi vrátit recenzentovi.

Jestliže článek odpovídá kritériím pro schválení, je administrátorem označen jako schválený a je dále poslán do výběru potencionálních článků pro tisk. Pouze účty typu administrátor a supeadministrátor mají právo tuto sekci prohlížet a dělat v ní změny Supeadministrátor však má přístup do této sekce pouze pro potřeby technické správy, a to v případě nefunkčnosti či jiných pro-

blémů, které administrátor nemá právo nebo nedokáže vyřešit.

administrátor má dále možnost na určitou dobu pozastavit funkčnost vybraného účtu, jestliže se například u uživatele objeví nevhodné chování, či z důvodu krádeže účtu jinou osobou.

2.1.4 Recenzent

Recenzenti dostávají žádosti o recenzi na vybraný článek od administrátorů. Na základě této žádosti recenzent vypracuje posudek na tento článek ve webovém rozhraní aplikace a následně jej odešle ke zpracování zpět administrátorům. Zde může nastat problém, jestliže recenze nevyhovuje, musí jí recenzent opravit tak aby bylo možné jí poslat autorům k nezbytné opravě článku.

2.1.5 Superadministrátor

Tento typ účtu je jedinečný a bude mít k němu přístup pouze jeden jediný uživatel. Superadministrátor bude mít kontrolu nad účty a jejich právy po stránce přidělování a odebírání funkcí pro vybraný typ účtu. Tím pádem bude mít možnosti v určité míře přizpůsobit systém dle aktuálních požadavků. Samozřejmě nebude mít práva všech ostatních typů účtů z důvodu, že by se jednalo o správce z řad IT pracovníků, a tudíž by bylo zcela nevyhovující, aby tento uživatel měl přístup do ostatních modulů. Jestliže je však toto nutností, je možné uživateli přidělit role s příslušnými moduly.

2.2 Články

Jelikož tato aplikace má býti zaměřená na recenzní řízení článků, jenž po schválení poputují do tisku, jsou články stěžejní částí celého připravovaného systému. Z důvodů bezpečnosti budou články dostupné pouze určeným osobám a rolím spolu s autorem, jenž článek uložil do databáze spolu s uvedenými spoluautory. Po nahrání bude článek možné odeslat administrátorům s požadavkem na vytvoření recenzního řízení.

2.2.1 Vkládání článků

U vkládání článků je nejdříve požadováno odsouhlasení, že práce již nebyla v minulosti nikde publikována, a následně umožnění vložení práce ve Word dokumentu či formátu PDF. Dále je však požadováno, aby byl článek posílán do recenzního řízení již v PDF formě, tudíž se bude muset nejspíše řešit převedení vkládaného článku z Word dokumentu do dokumentu PDF.

Při vkládání článku do systému bude po autorovi požadován výběr souboru obsahující text článku, název článku, abstrakt, klíčová slova, jež charakterizují článek a dále má autor možnost zadat spoluautory, jež jsou zaregistrovaní do administrace jakožto autoři. Spoluautoři budou mít samozřejmě omezené pravomoci v manipulaci s články. Volba spoluautorů je pouze volitelná a autor může uložit článek bez jejího vyplnění.

2.2.2 Úprava článků

Úpravy článků může provádět pouze autor, jenž článek do systému vložil nebo spoluautoři článku. Článek půjde upravovat pouze v tehdy, kdy je článek vrácen k úpravám, nebo ještě nebylo započato recenzní řízení článku. V takovém případě je jakákoli manipulace s článkem pro autora nepřístupná, aby nedocházelo k úpravám textu v průběhu vytváření recenzí.

2.2.3 Zobrazení článků

Články budou moci vidět samozřejmě autoři a spoluautoři článků, avšak také budou přístupné administrátorům a určeným recenzentům článků. Dále dle statusu článku budu mít autoři, spoluautoři, administrátoři a recenzenti možnost si článek stáhnout do svého počítače, pro potřeby kontroly, či recenze textu.

2.3 Recenze

Recenze budou psány přímo v systémovém prostředí na webu. Zde bude pro tyto potřeby přístupný WYSIWYG editor, kde recenzenti budou moci sepsat své recenze a dále je uložit na server v textové podobě. Recenze se budou načítat autorům přímo v systému pod kolonkou „upozornění“, či pod tlačítkem „nedávná aktivita“. Zde se již jedná o grafické zpracování elementu, které přenechávám kolegovi, jenž má na starosti front-end celého projektu.

3 Výběr technologií

Po specifikaci a analýze požadavků katedry tělesné výchovy a sportu, jsem se dále zaměřil na průzkum dostupných technologií, s nimiž by bylo možné požadovaný systém vytvořit. Postupně jsem vybíral technologie, které použiji a které ne. Jelikož se jedná o systém, který má v praxi stále a spolehlivě fungovat, vybíral jsem převážně mnou známé technologie.

3.1 Databáze

Dnes existuje mnoho různých cest, jak přistupovat k řízení dat, které chceme ukládat a spravovat. Každý programátor vyvíjející určitý systém si musí nejdříve odpovědět na několik otázek. Na otázky, jako jsou například: „Jakou aplikaci budu stavět? Kolik uživatelů bude aplikaci používat? Jak rozsáhlou databázi potřebuji pro ukládání všech dat?“ Na základě odpovědí na tyto otázky může dojít k výběru daného typu databáze a k výběru přístupu k datům, které budou uloženy v databázi. Zde máme na výběr hned z několika technologií databází. Pro náš systém jsem si zvolil relační, přesněji MySQL databázi.

3.1.1 Relační databáze

Tento typ databáze se sestává z tabulek, které obsahují dvou rozměrné pole sestávající se z řádků a sloupců. Sloupce jsou označovány, jako atributy tabulky což znamená, že u tabulky uživatel by například do atributů patřilo jméno, věk a tak dále. Jednotlivé řádky jsou jednotlivé záznamy v tabulce.

V tabulkách vznikají takzvané klíče. Jako klíč je označován konkrétní atribut, nebo skupina atributů. Zde se setkáváme hned s několika typy klíčů. Tyto klíče jsou označovány jako kandidátní klíč, primární klíč, alternativní klíč a cizí klíč.

Kandidátní klíč – Kandidátní klíč může být atribut, či skupina atributů, které jednoznačně identifikují určitý záznam v tabulce. Tento klíč se nazývá jako kandidátní, jelikož se může stát primárním klíčem či alternativním klíčem, a to v případě, že není vybrán za klíč primární.

Primární klíč – Primární klíč jednoznačně určuje daný zápis v tabulce, z čehož vyplývá, že primárním klíčem může být pouze jeden atribut, či více atributů dohromady, což je označováno jako složený klíč. Pro každý záznam v tabulce musí být primární klíč unikátní, který se v dnešní době vytváří převážně pomocí „úmělých“ klíčů, jako jsou například číselné hodnoty, které se z pravidla automaticky zvyšují s dalším záznamem.

Alternativní klíč – Alternativní klíče jsou v podstatě takové klíče, které by mohli sloužit jakožto primární klíče, avšak nebyly za primární klíč vybrány.

Cizí klíč – Tento typ klíče je velice důležitý pro zaznamenání vztahů, takzvaných relací, mezi jednotlivými tabulkami. Jestliže má být některá z tabulek v relaci s druhou tabulkou, což znamená, že údaje obsahující první tabulku souvisí s údaji v druhé tabulce, musí tato první tabulka obsahovat cizí klíč, který je v podstatě primární klíč druhé tabulky.

[1]

3.1.2 ORM

ORM neboli objektově-relační mapování. Nejde ani tak o nový druh databáze, nýbrž pouze o objektový přístup k relační databázi. V podstatě ORM je jakousi mezi vrstvou mezi systémem a databází.[2] O účinnosti ORM, se polemizuje v souvislosti s používání v rozsáhlejších systémech. To z hlediska zpomalování systému, jelikož se jedná o mapující mezi vrstvu je samozřejmostí, že i toto mapování si žádá určitou režii, což u větších systémů může být znatelné. Taktéž z hlediska určité „svázanosti“ programátora.

Tím to jsou myšlené situace při dotazování na databázi, kdy jsou potřeba složitější dotazy a zde je někdy autor systému nucen obcházet ORM klasickým SQL dotazem. Tudíž pak dochází k zanášení kódu dalšími SQL dotazy ve snaze dostat z databáze potřebné informace, které díky ORM nemusíme být schopni dostat.

