

Informační systém pro výzkumnou skupinu Research Group Information System

Bc. Kamil Lerch

Diplomová práce
2014



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2013/2014

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Kamil Lerch**
Osobní číslo: **A12770**
Studijní program: **N3902 Inženýrská informatika**
Studijní obor: **Informační technologie**
Forma studia: **kombinovaná**

Téma práce: **Informační systém pro výzkumnou skupinu**

Zásady pro vypracování:

1. Vypracujte literární rešerši v oblasti informačních systémů výzkumné skupiny.
2. Provedte návrh a specifikaci požadavků na systém.
3. Navrhněte strukturu databáze a datový model.
4. Navrhněte vhodné prostředky pro vytvoření konečného systému.
5. Uvedený návrh implementujte jako funkční prototyp.
6. Věnujte pozornost zabezpečení aplikace.
7. Provedte vyhodnocení přínosů aplikace a další možný rozvoj.

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. SOBELL, Mark G. Mistrovství v Linuxu: příkazový řádek, shell, programování. Vyd. 1. Brno: Computer Press, 2007. ISBN 978-80-251-1726-2.
2. DAN, Cederholm a Robert Banh TECHNICAL REVIEWER. Webdesign s webovými standardy. Vyd. 1. Brno: Zoner Press, 2004, 256 s. Wrox beginning guides. ISBN 80-86815-15-3.
3. HLAVENKA, Jiří a Robert Banh TECHNICAL REVIEWER. Vytváříme WWW stránky a spravujeme moderní web site. 6. vyd. Brno: Computer Press, 2002, 354 s. Wrox beginning guides. ISBN 80-7226-748-5.
4. NIXON, Robin. Learning PHP, MySQL, JavaScript, and CSS. 2nd ed. Sebastopol, CA: O'Reilly, 2012, xxi, 556 p. ISBN 14-493-1926-2.
5. CONVERSE, Tim a Joyce PARK. PHP5 and MySQL bible. [3rd ed.]. Indianapolis, IN: Wiley, 2004. ISBN 07-645-5746-7.
6. BORONCZYK, Tim a Joyce PARK. Beginning PHP6, Apache, MySQL web development. [3rd ed.]. Indianapolis, IN: Wiley Pub., 2009, xxvii, 807 p. Wrox beginning guides. ISBN 978-0-470-39114-3.
7. LENGSTORF, Jason a Robert Banh TECHNICAL REVIEWER. Beginning PHP6, Apache, MySQL web development. [3rd ed.]. [Berkeley, Calif: Apress], 2010, xxvii, 807 p. Wrox beginning guides. ISBN 978-143-0228-486.
8. BOIKO, Bob. Content management bible. 2nd ed. Indianapolis, IN: Wiley Pub., xlvii, 1122 p. ISBN 07-645-7371-3.

Vedoucí diplomové práce:

Ing. Radek Šilhavý, Ph.D.

Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce:

21. února 2014

Termín odevzdání diplomové práce:

20. května 2014

Ve Zlíně dne 21. února 2014

prof. Ing. Vladimír Vašek, CSc.
děkan



doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

ABSTRAKT

Práce se zabývá problematikou správy informací mezi členy výzkumné skupiny. Z toho důvodu byla provedena analýza potřeb sdílení informací, zadávání úkolů, kontrolu členů skupiny a jejich aktivity. Z těchto poznatků byl navržen informační systém na jednoduchou komunikaci ve skupině. Systém se snaží implementovat nejlepší řešení, která jsou v dnešní době objevena. Při řešení byla použita softwarová architektura, která odděluje datový model aplikace, grafické rozhraní a logiku řízení. Tato architektura se nazývá MVC. Pro implementaci byl vybrán framework napsaný ve skriptovacím jazyce PHP – CakePhp. Uložení dat je řešen databází MySQL. Aplikace bude typu Klient-Server.

Klíčová slova:

výzkumná skupina, informace, úkoly, informační systém, softwarová architektura, MVC, framework, PHP, CakePhp, MySQL, Klient-Server

ABSTRACT

This work deals with the problems of information among members of the research group. For this reason, an analysis of the need for sharing of information, tasks, control group members and their activities. From these findings, the information system designed to simple communication in the group. The system seeks to implement the best solutions that are nowadays discovered. When the solution was applied software architecture that separates data model application, graphical interface and control logic. This architecture is called MVC. To implement the chosen framework written in the scripting language PHP - CakePHP. Data storage is handled by MySQL. Applications will be client-server.

Keywords:

Research group, information, tasks, information system, software architecture, MVC, framework, PHP, CakePHP, MySQL, Client-Server

MOTTO

Nedělejte věci složitě jen proto, že to umíte. Udělejte to radši rychle a jednoduše, ať se v tom vyznají i normální uživatelé. Systém nepíšete pro sebe, ale pro ně.

PODĚKOVÁNÍ

Rád bych poděkoval svému vedoucímu Ing. Radku Šilhavému, Ph.D, který mi pomohl svými připomínkami a radami, bez kterých by tato diplomová práce nemohla vzniknout.

OBSAH

ABSTRACT.....	5
MOTTO.....	6
PODĚKOVÁNÍ.....	6
ÚVOD.....	10
I TEORETICKÁ ČÁST.....	12
1 ROZBOR PROBLEMATIKY.....	13
1.1 INFORMAČNÍ SYSTÉM - METODIKA VÝVOJE.....	13
1.1.1 Vodopádový přístup.....	13
1.1.2 RUP (Rational Unified Process).....	14
1.1.3 Prototypový přístup.....	15
1.1.4 Inkrementální přístup.....	15
1.1.5 Spirální přístup.....	16
1.1.6 RAD přístup (Rapid Application Development).....	17
1.1.7 Extrémní programování.....	17
1.2 TYPY ÚTOKŮ.....	18
1.2.1 Injection a SQL Injection.....	18
1.2.2 Spuštění nebo načtení souboru.....	19
1.2.3 Krádež session (sezení).....	19
1.2.4 Cross Site Scripting (XSS).....	19
1.2.5 Cross Site Request Forgery (CSRF).....	20
1.2.6 JavaScript Hijacking.....	20
1.2.7 Clickjacking (Click Hijacking).....	20
1.3 ZABEZPEČENÍ - SSL.....	21
1.4 DATABÁZE.....	21
1.4.1 Systém řízení báze dat (SŘBD).....	21
1.4.2 Relační databáze.....	22
1.4.3 MySQL.....	22
1.5 PHP FRAMEWORK.....	22
1.5.1 CakePhp.....	22
1.6 MVC.....	23
1.7 VĚDA A VÝZKUM.....	24
1.7.1 Výzkum dle funkce a využití výsledků v praxi.....	24
1.7.2 Výzkum dle realizace výzkumu.....	24
1.7.3 Výzkum dle použité metodologie.....	24
2 REŠERŠE STÁVAJÍCÍCH ŘEŠENÍ.....	25
2.1 INFORMAČNÍ SYSTÉMY PRO VÝZKUMNÉ SKUPINY.....	25
2.2 MAVENLINK.....	26
2.2.1 Výhody.....	26
2.2.2 Nevýhody.....	26