3.2 Systém

Při hledání technologií pro vytvoření systému, jsem se přiklonil k objektově orientovanému programování v jazyce PHP s využitím architektury MVC. Dále budu využívat Javascriptu a AJAXU pro nezbytné operace, mimo PHP či pro asynchroní načítání obsahu stránek.

3.2.1 PHP

Zkratkou PHP je označován objektově orientovaný, skriptovací, programovací jazyk, který je nejvíce používán na vytváření dynamických webových stránek a webových aplikací, jakožto bude výsledný projekt této bakalářské práce. Skripty PHP se na rozdíl od HTML provádějí na straně serveru, kdežto HTML se provádí na straně uživatele. K přenášení dat mezi uživatelem a serverem je nejčastěji používáno http dotazovacích metod. Proč zrovna PHP?

- Přenositelnost mezi platformami (Linux, Windows, Mac OS X a další) bez nutnosti zasahování do kódu
- PHP podporuje širokou škálu databází
- Je dostupné k používání zcela zdarma
- Je kompatibilní s téměř se všemi servery, jež se dnes používají (Apache, IIS a další)
- PHP se dá vcelku snadno naučit a běží na serverech efektivně

- PHP disponuje existencí velké škály knihoven pro různé účely. Například knihovny pro zpracování textu, grafiky, souborů, přístup k systémům databáze a jiné

[3]

3.2.2 MVC

MVC neboli architektura založená na modelu (Model), pohledu (View) a kontroleru (Controler). Tuto architekturu je dobré používat, jelikož je založena na principu nezávislosti jednotlivých částí aplikace mezi sebou. MVC funguje tak, že model má funkce převážně spojené s databází, jako je například vkládání informací do databáze, jejich selektování či úpravy. View neboli pohled má za úkol všechno potřebné uživateli předložit ve srozumitelné podobě, neboli vyrenderovat převzaté údaje společně s předpřipravenou šablonou do podoby klasické webové stránky. Mezi tímto vším se nachází kontroler, jenž všechny operace týkající se modelu a pohledu řídí. Například jestliže je třeba provést výpis uživatelů pro jejich správu, tak si zavoláme příslušnou metodu kontroleru, která nám zavolá příslušný model a jeho metodu, jenž nám vrátí požadované údaje z databáze. Zde tyto informace kontroler poskytne příslušnému pohledu. Ten nám již sestaví stránku do grafické podoby a pomocí metody vykreslování nám jí zobrazí. Z tohoto vyplývá důležitá zásada MVC, a to že model a view by neměli mezi sebou komunikovat na přímo, nýbrž přes kontroler, který zajišťuje veškerou komunikaci.[4]

Volání kontrolerů provádíme skrze URL adresu. Nejpřehlednějším a nejpřívětivějším způsobem se jeví pomocí speciálního `.htaccess` souboru nastavit URL adresu tak, aby místo:

`https://adresa.cz/?action=kontroler&method=medotaKontroleru&parametr1=param1&parametr2=param2...`

Vypadala takto:

`https://adresa.cz/kontroler/metodaKontroleru/param1/param2/...`

Z uvedeného příkladu je zcela jasné, jak tato cesta nám velice usnadní volání samotných kontrolerů a jejich metod. [4]

3.2.3 SQL

Jedná se o standardizovaný dotazovací jazyk, který se používá pro přístup k datům v relačních databázích a jejich manipulaci. Taktéž je možné tímto způsobem manipulovat jak se samotnými tabulkami, tak také s celou databází obsahující tabulky. Tudíž pomocí SQL může programátor vytvářet, měnit, mazat jak data, tak samotné tabulky či celé databáze.[5]

3.2.4 Javascript

Jedná se o multiplatformní, objektově orientovaný, skriptovací jazyk, který se dnes používá zejména k tvorbě webových stránek. Na webových stránkách je používán na ovládání, animaci a efekty HTML prvků, jako jsou například tlačítka, vkládací políčka a další elementy stránky. [6]

Javascript se spouští na klientské straně po stažení webových stránek na rozdíl třeba od PHP kde se provádí kód na straně serveru a posléze je stránka zasílána uživateli. Bohužel díky této vlastnosti vznikají jistá bezpečnostní rizika, která musí programátor řešit.[7]

Javascript není typický objektově orientovaný jazyk, jelikož se zde nepracuje přímo s třídami a instancemi. Toto je však v Javascriptu nahrazeno prototypováním, čímž může simulovat většinu mechanismů, které se používají u jazyků používajících třídy jako je například dědičnost. Zde nám funkce nahrazují konstruktory objektů, kde je potřeba používat pro parametry klíčové slovo „this“. Taktéž funkci metod zde přebírají funkce již se zmíněným slovíčkem „this“, a to přímo v konstruktoru objektu, nebo je možné metody deklarovat jako prototyp s použitím názvu konstruktoru, ke kterému chceme prototyp přidružit, přidáním klíčového slova „prototype“ a na konec námi požadovaný název metody.[8]

3.2.5 AJAX

AJAX je akronym pro *„Asynchronous Javascript and XML“[8].* Tímto je v informatice označována technologie pro asynchronní načítání obsahu některé části webové stránky bez nutnosti znovu načítání celé stránky. Toto řešení ulehčí nejen serveru a šetří přenášená data, ale taktéž je uživatelsky přívětivější. Například jestliže je uživatel někde v půlce dlouhé stránky a chce něco změnit bez použití technologie AJAX, tak je třeba načíst celou stránku znova.

Bohužel použití technologie AJAX se neobejde bez několika problémů. Například problém v podobě internetové latece, kdy při zpoždění uživatel vidí pouze zpomalenou odezvu a může si myslet, že požadavek na zpracování nebyl vyslán. Toto může vést uživatele k tomu, aby požadavek vyslal vícekrát, čímž pouze zvýší zátěž. Proto je toto třeba řešit naznačením zpracovávání požadavku znakem, či animací průběhu načítání.

Dále se pak může projevit problém u některých prohlížečů při neuložení aktualizované stránky skrze AJAX do historie prohlížeče. Tudíž v případě, kdy uživatel použije tlačítko zpět, tak bude navrácen zpět na stránku, která nebyla načítána skrze AJAX. [10]

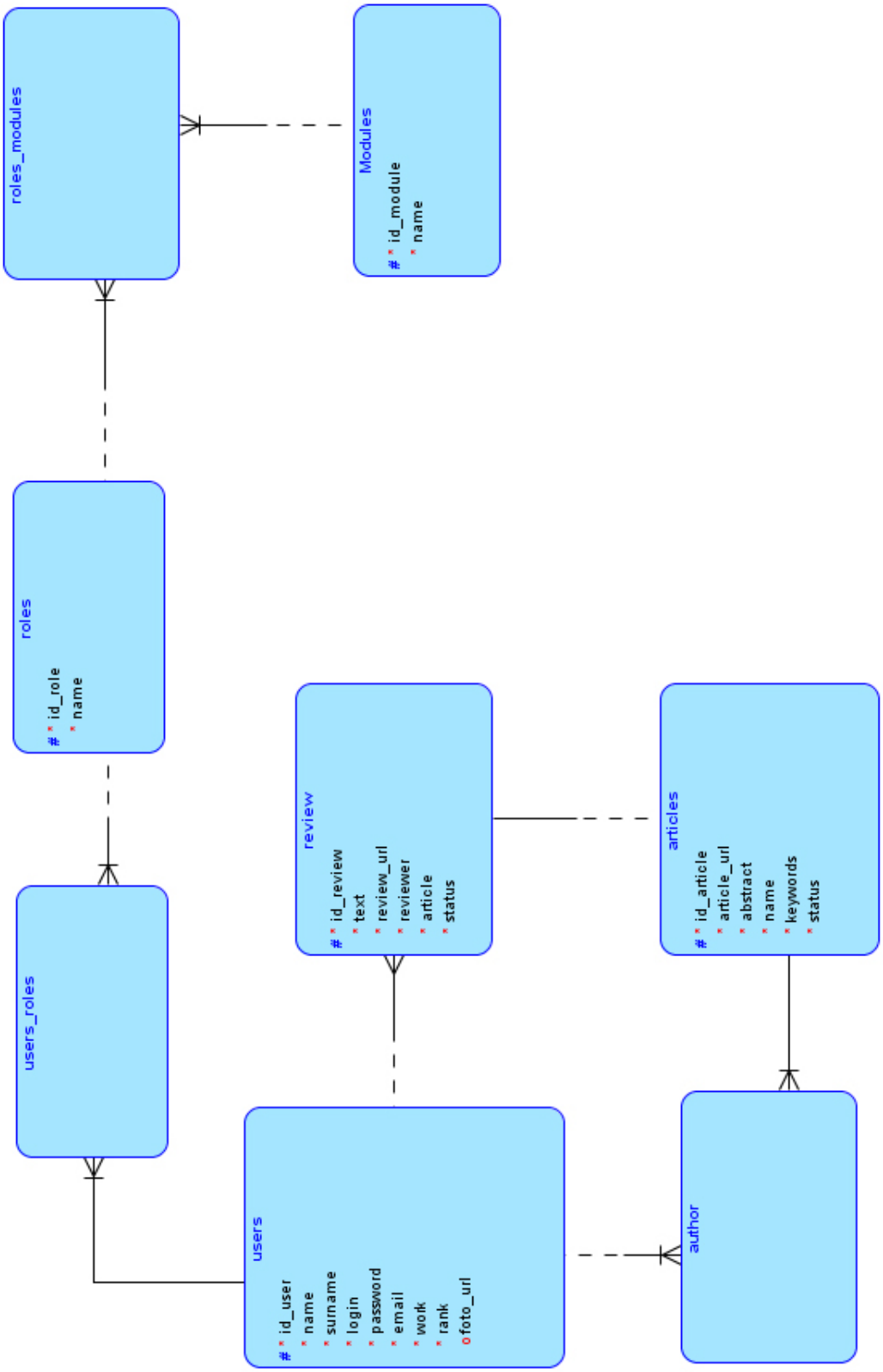
4 Návrh

Před samotným návrhem systému, bylo nezbytné udělat průzkum a projít si různé metody řešení. Dále jsem vybíral adekvátní řešení určitých částí systému, tak aby byly splněny podmínky pro stabilitu, bezpečnost a rychlost systému. Následně budu navrhovat jednotlivé části projektu pomocí grafického modelu databáze, diagramu případu užití pro vyjádření rozsáhlosti a funkčnosti systému jakožto celku a podrobnějších diagramů, jako je například stavový diagram, aktivity diagram pro potřebu detailnějšího popisu jednotlivých průběhů v systému.

4.1 Databáze

Model Relační databáze jsem si vypracoval v programu DataModeler. Návrh jsem vytvořil ve třetí normalizované formě. U některých tabulek, jako jsou například uživatelé a články, bylo nezbytné pro dodržení třetí normálové formy vytvořit relační tabulku „Authors“, která obsahuje dva cizí klíče. Primární klíče z tabulek „Users“ a „Articles“.

Návrh databáze byl zpracován tak, aby pro každého uživatele mohla být přiřazena jedna či více rolí a tyto role dále vytvářet. Rolím dále přiřazovat, nebo odebírat dostupné funkce. Taktéž v případě rozšíření funkčnosti systému, je možné díky tomuto návrhu zajistit rychle, bezpečně a bez problémů. Toto je zajištěno díky tabulkám modulů, rolí a relační tabulce, která tyto tabulky propojuje.



Obrázek 1: Diagram databáze

4.2 Bezpečnostní rizika

Jelikož v podobných systémech dochází ke komunikaci aplikace s databází za pomoci SQL dotazování, může se zde objevit několik bezpečnostních hrozeb. Jelikož je možné skrze SQL dotazy získat důležité, nebo důvěrné informace o uživatelích systému, je nutné mít tyto rizika zmapované a všemi dostupnými prostředky se jim snažit zabránit.

4.2.1 SQL Injection

SQL Injektion představuje hrozbu v té části, kdy nám aplikace přebírá vstupní data od uživatelů. Především se jedná o různé formuláře, jako je například formulář pro přihlášení do systému či formulář pro registraci a jiné vstupní pole. Jelikož uživatel znalý SQL dotazování je schopen tyto vstupní políčka využít a s jejich pomocí přetransformovat SQL dotaz tak, aby dostal informace ke kterým by vůbec neměl mít přístup.

„SQL injektion is one of the most devastating vulnerabilities that impact a business, as it can lead to exposure of all of the sensitive information stored in an application's database, including handy information such as usernames, passwords, names, addresses, phone numbers, and credit card details“ [11]

SQL injection funguje pouze tehdy, jestliže je programátor neopatrný a zapisuje dotazy v podobě, jako je například dotaz:

```
$dot = ,,SELECT * FROM users WHERE login=‘‘‘.$userLogin.‘‘‘‘;
```

Při takovémto zápisu dotazu nám hned vzniká problém, jelikož útočníkovi stačí zadat do příslušné kolonky místo loginu například:

```
Něco‘ OR ,1‘=‘1
```

A z toho nám vznikne takto transformovaný dotaz:

```
$dot = ,,SELECT * FROM users WHERE login=‘Něco‘ OR ,1‘=‘1‘“;
```

Takto upravený dotaz získá požadované informace i v případě, že není zadán platný login od uživatele. Toto je možné díky podstrčenému operátoru OR jenž říká, že musí být splněná alespoň jedna z podmínek a to buď správný login, nebo podmínka 1=1, která platí vždy a tudíž útočník získá požadované informace. [11]

Obrana proti SQL injection není nikterak složitá. Programátorovi pro zabezpečení vstupu postačí pouze aby ošetřil různými postupy vstupní data. Při Hledání nejlepší cesty, jak zabezpečit vstupy od uživatele, jsem narazil na rozdílné názory. Některé zdroje tvrdí, že bohatě postačí zabezpečení vstupů pomocí PHP metody `htmlspecialchars()`, jejíž funkce spočívá v konvertování speciálních znaků, jako jsou například apostrov, uvozovky, ampersand či znaky „<“ a „>“, do HTML entit. Například zápis znaménka ampersand &, nám tato funkce konvertuje do podoby `&`

Já se však se přikláním k tvrzení, že je dobré používat PHP rozhraní PDO. Toto rozhraní je postavené pro pohodlnější komunikaci s databází a taktéž díky funkci předpřipravených dotazů `prepare()` napomáhá zabránit provedení SQL Injection. Ve funkci `prepare()` si zapíšeme celý dotaz až na vkládané hodnoty z PHP. Tyto hodnoty nahradíme buďto otazníky a nebo speciálním zápisem proměnné, kde na jejím začátku se nachází dvojtečka například: „:proměnná“. Tato speciální proměnná se nazývá placeholder. Po předpřipravení dotazu dále použijeme metodu pro vykonání dotazu. tato metoda se nazývá `execute()`. V případě otazníku si při použití metody `execute()` musíme dávat pozor na pořadí hodnot, které chceme vkládat, jelikož první hodnota je přiřazena prvnímu otazníku, druhá hodnota druhému a tak dále. Však při použití speciálních placeholderů je naprosto jasné, jakému placeholderu přiřadíme, jakou hodnotu, a to bez ohledu na pořadí hodnot. [12]

Já se rozhodl používat obě metody zabezpečení zároveň. Tudíž použiji , jak metodu `htmlspecialchars()`, tak rozhraní PDO, jelikož takovéto počínání ničemu neublíží, spíše naopak nám to může pomoci i u napadení typu cross-site scripting o němž se budu níže více rozepisovat.