2.3	PLANCAKE.....	27
2.3.1	Výhody.....	27
2.3.2	Nevýhody.....	27
2.4	FLYSPRAY.....	28
2.4.1	Výhody.....	28
2.4.2	Nevýhody.....	28
2.5	MANTIS BUG TRACKER.....	29
2.5.1	Výhody.....	29
2.5.2	Nevýhody.....	29
II	PRAKTICKÁ ČÁST.....	30
3	NÁVRH.....	31
3.1	SOFTWARE PRO VYTVOŘENÍ NÁVRHU.....	31
3.2	POŽADAVKY NA SYSTÉM.....	31
3.2.1	Nefunkční požadavky.....	31
3.2.2	Funkční požadavky.....	32
3.3	DIAGRAM PŘÍPADŮ UŽITÍ.....	34
3.4	PŘIHLÁŠENÍ.....	35
3.5	REGISTRACE.....	37
3.6	ZALOŽIT PROJEKT.....	39
3.7	PŘIDAT ÚKOL.....	41
3.8	PŘIDAT SOUBOR.....	43
3.9	ZAVŘÍT PROJEKT.....	45
3.10	ZMĚNIT STAV ÚKOLU.....	47
3.11	DIAGRAM TŘÍD.....	49
	49
3.12	DIAGRAM SEKVENCÍ.....	50
3.13	DIAGRAM AKTIVIT.....	50
3.14	E-R DIAGRAM.....	50
4	IMPLEMENTACE.....	52
4.1	GUI.....	52
4.1.1	HTML.....	52
4.1.2	Hlavička – nepřihlášený.....	52
4.1.3	Registrace.....	53
4.1.4	Přihlášení.....	53
4.1.5	Hlavička – přihlášený.....	53
4.1.6	Přidání projektu.....	54
4.1.7	Přidání úkolu.....	54
4.1.8	Přidání uživatele k projektu.....	55
4.1.9	Přidání uživatele k úkolu.....	55

4.1.10	Přidání komentáře.....	56
4.1.11	Detail projektu.....	56
4.1.12	Náhled stránky Regris.....	57
4.1.13	Detail úkolu.....	58
4.2	NÁSTROJE.....	59
4.2.1	PhpStorm.....	59
4.2.2	WAMP.....	59
5	VÝSLEDKY.....	60
5.1	TESTOVÁNÍ.....	60
5.1.1	Registrace.....	60
5.1.2	Změna osobních údajů.....	61
5.1.3	Přidání projektu.....	61
5.1.4	Editace projektu.....	61
5.1.5	Přidání uživatele.....	62
5.1.6	Přidání úkolu.....	62
5.1.7	Editace úkolu.....	62
5.1.8	Přidání komentáře.....	63
5.1.9	Editace komentáře.....	63
5.1.10	Zavření úkolu.....	63
5.1.11	Zavření projektu.....	63
5.1.12	Náhled úkolu.....	64
5.1.13	Náhled projektu.....	64
	ZÁVĚR.....	65
	SEZNAM POUŽITÉ LITERATURY.....	67
	SEZNAM OBRÁZKŮ.....	69
	SEZNAM TABULEK.....	71
	SEZNAM PŘÍLOH.....	71

ÚVOD

Pokud chtějí uživatelé komunikovat a sdílet informace po internetu, je nejjednodušším způsobem použít informační systém. Informační systém je soubor metod pro komunikaci mezi uživateli, servery, ukládáním dat na externí nebo lokální úložiště a spravování těchto dat. Tyto data se musí zabezpečit před zneužitím. Data je možné zašifrovat některou z existujících metod nebo při přístupu k datům ověřovat autentizací uživatele. Toho je možné dosáhnout buď pomocí kombinace soukromý a veřejný klíč nebo pomocí uživatelského jména a hesla.

Ideální typ aplikace pro výzkumnou skupinu by mohla být webová aplikace typu Klient-server. Klientská aplikace by měla komunikovat s webovým serverem pomocí protokolu SSL, který využívá asymetrické šifry, a tudíž by šlo o zabezpečenou komunikaci HTTPS. Tímto dojde k zabezpečení hesel a dat.

Každá výzkumná skupina má určitou hierarchickou strukturu. Tato struktura se může lišit dle zaměření dané skupiny. Systém by měl modelovat aspoň dvě hierarchické úrovně a dle toho nastavovat viditelnost sekcí v aplikaci. Uživatelé mohou vytvarovat projekty, úkoly, ukládat a přiřazovat soubory známých formátů dle přidělených oprávnění.

Projekt je návrh určitého řešení vedoucího k vytvoření nějakého produktu nebo poznatku. Ten má buď jasné zadání a vyřešení tohoto zadání se rovná úspěšnému dokončení projektu, nebo se skládá z podprojektů, kterým se říká úkoly. Pokud projekt obsahuje úkoly, není možné jej dokončit, dokud se jednotlivé úkoly (pokusy, části programu nebo studie) nedokončí. Projekt může mít nastavený konec (deadline), který značí do kdy je platný. Toto může být nezbytné z hlediska vypršení grantu nebo odevzdání projektu zadavateli. Po vypršení platnosti se musí zobrazit upozornění a nastavit stav na „nedokončeno“ nebo posunout čas platnosti.

V implementovaném systému bude nutné zohlednit všechny tyto požadavky na informační systém. Bude potřeba vybrat správnou databázi a navrhnout databázovou strukturu, ve které bude jednoduché vyhledávat a ukládat do ní. Tyto operace musí být optimalizované pomocí indexu.

Velkou výhodou může být použití skriptovacího jazyka. Na výběr jsou sice i kompilované jazyky, ale i přesto, že existuje spousta dobrých frameworků, vývoj informačního systému by byl kvůli kompilacím zbytečně zdlouhavý. Na výběr je možné vybírat z PHP, Pythonu, Ruby nebo Perlu. Pro vytvoření aplikace je možné použít frameworky, které jsou úplně

zdarma. Jedná se například o PHP framework CakePhp, pro Python je to Django a pro Ruby například Espresso.

Pro vývoj aplikací se zdá být v dnešní době nejvhodnější použití softwarové architektury Model-View-Controller zkráceně MVC, kde jsou objekty s daty odděleny od řízení a zobrazování.

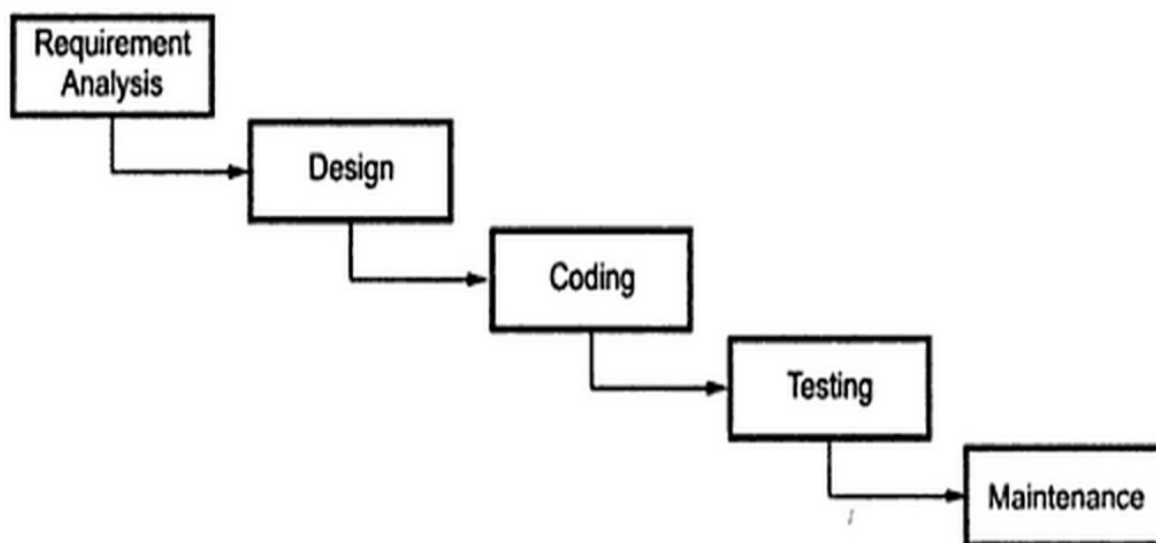
I. TEORETICKÁ ČÁST

1 ROZBOR PROBLEMATIKY

V této kapitole si rozebereme několik problematik spojených s vývojem softwaru a informačních systémů. Seznámíme se s možnými způsoby zabezpečení. Porovnáme si možnosti využití různých typů databází. Dále se zaměříme na vybrání nejvhodnějšího frameworku, v kterém se výsledný systém bude implementovat. Také si podrobněji vysvětlíme, co je to MVC. Nakonec si řekneme něco o požadavech výzkumných skupin na informační systémy a základním rozdělení výzkumných skupin.

1.1 Informační systém - Metodika vývoje

Jedná se o postupy, které vedou k vytvoření softwaru. Má blízko k softwarovému inženýrství, které se zabývá návrhem, plánováním, vývojem a testováním. Pro vytváření informačních systémů je to metodika při využívání postupů v týmu nebo organizaci. Díky těmto metodám se projekty neprodražují a je možné je dokončit ve stanovených termínech bez zpoždění.



Obrázek 1. Vodopádový přístup

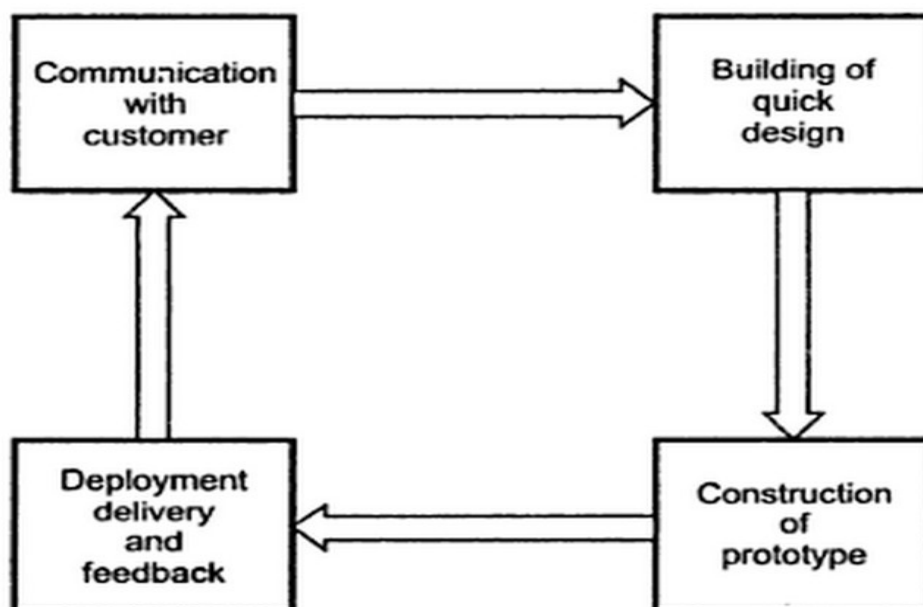
1.1.1 Vodopádový přístup

Tento přístup rozděluje vývoj softwaru do čtyř fází, jak je vidět na obrázku 1 [14]. Tyto fáze na sebe navazují a je nemožné jakoukoliv fázi přeskočit nebo vynechat. Jedná se o zpracování požadavků a specifikaci těchto požadavků. Další fází je návrh softwaru, implementace, testování softwaru a údržba systému. Vodopádový přístup striktně vyžaduje

aby každá fáze byla před přechodem do další fáze vždy dokončena a perfektně zpracovaná [1].

1.1.2 RUP (Rational Unified Process)

Je metodika, která podrobně popisuje systematický a organizovaný přístup k vývoji softwaru. Je vlastněna společností IBM, která ji dále vyvíjí a distribuuje. Cílem metodiky je dodávat v termínu a s předem daným rozpočtem kvalitní software, dle přání koncových uživatelů. Podrobně popisuje jednotlivé fáze životního cyklu vývoje softwaru, definuje všechny role zapojené do procesu, používané komponenty, stanovuje odpovědnosti a popisuje vývojové procesy. RUP udává základní problémy a příčiny neúspěšných projektů. Těmi mohou být nepřesné pochopení potřeb zákazníka, nemožnost změnit specifikaci v průběhu implementace, chybu návrhu a další problémy. Pro eliminaci je určeno několik zásad pro vývoj softwaru, která vychází z kolekce osvědčených praktik a postupů při jeho vývoji. Tato metodika je použitelná pro jakýkoliv rozsah projektu, ale díky vysoké rozsáhlosti modelu je vhodné přizpůsobit metodiku specifickým potřebám. RUP je vhodnější spíše pro rozsáhlejší projekty a větší vývojové týmy, neboť klade důraz na analýzu a design, plánování, řízení zdrojů a dokumentaci [2].