4.2.2 Cross-site scripting

Neboli XSS je typ útoku, kde se útočník snaží nahradit kus HTML stránky svým vlastním kódem. Toto používají útočníci pro nahrání Javascript souborů z externího serveru a odtud také pochází název Cross-site scripting. Útočník záměny docílí tak, že svůj vlastní kód zapíše do proměnné GET v URL adrese stránky. Tímto způsobem, může nejen připojit svůj externí script, což je nejzávažnější, ale taktéž měnit vzhled stránky. Díky této metodě se útočník může dostat k důvěrným informacím o uživatelském účtu či ke zneužitelným informacím uživatelů. Této metodě, se dá účinně bránit pomocí funkce `htmlspecialchar()`, jejíž funkcionalitu jsem již popisoval v podkapitole 4.2.1. [13]

4.2.3 Dictionary attack

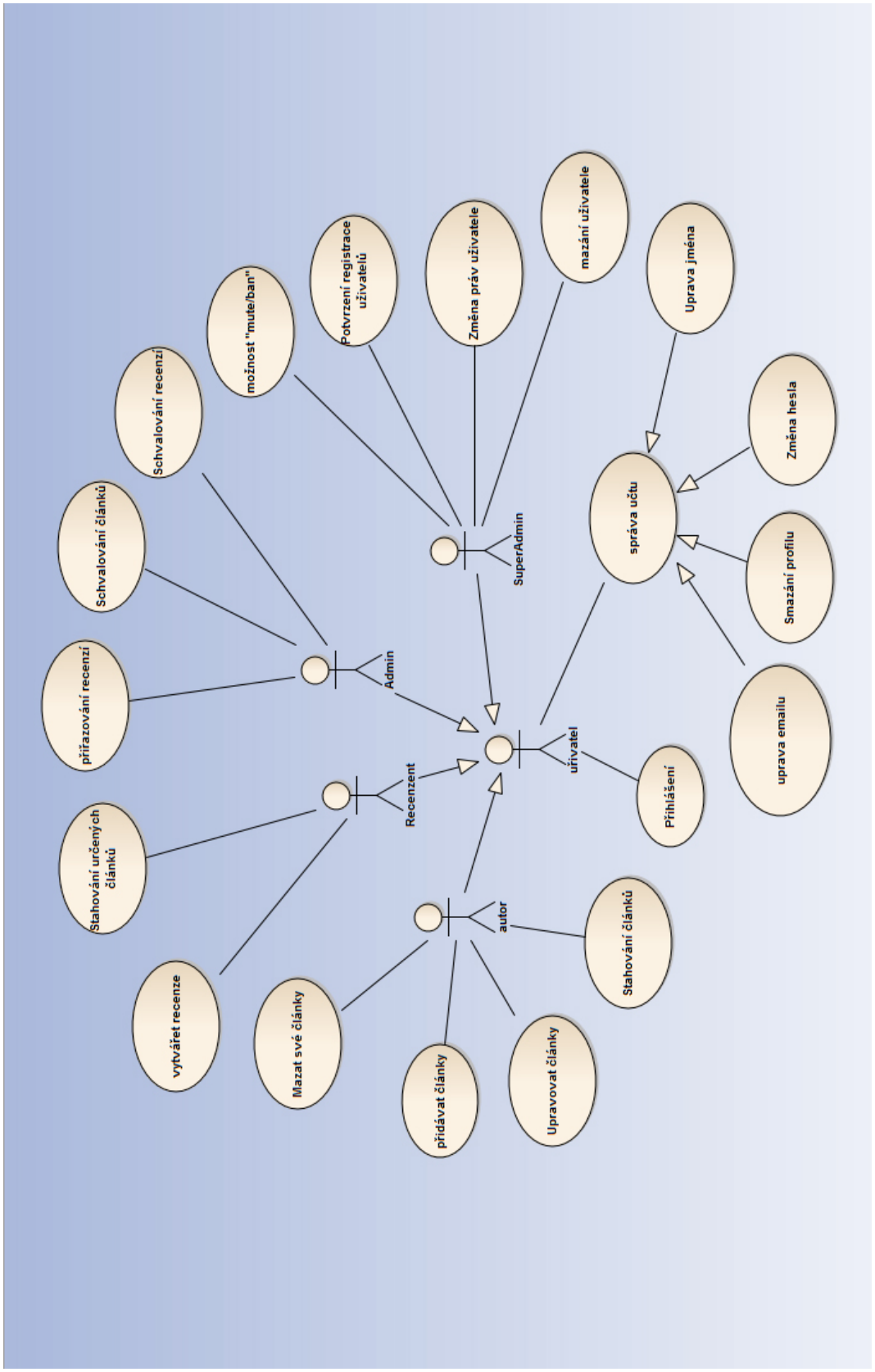
Jelikož hesla jsou většinou ukládaná do databáze v zaheshované, neboli zakódované podobě pomocí jednosměrné hashovací funkce a nejde výsledný hash dekódovat do původní podoby. Existuje takzvaný slovníkový útok. Toto je metoda, při níž útočník využívá velmi rozsáhlé databáze předpokládaných hesel a jejich zaheshované hodnoty. Tudíž zkouší heshe ze slovníku a porovnáváním těchto hashů se snaží za pomocí hrubé síly přijít na heslo.[14]

Proti tomuto útoku je možné se z hlediska tvůrce programu bránit takzvaným „solením hesla“. Solení hesla spočívá v tom, že programátor k heslu uživatele přiřadí znaky dle svého výběru a poté heslo zaheshuje. V tomto případě pro získání hesla musí útočník znát i řetězec použití pro solení hesel.

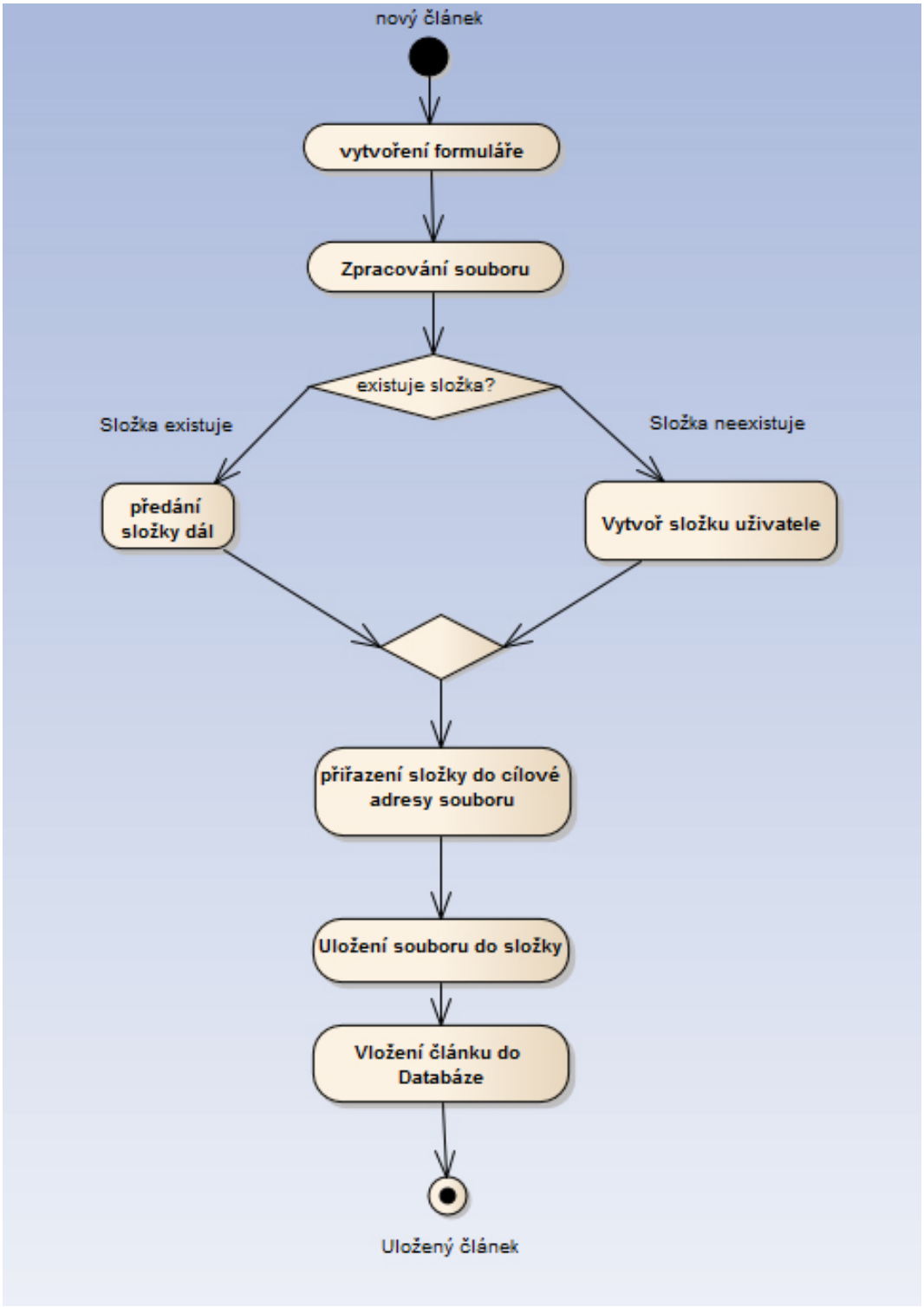
4.3 Systém

Nejdříve jsem vytvořil myšlenkovou mapu za pomoci programu FreeMind. Tato myšlenková mapa měla za úkol nastínit, jak by měl systém fungovat z hlediska rozdělení funkcí na typy požadovaných účtů a na stranu automatických úkonů na serverové straně. Po dokončení myšlenkové mapy jsem pokračoval v programu Enterprise Architekt vytvořením diagramu případů užití. Některé z jednotlivých případů užití byly rozpracovány v diagramu aktivit pro detailnější popis procesu.

Funkčnost systému, jsem zamýšlel tak, aby bylo možné systém rozšířit a následně jednoduše přidělit novou funkčnost jednotlivým rolím. Pro tento případ jsem navrhl existenci třetí role, jenž bude dostupná pouze jednomu uživateli a tuto roli jsem pojmenoval superadministrátor. Dostupné funkce pro tuto roli jsou již popsané v kapitole 2.1.5.



Obrázek 2: Use case diagram systému



Obrázek 3: Activity diagram přidávání článku

5 Implementace

Jakmile byla hotová analýza a návrh, začal jsem své návrhy implementovat. Nejdříve jsme si s kolegou Markem Musilem zvolili hosting pro vývoj a testování aplikace, která po svém dokončení bude nahraná na požadovaný hosting. Před samotnou prací na systému bylo nutné vytvořit databázi, do které by se ukládali všechny potřebné informace pro chod systému. Nakonec bylo možné na těchto základech vytvářet systém.

5.1 Databáze

Databáze byla vypracovaná dle návrhu v online rozhraní PHPMyAdmin na hostingu pro vývoj systému. Zde byla vytvořena prvotní verze databáze, jejíž diagram můžete vidět na obrázku 1. Prvotní databáze se od finální verze příliš nelišila. Ve finální verzi přibylo pouze pár nových řádek, jako je třeba řádek „mainAuthor“ do tabulky „author“. Tento řádek vznikl z nutnosti rozlišovat, zda se jedná o hlavního autora, jenž vkládal článek, nebo o spoluautora článku. Potřeba této úpravy vyvstala až při práci na samotném systému, a to z důvodu nejlepšího doložení uživatele, který tento článek nahrál do systému.

Tudíž v některých případech byla databáze upravována již při práci na systému, a to na základě objevení nových poznatků z hlediska možné funkčnosti systému.

5.2 Zabezpečení

Jelikož systém komunikuje s databází, do systému se uživatelé registrují, přihlašují, vkládají informace, nahrávají soubory, byl systém zabezpečen hned z několika hledisek.

5.2.1 Registrace

Registraci uživatele je nutné ošetřit proti nechtěné registraci robotů jež by nám mohli zahlcovat databázi. Pro rozeznání, zda jde o fyzického uživatele, či robota, existuje mnoho cest. Od těch nejjednodušších přístupů, jako je například sečtení dvou čísel, přes časová razítka, honeypoty (zjednodušeně Javascriptem skryté inputy, které uživatel nevyplní, ale robot ano)[15], až po ty nejsložitější jako je například nová reCAPTCHA. Jak je známo CAPTCHA, jako taková, kde se uživateli zobrazil obrázek s deformovaným textem a uživatel byl nucen tento text správně přepsat, je již prolomena a navíc je tato metoda ochrany pro uživatele velice nepohodlná, nepříliš přátelská a někdy až nepřekonatelná ochrana pro uživatele samotného. Proto bylo nutné přijít s jiným typem ochrany. Při hledání bylo nutné zvážit míru ochrany stránky a z části taktéž míru pohodlí uživatele.

Jakožto dobrou volbou, která balancuje na hranici těchto požadavků, se jevila nová reCAPTCHA z dílny Google. Tato ochrana proti robotům spočívá v několika ohledech jakožto je chování uživatele na stránce z hlediska pohybu kurzoru, scrollování, zoom. Dále je porovnává s údaji ve vlastní databázi a snaží se zjistit, zda jste člověk či ne, respektive jestli počínání uživatele se podobá více člověku, či robotu. Dále taktéž zaznamenává údaje o prohlížeči, ze kterého uživatel či robot přichází. Nová reCAPTCHA pracuje ve dvou režimech, a to v režimu se zaškrtávacím políčkem nejsem robot, nebo v režimu neviditelné validace, jenž uživatele neobtěžuje ani zaškrtávacím políčkem. [16]

Však i tato ochrana není stoprocentní a v případě, že si není reCAPTCHA zcela jistá, zda se jedná o fyzického uživatele, je uživatel nucen projít

poznáváním obrázků, či vybíráním jeho částí, kde se nachází určitý předmět. Pokud nastane tato situace je pak nová reCAPTCHA velice nekomfortní pro uživatele. Bohužel pro uživatele dotykových zařízení nastane tato situace vždy.

Jelikož dnešní roboti již dokáží obejít časová razítka, honeypoty taktéž nepřináší dostatečně velkou míru zabezpečení při cíleném útoku na webovou aplikaci nejsou tyto metody příliš dobrou volbou. Dále zde máme kontroly typu „Sweet Captcha“ a „PlayThru“ kde je uživatel nucen hrát minihry, aby prokázal, že není robot. Nehledě na míru obtěžování těmito kontroly je možné, že starší generace uživatelů by mohla mít s tímto typem kontroly problémy a tudíž tento typ kontroly taktéž nepřichází v úvahu.

Ze získaných poznatků jsem došel k názoru, že i přes své značné nevýhody se zdá nová reCAPTCHA invisible, jako nejlepší variantou filtrování robotů pro tento typ webové aplikace, a to z hlediska poměru bezpečnosti a uživatelské přívětivosti jenž nabízí.

5.2.2 SQL Injection

Tomuto útoku předcházím v první řadě pomocí funkce `htmlspeacialchars()` a v druhé řadě pomocí metody PDO rozhraní, kterou je metoda `prepare()`. V podstatě se jedná o předpřipravení dotazu s placeholdery, které se dodají do dotazu později s pomocí PDO metod `bindParam()`, `bindValues()`, nebo `execute()` [12]. Díky tomu, že se data od uživatele nedosazují přímo do SQL dotazu, se zabráni útočníkovi provést SQL injection. Jak toto přesně funguje již probírám v kapitole 4.2.1.

5.2.3 Vkládání souborů

Do systému je povoleno uživateli nahrávat soubory s články a obrázkové soubory pro potřeby profilové fotky uživatele. U vkládání souborů článků jsem řešil správnost souboru dle přípony, velikosti a ochranou před případným „swapováním“ přípony. Přípona soboru se pak porovnávala s hodnotami obsaženými v poli povolených přípon.

Takto jsem řešil i porovnávání nahrávaných obrázkových souborů, s tím rozdílem, že když nám projde podmínkou přípona souboru, tak zde také kontroluji o jaký obrázek se jedná a zda se vůbec jedná o obrázek. Toto provádím pomocí metody `getimagesize()`. Jelikož nám tato metoda v případě, že se jedná o obrázek, vrátí i informace o typu obrázku v číselné podobě [16].

5.3 Systém

Po vytvoření databáze započali práce na samotném systému. Nejdříve bylo nutné vytvořit speciální .htaccess soubor a knihovny, které zajištují spolu dohromady chod systému v MVC struktuře. Pro tento účel je důležitou knihovnou soubor Boot.php. Tato třída přebírá url adresu a poté si tuto adresu rozdělí dle lomítek na volané kontrolery, jejich metody a argumenty volaných metod. Po zavolání kontroleru tato třída zjistí, zda požadovaný kontroler existuje. Jestliže požadovaný kontroler neexistuje, tato třída přilinkuje kontroler chyby a poté jí zobrazí. V případě, že požadovaný kontroler existuje, přilinkuje se soubor onoho kontroleru, a poté se provedou požadované operace. Další knihovnou je soubor Controller.php. Tato třída je rodičovskou třídou všech kontrolerů a jakožto taková zajištuje vytvoření pohledu pro kontroler, neboli view a takté zajištuje načítání požadovaného modelu pro určený kontroler.

Knihovnu view jsem vytvořil, aby sloužila jako rodičovská třída pro pohledy všech kontrolerů a pro tyto účely obsahuje metodu `lang()`, která slouží pro načítání definovaných jazykových slovníků. Metoda `lang()` se volá rovnou v konstruktoru třídy a také za určitých podmínek v metodě render, kde

je toto důležité pro renderování pohledů skrze AJAX bez hlavičky do různě se otevírajících oken, jako jsou například modální okna, nebo vyjížděcí okna funkcí.