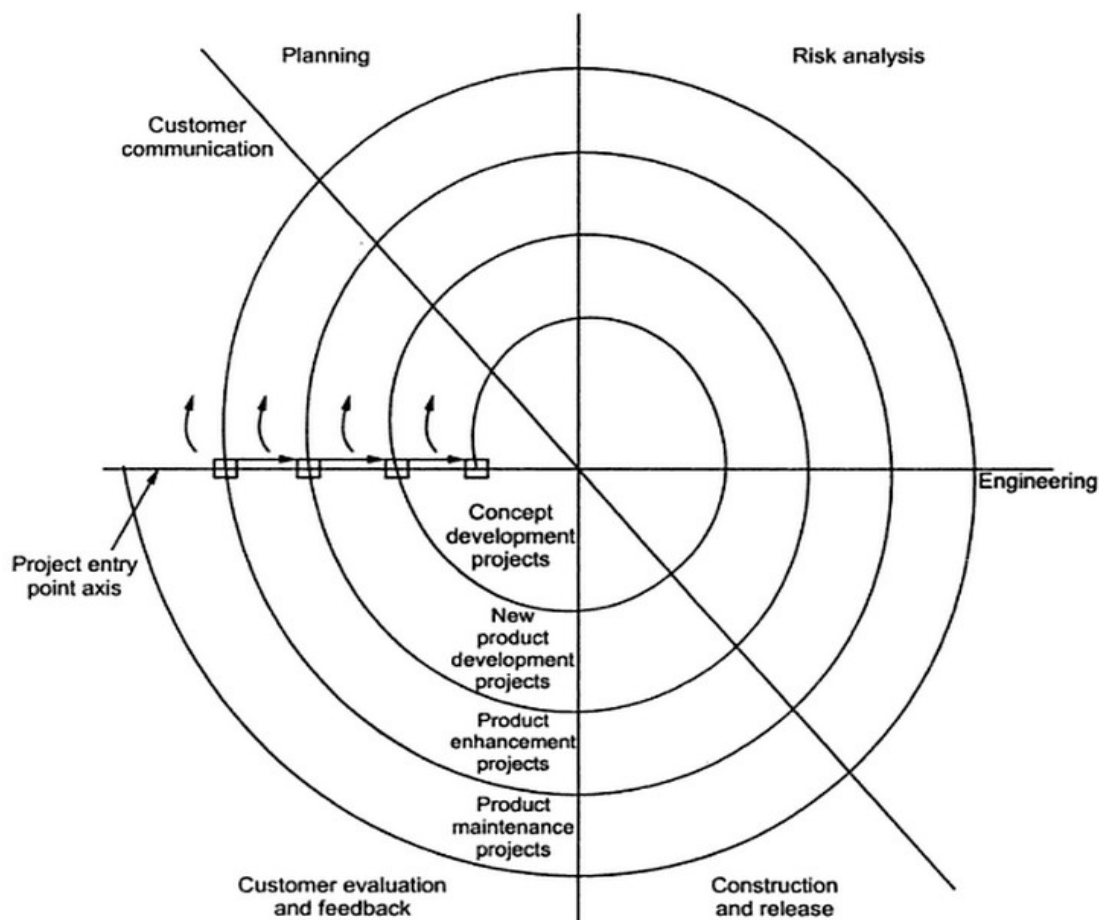
Obrázek 2. Prototypový přístup

1.1.3 Prototypový přístup

Vývoj softwaru v případě tohoto přístupu je v tom, že se vytváří prototypy. Tyto prototypy jsou malými verzemi výsledného softwaru. Prototypy se dělají proto, aby bylo možné co nejdříve představit zákazníkovi postup ve vývoji softwaru. Jedná se například o přenastavení například přihlašovací stránky. Poté by měl být tento prototyp zahozen. Základní princip tohoto přístupu je takový, že tento přístup není samostatným ani kompletním přístupem metodiky vývoje. Jedná se hlavně o přístup ke každé části vývoje samostatně. Jde vlastně o rozbití metodik, které jsou větší a tradiční. Snaha je především o snížení nebezpečí projektových rizik. V tomto přístupu je možné projekt rozdělit na menší části a tyto části zjednodušit tak, že je možné provést změny i v průběhu procesu vývoje. Důležitou osobou je v tomto přístupu uživatel. Ten se podílí na vývojovém procesu celého softwaru. Tento fakt zvyšuje pravděpodobnost přijetí konečného výsledku softwaru zákazníkem. Malé části softwaru jsou vyvíjeny iterativním procesem. Tento proces probíhá tak dlouho, dokud není prototyp nevyvinut tak, že jsou všechny požadavky uživatele splněny. Tento případ je vidět na obrázku číslo 2 [14]. Spousta prototypů je vyvíjena s tím, že budou zahozeny. Je však ale možné za určitých podmínek tento prototyp zachovat a dostat se od prototypu k plně funkčnímu systému nebo jeho části [3].

1.1.4 Inkrementální přístup

Tento přístup je ideální pro kombinaci sekvenčních a iteračních metodik vývoje softwaru. Cílem tohoto přístupu je omezit rizika z rozdělováním projektu. Když se nám povede projekt rozdělit na menší segmenty je možné zjednodušit možnost zavedení změn během vývoje. Základem tohoto přístupu je provádění vývoje pomocí sérií malých vodopádových modelů, kde je každá iterace prováděna jen pro malou část systému. Pokud je tento malý vodopád dokončen, je možné pokračovat s dalším přírůstek. Další obecné požadavky musí být definovány ještě dříve, než se přikročí k dalšímu vývoji pro jednotlivé systémové přírůstky. Fáze tohoto přístupu se dělí na prvotní koncept, analýzu požadavků, design architektury a systémové jádro. Každý je definován vodopádovým přístupem. Jako další následuje iterativní prototypový přístup. Tento přístup vrcholí instalací konečného prototypu jako plně funkčního systému [3].



Obrázek 3. Spirální přístup

1.1.5 Spirální přístup

Toto je přístup pro vývoj softwaru, který spojuje dohromady prvky designového přístupu a prototypového přístupu. Je to z důvodu, aby byly zkombinovány výhody konceptu shora-dolů (prototypování) a zdola-nahoru (designování). Spirálový přístup je zaměřen na analýzu a minimalizaci projektových rizik. Toho je docíleno rozdělením projektu na malé části a tím pádem je možné provádět změny i během procesu vývoje. Během vývoje je možné vyhodnocovat rizika a zvážit další postupy pro pokračování projektu v průběhu životního cyklu. Každá otočka spirály spouští vždy stejnou množinu kroků pro každou část tvorby produktu. Během jednoho otočení spirály jsou vždy spouštěny čtyři základní fáze. Tyto fáze se nazývají analýza, vyhodnocení alternativ, identifikace a řešení rizik, vývoj produktu, kontrola výsledků a plánování, jak ukazuje obrázek číslo 3 [14]. Na začátku každého cyklu je důležité identifikovat zapojené subjekty do vývoje softwaru a zanést jejich podmínky kladené na úspěch iterace. Nakonec je z každého cyklu prováděna revize a předání [1].

1.1.6 RAD přístup (Rapid Application Development)

Přístup RAD se používá k vývoji softwaru, který potřebuje řešit problém pomocí iterativního vývoje prototypů. Mezi zásady RAD patří hlavně co nejrychlejší vývoj a zavedení systémů vysoké kvality při co nejnižších investičních nákladech. Je zde snaha snížit nebezpečí rizik rozdělením projektu na části o menší velikosti a umožnění změn během vývoje softwaru. Rapid Application Development se snaží o co nejrychlejší vývoj s co nejvyšší kvalitou systémů především pomocí iterativního prototypování. Toto prototypování je použito ve všech stádiích vývoje. Dále je velice důležité mít co nejvíce aktivních a do vývoje softwaru zapojených uživatelů. Další podstatnou částí RAD jsou správné nástroje pro vývoj. Tyto nástroje mohou zahrnovat generátory GUI, programovací jazyky 4. generace, generátory kódu a OO techniky. Velký důraz je kladen na naplňování business potřeb. Tyto potřeby mají přednost před technologickou nebo technickou dokonalostí. Na to je kladen menší důraz. Při řízení projektu je důležité stanovit priority vývoje. Důraz je kladen na zapojení uživatelů. Jejich aktivita je nutností. Při tomto přístupu je vytvářena dokumentace, aby byl v budoucnu snazší vývoj a údržba [4].

1.1.7 Extrémní programování

XP je agilní metodologie, která předepisuje určité činnosti všem účastníkům vývojového procesu. Jedná se o tradiční činnosti, které jsou však dovedeny do extrému. Díky tomu by mělo být extrémní programování schopné se lépe přizpůsobit měnícím se požadavkům zákazníků a dodávat systém vyšší kvality. Extrémní programování uznává čtyři hodnoty. Těmi jsou komunikace, jednoduchost, zpětná vazba a odvaha. V rámci projektu je třeba udržovat kvalitní komunikaci mezi všemi zainteresovanými subjekty. Pokud selže komunikace mezi programátorem a zákazníkem, dostaneme zákazník software, který mu nemusí vyhovovat. XP přichází s postupy práce, které komunikaci podporují nebo i vyžadují. Patří mezi ně například párové programování. V extrémním programování se vždy programuje pouze to, co je třeba ke splnění aktuálních požadavků. Nikdy se nevytváří kód, který se bude „nejspíš“ hodit v budoucnu. XP vychází z předpokladu, že požadavky se mohou zítra změnit, jakákoliv práce navíc by pak byla zbytečná. Plýtvání se tedy předchází, kód navíc se nepíše. Zpětná vazba se v projektu projevuje hned na několika místech. V první řadě programátoři neustále tvoří jednotkové testy, takže v krátkých časových intervalech dokáží ověřit, že vše funguje tak jak má a že se nic poslední změnou kódu nerozbito. Dále je zde například zpětná vazba pro zákazníky, kteří prováděním svých akceptačních testů dokáží

zjistit, v jakém stavu se program nachází. Odvaha je u programátora potřeba, pokud například narazí na významný problém, jehož oprava nejspíš způsobí kaskádu souvisejících problémů. Pustit se do tak velké změny vyžaduje odvahu. Stejně tak vyžaduje odvahu zahodit kód, na kterém programátor pracoval třeba celý den, ale stále se mu nedaří ho zprovoznit. Zahození celodenní práce vyžaduje odvahu, ale druhý den programátor pravděpodobně dokáže najít optimální řešení mnohem rychleji [5].

1.2 Typy útoků

Zabezpečení citlivých dat, postupů a řešení má při implementaci informačního systému nejvyšší prioritu. Pokud se k webové aplikaci dostanou neznámí uživatelé, je téměř jisté, že se pokusí získat neoprávněný přístup. Servery, které jsou dostupné veřejnosti na internetu, jsou neustále vystavovány útokům. Systémy mohou útočníci napadnout jak na straně serveru, kde může běžet operační systém, ať už Windows nebo Linux, nebo přímo na straně uživatele. Uživatel může mít napadený buď přímo svůj systém nebo mu může být podstrčen škodlivý software přímo webovou stránkou v prohlížeči. V této práci budou zanalyzované útoky především při komunikaci uživatele s webovou aplikací přes internetový prohlížeč [6].

1.2.1 Injection a SQL Injection

První typ útoku může být způsoben špatným ošetřením vstupu od uživatele. Jedná se většinou o parametry formuláře nebo například samotné URL webového prohlížeče. Tato situace nastává především při sestavování dotazů a zasilání parametrů do vzdálených systémů. Tento typ útoku se nazývá SQL Injection. Je způsoben především špatným sestavováním parametrů při vytváření SQL dotazů. Tento problém má velice jednoduché řešení. Stačí nesestavovat SQL dotaz pomocí jednoduchého skládání řetězců. V případě nových frameworků můžeme použít pojmenovaných nebo pozicových parametrů. Ty dokáží pomocí skládání jména jednotlivých parametrů vytvořit správné obalení parametrů. Injection útok se ale netýká jen SQL. V každém dotazovacím systému by bylo možné v případě neošetření provést tento útok [6].

1.2.2 Spuštění nebo načtení souboru

Pokud je možné v aplikaci pomocí parametru načíst nebo spustit soubor z disku, je možné provést tento útok. Jedná se o případy, kdy není správně kontrolován vstup. Tímto je možné přesvědčit systém aby načel nebo spustil soubor na straně serveru. Jedná se například o soubor passwd na linuxových serverech. Obranou proti tomuto útoku je buď možnost nepovolit spouštění souboru jiných než v určitém adresáři a nebo kontrola parametrů dle seznamu povolených hodnot. [6]

1.2.3 Krádež session (sezení)

Uživatelům se po prvním načtení stránky aplikace vytváří sezení. Toto sezení se ukládá pod identifikátorem, například (PHPSESSID) pomocí cookies nebo pomocí předávání jako parametry v URL. Tohoto je ale možné zneužít. Pokud útočník zvládne získat tento identifikátor sezení, může se tvářit jako přihlášený uživatel, kterému tento identifikátor ukradl. Z tohoto důvodu je nutné tyto identifikátory náhodně generovat nejlépe s pravděpodobností rovnoměrného rozložení. Dále je výhodné kontrolovat IP adresu uživatele. Pokud aplikace používá cookies, mělo by být samozřejmostí omezit cookies jen na určitou doménu, kontrolovat IP adresu odkud se uživatel přihlásil a také omezit platnost takto vytvořené cookies [6].

1.2.4 Cross Site Scripting (XSS)

Pokud dovolujeme uživateli komunikovat se serverem pomocí nějakých vstupů, je důležité, aby hodnoty těchto vstupů nebyly vypisovány na výstup do HTML stránky přímo, ale jen jako text. Toho je možná docílit zakódováním a nahrazením HTML znaků za prostý text. Jinými slovy, musí dojít k nahrazení pro HTML klíčových znaků za vyjádření zakódovaných nebo nahrazených entit. Vstupy, které nechceme převést nebo nahradit, pak musíme explicitně označit. Pokud bychom chtěli uplatnit opačný přístup, to znamená, že by jsme označovali text, který se má převést, mohlo by dojít k přehlédnutí některého slabého místa. To by mohlo způsobit chybu XSS. Prevenci před tímto útokem lze rozdělit na dvě části. Jednou je oprava HTML kódu, čehož se docílí doplněním chybějících značek a druhá je kontrola a odstranění nalezených značek, které nejsou povoleny [6].

1.2.5 Cross Site Request Forgery (CSRF)

Tento útok předpokládá, že se uživatel právě přihlásil do aplikace a má platné cookies. Tento útok se dá nejlépe provést pokud jsou v cookies uloženy přihlašovací údaje (uživatelé to u webových aplikací vyžadují). V tomto případě není prevencí ani použité odesílání parametrů metodou POST. Tyto parametry je možné pomocí JavaScriptu snadno odchytit. Je možné se bránit jen pomocí přidání dalších bezpečnostních opatření. Těmi může být potvrzování operací pomocí CAPTCHA nebo pomocí SMS. Pak je možné neprovádět operace jen přímo z prohlížeče. Tím docílíme toho, že data není možné jednoduše podvrhnout. Pro zabránění Cross Site Request Forgery u ostatních aplikací je důležité používat metody, které při zobrazení formuláře uživatelem vygenerují náhodný kód. Může se jednat o neviditelné prvky a obsah těchto prvků je možné očekávat dalším odesláním formuláře. Tento kód by měl být náhodný a pro každého uživatele jiný. Toho jde docílit například při registraci uživatele vytvořením semínka, které se bude přidávat při generování ověřovacího kódu. Pokud máme kód vygenerovaný, je možné po odeslání formuláře tento kód na zkontrolovat na straně serveru. Tento kód smí být shodný po celou dobu platnosti sezení. Pokud je to možné, je dalším řešením problému CSRF mít co nejkratší platnost sezení [6].

1.2.6 JavaScript Hijacking

S rozvojem aplikací používajících JavaScript se objevil i nový typ útoku nazvaný JavaScript Hijacking. Týká se aplikací používajících JavaScript pro výměnu dat, např. pomocí formátu JSON. Pokud aplikace nabízí uživateli přenos dat ve formě JSON, sezení je pravděpodobně uloženo v cookies a obsah se tedy bude měnit dle cookies a dle uživatele, je možné, že se tyto data stanou terčem útoku. K útoku dojde, pokud je uživatel právě přihlášen a navštíví stránku se škodlivým kódem. Tento útok je vlastně odposlouchávání dat a jejich možná modifikace, ztráta nebo zneužití [7].

1.2.7 Clickjacking (Click Hijacking)

Tento útok využívá nepozornosti uživatele. Uživateli se zobrazí stránka, ve které jsou elementy rozmístěny jako na obyčejné stránce. To znamená, že se zde nachází nějaký obsah a tlačítko. Tlačítko může pocházet z jiné aplikace, ale tváří se tak, že je umístěno pomocí DIVU nebo IFRAMU ve stránce. Uživatel o tom však nemá ponětí a považuje je za součást stránky, kterou vidí. Tato podvodná stránka, která je uživateli zobrazena, dává tomuto tlačítku úplně jiný význam. Jedná se ale stále o původní tlačítko. To vykonává stále původní

činnost, a to i přesto, že je aplikace zabezpečená proti CSRF útokům. Jedná se totiž o případ, kdy uživatel tuto akci potvrdil. Jedním z řešení je nepovolit vložení aplikace do IFRAME nebo nespouštění kontroly na jiném serveru. Je toho možné dosáhnout JavaScriptovou kontrolou neznámých elementů na stránce [7].