Dále pak knihovna Model, která je rodičovskou knihovnou pro všechny modely kontrolerů. Obsahuje metodu `securityHash()` zajištující zakódování hesel při komunikaci s databází. Přímo v konstruktoru třídy „Model“ je vytvořen objekt databáze z knihovny Database.php, jenž je rozšíření pro rozhraní PDO a dále je děděn ostatními modely.

U komunikace s databází za pomoci rozhraní PDO jsem narazil na jeden problém. Pro zabránění výpisu všech položek z tabulky najednou, bylo nutné používat SQL dotaz jenž obsahuje LIMIT. Aby byl systém co nejjednodušeji nastaven, jsem zřídil proměnou třídy, kterou později pomocí PDO metody `execute()` dosazuji do dotazu.

```
$state->execute(array(':limit' => $this->limit))
```

Zde však nastává problém, jelikož takto dosadíme proměnnou do dotazu, jako textový řetězec a SQL příkaz LIMIT očekává hodnotu typu integer. Proto nám takto provedený příkaz execute nic nevrátí. Pro tyto účely je vhodné použití metody `bindParam()` takto:

```
$state->bindParam(':limit ', $this->limit, PDO::PARAM_INT);
$state->execute();
```

Po vytvoření všech knihoven pro základní funkcionalitu MVC architektury, jsem si vytvořil soubor pro konfiguraci připojení k databázi, soubor pro definování konstant a dále jsme začal do příslušných složek vytvářet kontrolery, modely a pohledy pro části systému. Dále v textu budu popisovat jednotlivé funkce a problémy jež jsem v nich řešil.

5.3.1 Články

Při vkládání článků je nejdříve nutná validace správnosti všech vstupních dat, které uživatel zadává. Po nahrání souboru a vyplnění všech potřebných informací se pomocí technologie AJAX pošlou informace do příslušného PHP souboru, zde u článků article.php, jenž plní funkci kontroleru dané sekce. V tomto kontroleru se předají data příslušné metodě a ta provede validaci dat zaslaných z formuláře. Pokud je zde vše v pořádku tak se zavolá metoda, která zjistí zda se jedná o požadovaný formát souboru či ne. Toto se určuje dle velikosti souboru a přípony souboru. Zde se kontroluje, zda nemá soubor podvrženou příponu pomocí swapování přípony, a to za pomoci sečtení velikosti pole po rozparsování názvu souboru dle znaménka tečky, či jestli pole neobsahuje nepovolené přípony jakožto .js, .php a jiné.

Z jednoho požadavku na vkládání článků vyplývá nutnost převádět články z word do PDF dokumentu. V této oblasti jsem našel dosti takových služeb, které však jsou poskytovány třetí stranou. Tudíž v rámci zabezpečení toto řešení nedoporučuji. Dále jsem testoval jiné možnosti řešení, jako například mPDF, však v tomto řešení docházelo k nepřesnostem ve formátování textu. Z těchto důvodů by bylo nejlepší vytvořit vlastní konvertovací script, však to již přesahuje rámce této práce, tudíž toto doporučuji jako pozdější rozšíření systému a prozatím nahrávat pouze PDF soubory.

Samotné vkládání článků jsem realizoval tak, že po přijetí dat a validaci jak formuláře, tak vybraného souboru, se nejdříve zjistí zda ve společné složce „articles“ již existuje složka uživatele. Pokud složka uživatele neexistuje, tak se vytvoří ze spojení ID čísla uživatele a příjmení uživatele bez diakritiky. Poté je do existující složky soubor nahrán a související informace s článkem jsou uloženy do databáze.

Při tvorbě funkcionality ohledně operací s články po uložení do databáze, jsem narazil na nutnost rozlišit v jakém stádiu se článek zrovna nachází. Z počátku status článku měl pouze indikovat, jestli je článek schválen k tisku, či ne.

Později se však ukázalo, že je nutné rozlišovat status článku v širších souvislostech, jako například jestli je článek v recenzním řízením, jeli recenzní řízení dokončeno, či zda je článek upravován. Nutnost rozlišit v jakém konkrétním stádiu se článek nachází vyvstala z potřeby úpravy článků autorem a spoluautory článku. Tudíž v třídě „Article“ jsem si definoval proměnné s číselnými hodnotami označujícími rozdílné stavy. Tyto stavy a jejich významy budu ve vztahu k základním rolím, jež může uživatel v systému mít, popisovat níže v tabulce 1.

Status	Stav	Autor	Admin	Recenzent
0	Upravování	(U, S, T)UA	–	–
1	Vytvořený článek	U, S, T, OA	–	–
2	Článek pro adminy	U, T	VR, T, S	–
3	Článek v recenzi	T	ZR, T	ZR, T
60	Článek přijat	T	T	–

Tabulka 1: Přehled dostupných operací[1] s články vzhledem ke statusu

Z tabulky číslo 1 nám vyplývá, že po uložení nového článku je hodnota statusu 1. Toto nám označuje, že je tento článek v první fázi celého životního cyklu článků v systému. V první fázi jsou pro autora dostupné všechny funkce určené pro autory článků, však si můžeme povšimnout, že ostatních rolí, jakožto je admin a recenzent, jsou zcela omezeny. Tento fakt se v průběhu životního cyklu článku mění. S přibývající hodnotou statusu ubývá dostupných operací pro autora, a taktéž se mění dostupné funkce pro ostatní. Však vyjímaje statusu hodnoty 0, který označuje probíhající úpravu článku. Při úpravě článku autorem, článek nabývá stav hodnoty 0 a tudíž ho může upravovat a smazat pouze upravující autor a pro ostatní spoluautory jsou jakékoliv operace s člán-

[1] V = Vytvořit, Z = zrušit, S = Smazat, T = Tisk, U = Upravit, OA = odeslat adminům, R= recenze/recenzovat

kem uzamknuty. Po dokončení a uložení úprav článek opět nabývá hodnoty 1 a všechny operace s článkem pro autory jsou dostupné. Tímto je zamezeno „přeuložení“ článku více autory najednou.

Po odeslání článku administrátorům status článku nabývá hodnotu 2, což znamená, že článek bude poté zobrazen administrátorům, kteří mají dostupnou možnost článek smazat, vytisknout a hlavně pro daný článek vytvořit recenzní řízení. Tím je rozuměno, že administrátor pro požadovaný článek vybere dle zaměření recenzenty. Jestliže je takto učiněno administrátorem, jsou pro článek přiřazeni recenzenti, vytvořeno recenzní řízení a status článku nabývá hodnoty 3. Zde se již dostáváme k bodu, kdy autoři mohou článek pouze vytisknout a nahlížet na podrobné informace. Recenzent má přiřazené recenzní řízení, a tudíž může článek stahovat a napsat na něj recenzi. Administrátor má na víc možnost recenzní řízení zrušit pro vybrané recenzenty.

5.3.2 Recenze

I zde u recenzí jsem rozlišoval stav, ve kterém se recenze nachází. Však zde je tento model rozlišování mnohem jednodušší. Rozlišují se zde pouze tři stavy, a to 0, 1, 2. Jakmile admin vybere recenzenty a započne recenzní řízení článku, jsou vytvořeny recenze pro jednotlivé recenzenty. Těmto recenzím je přiřazen status 0 a text recenze je vyplněn defaultním textovým řetězcem. V tomto stavu je recenze připravena pro zapsání textu recenzenty. Tudíž recenzent již existující recenzi pouze upravuje pomocí vloženého WYSIWYG editoru. Takto přidělenou recenzi může samozřejmě recenzent odmítnout či admin toto přiřazení zrušit. Jestliže recenzent napíše svou recenzi a uloží jí do systému, bude původní text aktualizován a recenzi se změní status z 0 na 1. Status 1 značí ty recenze, které již byly recenzenty zpracovány a jsou viditelné pro administrátory. Administrátor si zde může přečíst recenze, odeslat je autorům, zrušit požadavek recenze od vybraného recenzenta, či zrušit celé recenzní řízení článku. Jestliže je recenze v pořádku, administrátor odesílá recenzi autorům článku. Zde recenze nabývá status 2 a je viditelná pro autory. Po úpravě článku je

opět předán k novému recenznímu řízení. Tento proces se opakuje do té doby, než článek dosáhne finální podoby a statusu. Dále při konzultaci s vedoucím práce, panem doktorem Pexou, zazněla připomínka k vytváření recenze pouze ručně. Pan doktor podotkl, že jelikož recenzní systémy obsahují již předvyplněné šablonové možnosti, byla by vhodná jejich přítomnost i v tomto systému. S tímto doporučením jsem zcela souhlasil a rozhodl se vytvoření těchto šablon nabídnout katedře tělesné výchovy a sportu, jelikož zde pak recenzent může využít již předpřipravených obecných odpovědí a není nutné nic psát.