1.3 Zabezpečení - SSL

Secure Sockets Layer, SSL je protokol (vrstva) vložený mezi vrstvu transportní (TCP/IP) a aplikační (HTTP), který poskytuje zabezpečení komunikace šifrováním a autentizací komunikujících stran. Následovníkem SSL je Transport Layer Security (TLS). Protokol SSL se nejčastěji využívá pro bezpečnou komunikaci s webovými servery pomocí HTTPS, což je zabezpečená verze protokolu HTTP. Po vytvoření SSL spojení (session) je komunikace mezi serverem a klientem šifrovaná, a tedy zabezpečená. Ustavení SSL spojení funguje na principu asymetrické šifry, kdy každá z komunikujících stran má dvojici šifrovacích klíčů. Jedná se o veřejný a soukromý klíč. Veřejný klíč je možné zveřejnit a pokud tímto klíčem kdokoliv zašifruje nějakou zprávu, je zajištěno, že ji bude moci rozšifrovat jen majitel použitého veřejného klíče svým soukromým klíčem. Během první fáze ustanovení bezpečného spojení si klient a server dohodnou kryptografické algoritmy, které budou použity [7].

1.4 Databáze

Databáze je systém určený k organizování, ukládání a načítání velkého množství dat. Skládá se z organizovaného sběru dat pro jedno nebo více použití, nejčastěji v digitální podobě. Jeden ze způsobů, jak třídit databáze zahrnuje typ jejich obsahu. Databáze mohou být bibliografické, nebo statistické. Digitální databáze jsou spravovány pomocí systémů řízení databází, které ukládají obsah databáze a umožňují vytváření a údržbu dat, vyhledávání a jiný přístup [8].

1.4.1 Systém řízení báze dat (SŘBD)

SŘBD je sada počítačových programů, které řídí vytváření, údržbu a použití databáze. Dále je to systém, který pomáhá využití integrovanému sběru dat záznamů a souborů, známý jako databáze. To umožňuje uživateli přes různé aplikační programy snadný přístup ke stejné databázi. SŘBD může použít kterýkoli z databázových modelů, jako je například síťový model nebo relační model. Ve velkých systémech, SŘBD umožňuje uživatelům i

jiný software k ukládání a načítání dat strukturovaným způsobem. Místo toho, aby musel psát počítačové programy k získání informací, může uživatel klást jednoduché otázky v dotazovacím jazyku [8].

1.4.2 Relační databáze

Základním konstruktorem relačních databází jsou relace (databázové tabulky), což jsou dvourozměrné struktury tvořené záhlavím a tělem. Jejich sloupce se nazývají atributy, řádky tabulky jsou pak záznamy. Atributy mají určen svůj konkrétní datový typ a doménu, což je množina přípustných hodnot daného atributu. Řádek je řezem přes sloupce tabulky a slouží k vlastnímu uložení dat. Pojem „relační databáze“ souvisí s teorií množin. Každá konkrétní tabulka totiž realizuje podmnožinu kartézského součinu množin přípustných hodnot všech sloupců [8].

1.4.3 MySQL

MySQL je multiplatformní databáze. Komunikující pomocí dotazovacího jazyka SQL. Podobně jako u ostatních SQL databází se jedná o dialekt tohoto jazyka s některými rozšířeními. Pro svou snadnou implementovatelnost, výkon a především díky tomu, že se jedná o volně šiřitelný software, má vysoký podíl na v současné době používaných databázích. Velmi oblíbená a často nasazovaná je kombinace Linux, MySQL, PHP a Apache jako základní software webového serveru. MySQL bylo od počátku optimalizováno především na rychlost, a to i za cenu některých zjednodušení. Jednalo se hlavně o jednoduché způsoby zálohování, a až donedávna nepodporovalo pohledy, triggeru a uložené procedury [8].

1.5 PHP Framework

Framework je softwarová struktura, která slouží jako podpora při programování, vývoji a organizaci jiných softwarových projektů. Může obsahovat podpůrné programy, knihovny, podporu pro návrhové vzory nebo doporučené postupy při vývoji [9].

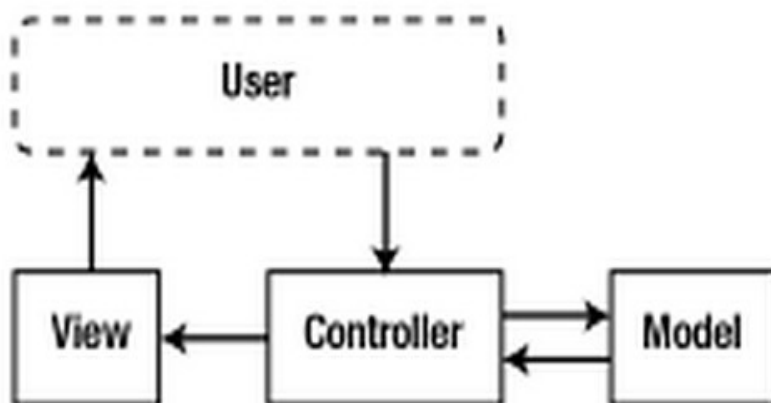
1.5.1 CakePhp

Filosofií tohoto frameworku je jednoduchost. Inspiruje se v RoR (Ruby on Rails - je framework pro vývoj webových aplikací napojených na databázi). Dobře navrhnuté základní konvenční chování, ale přesto výrazná variabilita. Framework se zaměřuje především na přehledný a krátký zdrojový kód, kvalitní databázovou vrstvu typu active record, promyš-

lené view helpery a pokrývá běžné operace, které programátor potřebuje. Pro vývoj standardního CMS či e-shopu je to velice vhodný nástroj. CakePHP lze velice snadno integrovat se Zend Frameworkem a v této kombinaci je výborný, protože kombinuje svou jednoduchost a přímočarost se silou Zendu [10].

1.6 MVC

Model-View-Controller je softwarová architektura, která rozděluje datový model aplikace, uživatelské rozhraní a řídicí logiku do tří nezávislých komponentů tak, že modifikace některého z nich má jen minimální vliv na ostatní. MVC je často chápán jako návrhový vzor, nicméně se týká architektury aplikací mnohem více než klasický návrhový vzor. Vytváření aplikací s využitím architektury MVC vyžaduje vytvoření tří komponentů. Model, což je doménově specifická reprezentace informací, s nimiž aplikace pracuje. View (pohled), který převádí data reprezentovaná modelem do podoby vhodné k interaktivní prezentaci uživateli. Controller (řadič), který reaguje na události (typicky pocházející od uživatele) a zajišťuje změny v modelu nebo v pohledu. Komponenty řadič a pohled jsou ve standardním rozdělení vrstev na prezentační, doménovou a datovou obvykle zařazovány jako prezentační vrstva, jak je vidět na obrázku číslo 4 [15]. V MVC je tato prezentační vrstva rozdělena mezi komponenty řadič a pohled, nicméně nejdůležitější rozdělení je mezi prezentací a doménovou vrstvou [10].



Obrázek 4. Model-View-Controller Diagram

1.7 Věda a výzkum

Vědecká, výzkumná, vývojová a publikační činnost je velmi důležitou a specifickou součástí činností zajišťovaných výzkumnými skupinami. Poskytuje prostor pro tvůrčí základní i aplikovaný výzkum, vývoj prototypů, národní a mezinárodní spolupráci, je nezbytná pro vědecký a odborný růst, je podmínkou habilitací docentů, jmenování profesorů a obhájení disertací studentů doktorského studijního programu. Je rovněž významným ekonomickým zdrojem zajišťujícím rozvoj [11].

1.7.1 Výzkum dle funkce a využití výsledků v praxi

Základní výzkum, který se orientuje na řešení klíčových teoretických problémů, usiluje o hlubší odhalení vnitřní povahy jevů, o objasnění jejich příčin s cílem obohatit dosavadní vědění. Nezabývá se přímo společenskou využitelností výsledků. Jako další rozdělení je zde aplikovaný výzkum. Ten vzniká z potřeby zkoumat i řešit praktický problém, případně získat informace o problémovém jevu. Cílem je hledat cesty a způsoby, jak využít vědecké poznatky získané základním výzkumem v praxi. Poslední rozdělení je metodologický výzkum. Jeho cílem je ověřovat stávající a hledat nové metody a techniky, vhodnost jejich použití pro různé druhy výzkumů, zjišťovat jejich spolehlivost [11].

1.7.2 Výzkum dle realizace výzkumu

Výzkum se realizuje jako individuální. To znamená, že výzkumník pracuje sám. Dále je možné výzkum realizovat jako skupinový (týmový). Tedy větší i menší skupina výzkumníků. Zkušební způsob realizace slouží jako předvýzkum. Používá se k ověřování vhodnosti a validity výzkumné metodiky. Experimentální výzkum je pokusné zjišťování působení určitých faktorů v přesně vymezeném procesu a komparativní (srovnávací) se používá na základně výsledků z minimálně dvou výzkumů, z nichž je provedena komparace získaných údajů [11].

1.7.3 Výzkum dle použité metodologie

Jedná se buď o kvalitativní výzkum. To znamená že se pracuje s malým souborem dat bez nároku na statistickou reprezentativnost. Analýza dat se provádí vyhodnocováním jednotlivých případů aplikací metod kvalitativní analýzy. Druhou metodologií je použití kvantitativního výzkumu. Jeho úkolem je statisticky popsat typ závislostí mezi proměnnými a změnit intenzitu této závislosti. Pracuje s větším souborem dat [11].

2 REŠERŠE STÁVAJÍCÍCH ŘEŠENÍ

2.1 Informační systémy pro výzkumné skupiny

Pod tímto specifickým označením informačního systému se rozumí systém pro podporu spolupráce lidí v týmu na společném projektu. Pro usnadnění spolupráce a lepší komunikaci se používají nástroje, jako jsou elektronická pošta, která slouží ke komunikaci uživatelů nebo oznamování změn v projektu, diskuzní fóra, kde uživatelé zadávají své poznatky a řeší problémy dalších uživatelů, chaty pro rychlejší komunikaci a vyřešení problému dřív, je zde možné dotazy směřovat na konkrétního uživatele či pořádat konference více uživatelů, kalendáře, které pomáhají k časové organizaci a nastavení upozornění na konce řešení úkolů či projektů a samozřejmě nástroje na sdílení a organizaci dokumentů na takzvané úložiště. Pracovníci výzkumné skupiny jsou tým a jako tým jsou i vedoucím hodnoceni a proto je důležité aby pracovníci mohli pracovat i na dálku.

Každý společný úkol nebo úloha může být rozdělena na více závislých nebo nezávislých částí. Pracovníci pracující na takovém úkolu jsou povinni v případě závislých částí spolupracovat a spojovat a konzultovat dané části s ostatními. Rychlá komunikace může být klíčem k rychlému řešení daného problému. Další důležitou vlastností takových systémů je sdílení dat a souborů. K tomu mohou sloužit například systémy pro správu dokumentů (Document Management System). Díky takovému subsystému smí uživatelé spravovat nejen své soubory a dokumenty, ale mohou editovat i ty, které vytvořili jejich kolegové. Propracovanější subsystémy, které se používají pro správu dokumentů, dokáží nejen řešit přístup uživatelů k jednomu souboru, ale řeší i přístup dle role, rozdělují uživatele do skupin a řeší i archivaci a správu verzí.

Každý takový systém má další podpůrné prostředky pro co nejlepší řízení týmu nebo výzkumných skupin. Mezi ně patří například oběh dokumentů (workflow), které vyhodnotí počítač pomocí definovaných pravidel. Jedná se o automatické archivování, zamítání požadků či mazání. Vhodné je i mít možnost přistupovat přes internet a dovolovat uživatelům mobilitu. Výhodou mohou být i konference a sdílené adresáře.

Tyto systémy nejvíce využívají pracovní týmy a skupiny pracující na jednom nebo více propojených projektů, dálkově pracující skupiny, virtuální týmy ve virtuálních organizacích, studenti na školních projektech, týmy v zájmové činnosti [16].

2.2 Mavenlink

Mavenlink je software založený na cloudovém základu a díky tomu je nezávislý na operačním systému a slouží pro spolupráci týmu a pro správu projektů. V některých případech se dá použít jako software pro profesionální automatizaci služeb. Software byl vyvinutý v USA a díky SaaS (Software jako služba) platformě nabízí svým klientům profesionální řešení k efektivnímu řízení projektů. Mavenlink obsahuje aplikace pro sdílení souborů, správu dokumentů, správu úloh, plánování zdrojů, sledování milníků, času a výdajů a online fakturace. Tento systém používá 256-bitové SSL šifrování k zajištění bezpečnosti komunikace a ochrany dat jejich uživatelů. Webová adresa, kde je možné tento software vyzkoušet je <http://www.mavenlink.com>. Systém poskytuje deseti denní trial verzi na vyzkoušení funkcí. Je možné si také stáhnout mobilní aplikaci na Google Apps. Další výhodou je integrace aplikací od Googlu jako je Gmail, Google Docs, Google Calendar, Google Tasks. Firma software nabízí ve dvou verzích. První verze nazývaná se TEAMS je za 48 dolarů ročně za uživatele nebo 6 dolarů za uživatele měsíčně. Tato verze neobsahuje ekonomická odvětví jako Finanční řízení, Řízení zdrojů a Analýzy a výkazy. Druhá verze nazývaná se Premier je dražší a předplatné pro uživatele se pohybuje od 11 do 49 dolarů měsíčně za jednoho uživatele v závislosti na velikosti společnosti a počtu licencí.

2.2.1 Výhody

- Sdílení souborů
- Sdílení úkolů
- Plánování
- Management zdrojů

2.2.2 Nevýhody

- Cena
- Přehlednost aplikace
- Jazykové mutace
- Demo pouze jako trial verze – nutnost se registrovat
- Není Open Source
- Pouze dvě verze programu (podle ceny)

- Data pouze na jejich serverech – není možné mít lokální kopii

2.3 Plancake

Plancake je Open Source projekt, který je možné si provozovat i na svém webovém serveru. Aplikace je postavena na frameworku Symfony, je psaná v PHP5 a využívá databázi MySQL. Pokud si uživatelé nechtějí instalovat celý webový server, je možné si vytvořit účet na serveru <http://www.plancake.com/> a po úspěšné registraci zvolit typ aplikace, který chtějí uživatelé využívat. Na výběr je verze zdarma, která umožňuje uživateli využívat maximálně 500 aktivních úkolů ve 20 seznamech. Je zde možné použití 5 tagů pro organizování úkolů a maximální záloha dokončených úkolů jsou dva měsíce. Dále aplikace umožňuje plánovat opakující se úkoly, využívat mobilní aplikaci a velké množství reportů. Druhou možností je placená verze, která uživatele vyjde na 23,90 euro za rok. Oproti verzi zdarma obsahuje prodlouženou dobu archivace dokončených úkolů na 6 měsíců. Placená verze navíc nabízí neomezený počet úkolů, seznamů, tagů a umožňuje export do Excelu. Dále je v této verzi kladen důraz na bezpečnost, protože verze nabízí lepší zabezpečení přes SSL, emailové připomenutí na blížící se konec úkolů a vyšší prioritu v případě pomoci s aplikací nebo některým dotazem.