5.3.3 Nastavení uživatelského účtu

U nastavení účtu uživatele se jedná o obyčejný formulář pro editaci údajů a nahrání fotografie. Však tato část stojí za zmínku, neboť zde jsem přišel na několik problémů, které se netýkají samotného formuláře, ale dat, které jsou později zpracovávané v PHP. Při zpracovávaní údajů z formuláře, si generuji SQL dotaz pro PDO `prepare()` metodu pomocí cyklu `foreach` a taktéž tímto cyklem jsem chtěl přidávat parametry do placeholderů pomocí metody `bindParam()`. Zde však docházelo k chybě, jelikož metoda `bindParam()` vezme poslední dosazovanou hodnotu a dosadí jí do všech placeholderů. Pro názornost tuto situaci nastíním přímo v zápisu metod. Když zapíšeme metody bindParam za sebou takto:

```
$state->bindParam(':jmeno ', $jmeno, PDO::PARAM_STR);
$state->bindParam(':prijmeni ', $prijmeni, PDO::PARAM_STR);
$state->bindParam(':adresa ', $adresa, PDO::PARAM_STR);
```

Je totéž, jako kdybychom je zapsali takto:

```
$vyrok->bindParam(':jmeno ', $adresa, PDO::PARAM_STR);
$vyrok->bindParam(':prijmeni ', $adresa, PDO::PARAM_STR);
$vyrok->bindParam(':adresa ', $adresa, PDO::PARAM_STR);
```

Z výše uvedeného příkladu můžeme vyčíst podstatu chyby, jež jsem popisoval výše. Tuto chybu lze vyřešit použitím metody `bindValue()`. Tato metoda pracuje v tomto případě zcela tak, jak bychom očekávali. Tudíž zápis:

```
$vyrok ->bindValue(':jmeno ', $jmeno);
$vyrok ->bindValue(':prijmeni ', $prijmeni);
$vyrok ->bindValue(':adresa ', $adresa);
```

nám správně uloží do placeholderů požadované informace.

Dále jelikož jsem řešil ukládání obrázků do složky a nahrávání fotografie uživatele přes plugin cropit pro oříznutí obrázků, tak zde vyvstal další problém a to ten, že oříznutý obrázek se z pluginu dá vypsat pouze jakožto textový řetězec. Proto je potřeba v PHP metodě, kde zpracováváme obrázek, tento textový řetězec opět dekódovat na obrázek před samotným uložením do složky. Nejdříve je potřeba provést selekci samotného řetězce obrázku, jelikož na začátku tohoto řetězce se nachází informace o kódování obrázku. Tohoto jsem docílil pomocí PHP metody `explode(',', $oriznutyObrazek)` Dále pak už bylo pouze nutné konečnou hodnotu v poli dekódovat pomocí metody `base64_decode()` A obrázek bylo možné nahrát do určené složky jakožto soubor.

5.3.4 Správa uživatelů a rolí.

Každý rozsáhlejší systém, jenž pracuje na principu uživatelských účtů, nemůže fungovat bez správy uživatelů. Jelikož jsem se rozhodl pro co největší modifikovatelnost systému a práv uživatelů, bylo důležité, aby existovali moduly pro správu uživatelů a rolí. Tyto dva moduly jsem defaultně přidělil roli „Superadministrátor“.

Modul správy uživatelů obsahuje běžné operace s uživateli, jakožto jsou možnosti: „mute/ban“ uživatelského učtu, výpis informací o uživateli, možnost resetovat uživateli heslo při zapomenutí. Taktéž v tomto modulu lze přiřazovat

a odebírat jednotlivým uživatelům různé role. Dále modul obsahuje možnost pro potvrzení účtu uživatele. Toto vyplývá z požadavku katedry tělesné výchovy a sportu, aby systém byl „polouzavřený“, což znamená, že registrace pro uživatele je dostupná zcela všem bez výjimky, ale po registraci je účet neaktivní, a to do doby, než ho příslušná role pomocí správy uživatelů nepotvrdí. Takto může učinit defaultně pouze superadministrátor, avšak díky druhému modulu správy rolí, se toto nastavení může lišit.

Modul správy rolí obstarává, na jaké moduly mají určité role právo. Zde bude mít oprávněný uživatel právo modifikovat již existující role, vytvářet nové a dále jim přidělovat či odebírat dostupné moduly. V podstatě jde o možnost modifikování systému a práv rolí. Například můžeme autorovi přidat možnost články recenzovat, či mu ubrat již stávající možnosti. Dále pak zde můžeme vytvářet nové zcela speciální role, pro jednotlivé uživatele. Zde však musíme počítat s nefunkčností jazykových verzí, pokud toto ovšem neprovedeme zcela dle pokynů pro přidání nové role uživatele popsané v kapitole 7.1.

Taktéž správa rolí umožňuje snadněji přidat nový rozšiřující modul již existující, či nové roli. Toto je možné po nahrání souborů modulu do systému a jeho zanesení do databáze dle pokynů pro přidání nového modulu. Pokyny, jež zmiňuji budou dále popsány v kapitole 7.2.

5.3.5 Zobrazení funkcí

Zobrazení funkcí jsem naprogramoval tak, že po přihlášení uživatele do administrace za pomoci uživatelského jména a hesla, mohou nastat tři odlišné situace. Jestliže tento účet není ještě schválený, či je účet pozastavený, je zobrazena pouze stránka s vypsaným statusem účtu. Ale pokud je účet schválen, zobrazí se uživateli všechny dostupné moduly pro role, které jsou přiřazeny pro tento účet.

Samotný výpis modulů obstarává kontroler „Backoffice“, který si pomocí svého modelu převezme ze session role, jež jsou přiřazeny uživateli. Dle těchto rolí dále vyhledá skrze relační tabulku `roles_modules` příslušné moduly pro

určité role. Tyto informace jsou přes kontroler poskytnuty pohledu backoffice a vytisknuty dle požadavků. Zde však vyvstal problém v situaci, kdy existuje uživatel s více rolemi, které obsahují tentýž modul současně. Jelikož moduly při tisku mají své unikátní id které ovládá spouštění tohoto modulu, tak zde nastává komplikace ve výskytu více stejných id na jedné stránce a taktéž problému určení na jakém místě se modul vyžaduje. Respektive jestliže máme zobrazenou nabídku tohoto modulu na stránce vícekrát, například jeden nahoře a jeden veprostřed stránky, a klikneme na tlačítko tohoto modulu veprostřed, vyjede nám nabídka modulu u tlačítka nahoře. Toto bylo nutné řešit přidáním k id modulu taktéž označení role pro kterou je modul požadován a upraveném volací funkce pro modul. Tímto řešením bylo zamezeno duplicitnímu id na jedné stránce a taktéž vyřešen problém se správností umístění zobrazení modulu.

5.3.6 Stahování souborů

Ačkoli se jedná o celkem jednoduchou záležitost, tak s použití MVC architektury a úpravou URL adresy, jenž byla nastíněná v kapitole 3.2.2, jsem zde narazil na problém předání cesty stahovaného souboru bez rozčlenění na různé parametry. Z definice uvedené ve výše zmiňované kapitole vyplývá, že pokud pošleme cestu k souboru bez úpravy do URL adresy jako parametr, bude tato cesta rozčleněna na různé parametry. Dejme tomu, že chceme stáhnout soubor jehož cesta vypadá takto: `clanky/uživatel/clanek.pdf`. Po dosazení do URL adresy by vypadala adresa následovně:

```
https://adresa.cz/stahni/clanek/clanky/uživatel/clanek.pdf
```

Zde se nám již z předešlé definice URL adresa rozdělí podle lomítek na kontroler, metodu kontroleru a na parametry této metody. Toto jsem však nechtěl a tudíž jsem hledal jiné řešení jak tento soubor stáhnout. Nejdříve jsem pomýšlel na předávání cesty souboru skrze AJAX, avšak toto řešení bylo nešťastné

z hlediska zobrazení vyskakujícího okna pro uložení souboru. Proto jsem zvolil mnohem jednodušší a schůdnější cestu, a tou byla záměna lomítek v cestě souboru za speciální řetězce, jako je například „_lom_“, které se poté zase převedou zpátky na lomítka. Poté tato adresa nebude mít problém s rozčleněním a bude vypadat kupříkladu takto:

`https://adresa.cz/stahni/clanek/clanky_lom_uživatel_lom_clanek.pdf`

6 Testování

Zde u této práce jsem prováděl testování trojího typu, a to průběžné testování, finální testování systému a testování v provozu. Jak jsem již nastínil v předešlých kapitolách, bylo důležité provádět průběžné testování funkčnosti jednotlivých funkcí. Toto bylo prováděno z důvodu zjištění chyb obsažených v metodách, nebo správnosti hodnot, s kterými metody pracují. Tyto chyby, na které jsem přišel při průběžném testování, popisuji průběžně v předešlých kapitolách.

Po dokončení systému jsem započal testování finální. Toto testování spočívalo v simulaci celého průběhu vkládání, recenzování a schvalování článků. Tímto byl systém otestován jakožto celek a účelem bylo zjistit, jak jednotlivé části systému spolu komunikují. Výsledky tohoto testu, byly výborné a systém při tomto testování fungoval tak, jak bylo očekáváno.

Testování v provozu zahrnuje průběžné testování již po nasazení systému do provozu. Toto je nutné zajistit z důvodu možného výskytu chyby při větší zátěži systému. Proto výsledky toho testu budou odhaleny až časem a používáním systému.

7 Rozšíření systému

V této kapitole budu popisovat možné způsoby a postupy rozšíření aplikace novými moduly a rolemi. Tyto postupy jsou důležité pro to, aby moduly a role správně fungovali a byly pro ně načítány všechny jazykové verze.

Soubory jazykových verzí jsou uloženy v adresáři „system/langpacks“. V případě rozšíření systému o další jazyk, je nutné vytvořit PHP soubor pro daný jazyk, který bude pojmenován takto: „lang_zkratkajazyka.php“. Například u českého jazyka tento soubor pojmenujeme „lang_cz.php“. V těchto souborech poté definujeme konstanty s požadovaným prefixem „_l_“, který jsem si zvolil, jako označení jazykových konstant. Samozřejmostí je taktéž přidání elementu volby tohoto jazyka do systému.

7.1 Vytvoření nové role

Vytváření rolí je možné dvěma způsoby, a to je „rychlé“ a „kompletní“ přidání role. Roli můžeme vytvořit rychle ve správě rolí pouze pomocí přidávacího formuláře role. Toto řešení ovšem funguje pouze pro jednu jazykovou verzi a je jí možné využít v případě, že tato role je vytvářena speciálně pro jednu osobu či skupinu uživatelů, jenž ovládají stejný jazyk.

Pokud však chceme vytvořit kompletní roli, která bude správně překládána i do ostatních dostupných jazyků, musíme název role zapsat ve správné formě a tu taktéž zanést do definovaných jazykových slovníků. Například pokud chceme přidat roli s názvem „specialista“ musíme toto zapsat s prefixem „_l_“ takto „_l_specialista“. Tudíž pokud se v názvu nachází tento prefix, je název role vytisknut jako konstanta, která musí být definována v požadovaném slovníku. Jestliže se v názvu nenachází prefix, je tento název vytisknut pouze jako textový řetězec.

7.2 Vytvoření nového modulu

Moduly lze přidávat pouze mimo webovou aplikaci. Prvním krokem pro přidání modulu je nahrání kontroleru, modelu a pohledu modulu. Po nahrání požadovaných souborů, je nutné vložit moduly do databázové tabulky „modules“. U modulů je velice důležité k názvům modulů přidávat prefix „_ *l*_ “, jelikož se u modulů předpokládá, že budou dostupné více lidem, a tudíž i ve více jazykových formách. Z tohoto důvodu je nezbytné tento název definovat v souborech jazykových verzí.

8 Závěr

V této bakalářské práci jsem shrnul a popsal nejdůležitější aspekty, které je třeba řešit a dodržovat při práci na podobně rozsáhlých systémech. Taktéž zde popisuji problémy, na které jsem při samotné tvorbě systému narazil a také způsob řešení, který jsem použil.

Doufám, že tento systém bude v mnohém ulehčovat recenzní řízení článků pro časopis Studia Kinanthropologica a že tato bakalářská práce dokáže v budoucnu posloužit jakožto manuál, či návod k tomu, jak by měli budoucí programátoři při tvorbě podobných systémů postupovat.

Při práci na bakalářské práci a tvorbě samotného systému, jsem získal mnoho cenných informací a zkušeností, které určitě zúročím v mé budoucí profesi.

Seznam použité literatury a zdrojů

[1] CONOLLY, Thomas, Carolyn E. BEGG a Richard HOLOWCZAK. *Mistrovství - databáze: profesionální průvodce tvorbou efektivních databází.* Brno: Computer Press, 2009. ISBN 978-80-251-2328-7.

[2] *Objektově relační mapování.* Wikipedia [online]. 2014 [cit. 2017-04-24]. Dostupné z:<https://cs.wikipedia.org/wiki/Objektově_relační_mapování>

[3] *PHP Introduction.* W3schools [online]. c1999-2017 [cit. 2017-04-24]. Dostupné z:<https://www.w3schools.com/php/php_intro.asp>

[4] PITT, Chris. *Pro PHP MVC.* New York: Distributed to the book trade worldwide by Springer Science+Business Media, c2012, s. 2-31. Expert's voice in open source. ISBN 9781430241645.

[5] *Introduction to SQL.* W3schools [online]. c1999-2017 [cit. 2017-04-24]. Dostupné z:<https://www.w3schools.com/SQL/SQL_intro.asp>

[6] *JavaScript.* Wikipedia [online]. 2017 [cit. 2017-04-24]. Dostupné z:<https://cs.wikipedia.org/wiki/JavaScript>

[7] *Úvod do JavaScriptu.* Jak psát web [online]. [cit. 2017-04-24]. Dostupné z:<https://www.jakpsatweb.cz/javascript/javascript-uvod.html>

[8] *Třídy, dědičnost a OOP v Javascriptu - I.* Zdroják [online]. 2010 [cit. 2017-04-24]. Dostupné z: <https://www.zdrojak.cz/clanky/oop-v-javascriptu-i/>

[9] *AJAX.* Wikipedia [online]. 2016 [cit. 2017-04-24]. Dostupné z:<https://cs.wikipedia.org/wiki/AJAX>

[10] RESIG, John. *JavaScript a Ajax: moderní programování webových aplikací*. Přeložil Ondřej. BAŠE, přeložil Ondřej. ŽIŽKA. Brno: Computer Press, 2007. ISBN 80-251-1824-x.

[11] CLARKE, Justin. *SQL Injection Attacks and Defense*. Burlington, MA: Syngress Pub., 2009, s. 6-13, s. 6-8. ISBN 15-974-9424-0.

[12] *PHP Data Objects*. php [online]. [cit. 2017-04-24]. Dostupné z: <`http://php.net/manual/en/book.pdo.php`>

[13] *Co je Cross-site scripting jak mu předcházet*. Zdroják [online]. [cit. 2017-04-24]. Dostupné z: <`https://www.zdrojak.cz/clanky/co-je-xss-jak-mu-predchazet/`>

[14] SHIREY, Robert W. *Internet Security Glossary*. [online]. [cit. 2015-03-18]. Dostupné z: <`http://tools.ietf.org/html/rfc2828`>

[15] *HTML form honeypots and autofill/autocomplete*. The electric toolbox [online]. 2010 [cit. 2017-04-24]. Dostupné z: <`https://www.electrictoolbox.com/html-form-honeypots-autofill/`>

[16] *Invisible reCAPTCHA*. Google [online]. [cit. 2017-04-24]. Dostupné z: <`https://www.google.com/recaptcha/intro/comingsoon/invisiblebeta.html`>

[16] *PHP Manual*. PHP [online]. [cit. 2017-04-24]. <`Dostupnéz:http://php.net/manual/en/index.php`>

Seznam obrázků

Seznam tabulek

A Příloha

CD/DVD

B Příloha

Přihlášení: `http://www.pf.jcu.cz/stru/katedry/tv/studia_kinantropologica/pages/index.html`

Systém: `http://www.pf.jcu.cz/stru/katedry/tv/studia_kinantropologica/system/backoffice`