2.3.1 Výhody

- Open Source
- Funguje na počítačích, tabletech nebo mobilech
- Možnost offline režimu
- Getting Things Done (řízení pracovního procesu) organizace práce
- Jednoduchost

2.3.2 Nevýhody

- Sdílení úkolů mezi uživateli
- Nepodporuje komentáře
- Cena za neomezenou verzi
- Není možné přidávat uživatele
- Chybí čeština

2.4 Flyspray

Flyspray je jednoduchý, webově založený systém pro zadávání úkolů a díky tomu je na platformě nezávislý. Tento systém je napsaný v PHP a podporuje více databází. V současné době jsou to databáze MySQL a PGSQL. Flyspray je svobodný software, šířený pod GNU Lesser General Public Licence verze 2.1. To znamená, že každý uživatel může získat Flyspray a používat jej zdarma, ať už pro týmovou spolupráci nebo pro účely výzkumu. Tento software nenabízí možnost se přímo registrovat a používat software. Vždy je nutné rozhodit software a zaplatit si vlastní hosting, nebo užívat software jen v lokální síti. Zdrojový kód je k dispozici na adrese <http://flyspray.org/download>. Software se vyznačuje jednoduchou instalací a snadným používáním. Flyspray dovoluje vytvářet více projektů a je možné nastavit sledování úkolu, které je řešeno přes email nebo Jabber. Díky tomu jsou přiřazení uživatelé informováni o všech změnách souvisejících s úkolem. Dalšími vlastnosti, které aplikace podporuje jsou kompletní historie úkolů, vkládání souboru do komentářů a možnost výběru z předdefinovaných vzhledů. Mezi bonusové funkce, které Flyspray implementuje jsou grafy závislostí, hlasování pro úkoly a pokročilé vyhledávací funkce.

2.4.1 Výhody

- Jednoduchý vzhled
- Webová aplikace
- Možnost přiřadit více uživatelů
- Cena
- Open Source
- Více jazykových mutací

2.4.2 Nevýhody

- Nevyvíjí se – poslední release 28. května 2012
- Pluginy
- Je nutná instalace webového serveru
- Nemá češtinu

2.5 Mantis Bug Tracker

Mantis Bug Tracker je bezplatný, open source, webový systém pro sledování chyb, šířen pod podmínkami licence GNU General Public verze 2. Nejvíce obvyklé použití MantisBT je řízení projektů a sledování vývoje a řešení úkolů. MantisBT však díky vestavěným funkcím může být uživateli nakonfigurován jako sledovací systém a nástroj. Název Mantis a logo projektu je odvozeno od latinského názvu Kudlanky nábožné (Mantis religiosa) pocházejícího z řad kudlanek (Mantodea), které jsou známe pro sledování a požívání jiných druhů hmyzu. Toto má souvislost s tím, že často se říká chybám v softwaru bug, což znamená v říši zvířat brouk. Název projektu je obvykle zkrácen jen na MantisBT nebo Mantis. Systém poskytuje 30 denní trial verzi pro vyzkoušení funkcí. Po jejím vypršení nabízí uživatelům tři varianty placených verzí. Ty jsou pojmenovány dle hodnoty cených kovů. První verze jejíž název je Bronze, je limitovaná počtem projektů na jeden, stojí 14,95 dolarů měsíčně a její výhodou je že umožňuje neomezený počet uživatelských účtů. Nemá však podporu oznamování emailem a možnost stáhnout mobilní aplikaci. Take maximální diskový prostor je jen 200 megabajtů. Druhá verze aplikace se jmenuje Silver a limitovaná je jen maximálním počtem projektu a místem na disku. Těch může být při zaplacení 19,95 dolarů měsíčně až 10 s omezením diskového místa na 2 gigabajty. Za tuhle cenu tým získá plnou podporu oznamování emailem a mobilní aplikaci. V případě že i tato verze nevyhovuje je možné zakoupit za 24,95 dolarů měsíčně verzi, která je limitovaná jen maximálním diskovým prostorem a to je 4 gigabajty.

2.5.1 Výhody

- GNU General Public License
- Jednoduchá instalace
- Lokalizace
- Mobilní aplikace
- Pluginy

2.5.2 Nevýhody

- Zastaralý vzhled
- Cena hostingu aplikace

II. PRAKTICKÁ ČÁST

3 NÁVRH

Cílem projektu je vytvořit systém pro správu projektů ve výzkumné skupině. V systému budou existovat role typu SuperAdmin, ProjectAdmin, TeamLeader a Worker. Pro každou roli bude platit jiná množina nastavení. Nejvyšší autoritou bude SuperAdmin, který má právo nahlížet do všech projektů, spravovat uživatele, přiřazovat je k projektům, vytvářet podprojekty a úkoly. ProjectAdmin bude role, která bude moct spravovat a přiřazovat ke svým projektům uživatele a vidět stav jejich práce. TeamLeader má omezená práva na vytvořený tým programátorů a spravuje jen lidi přiřazené k projektu. Worker je základní role uživatelů. Mohou jen přidávat úkoly, měnit stav úkolů, zavírat úkoly, ale ne celé projekty.

3.1 Software pro vytvoření návrhu

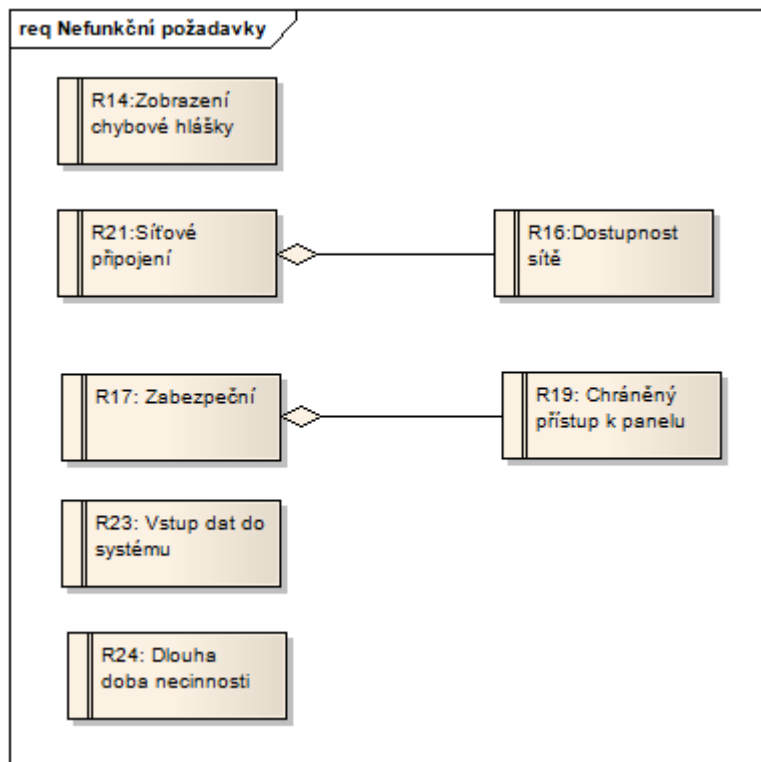
Pro vytvoření návrhu byl použit program Enterprise Architect od společnosti Sparx Systems je kompletní nástroj pro systémovou analýzu a návrh, který pokrývá celý životní cyklus vývoje systému. Je schopné vytvořit model všech požadavků přes analýzu stavů, návrh modelů, testování a údržbu, vše s využitím diagramů v UML.

3.2 Požadavky na systém

Strukturovaný dokument obsahující detailní popis funkcí systému. Definiující, co by mělo být implementováno. Kontakt mezi klientem a vývojářem.

3.2.1 Nefunkční požadavky

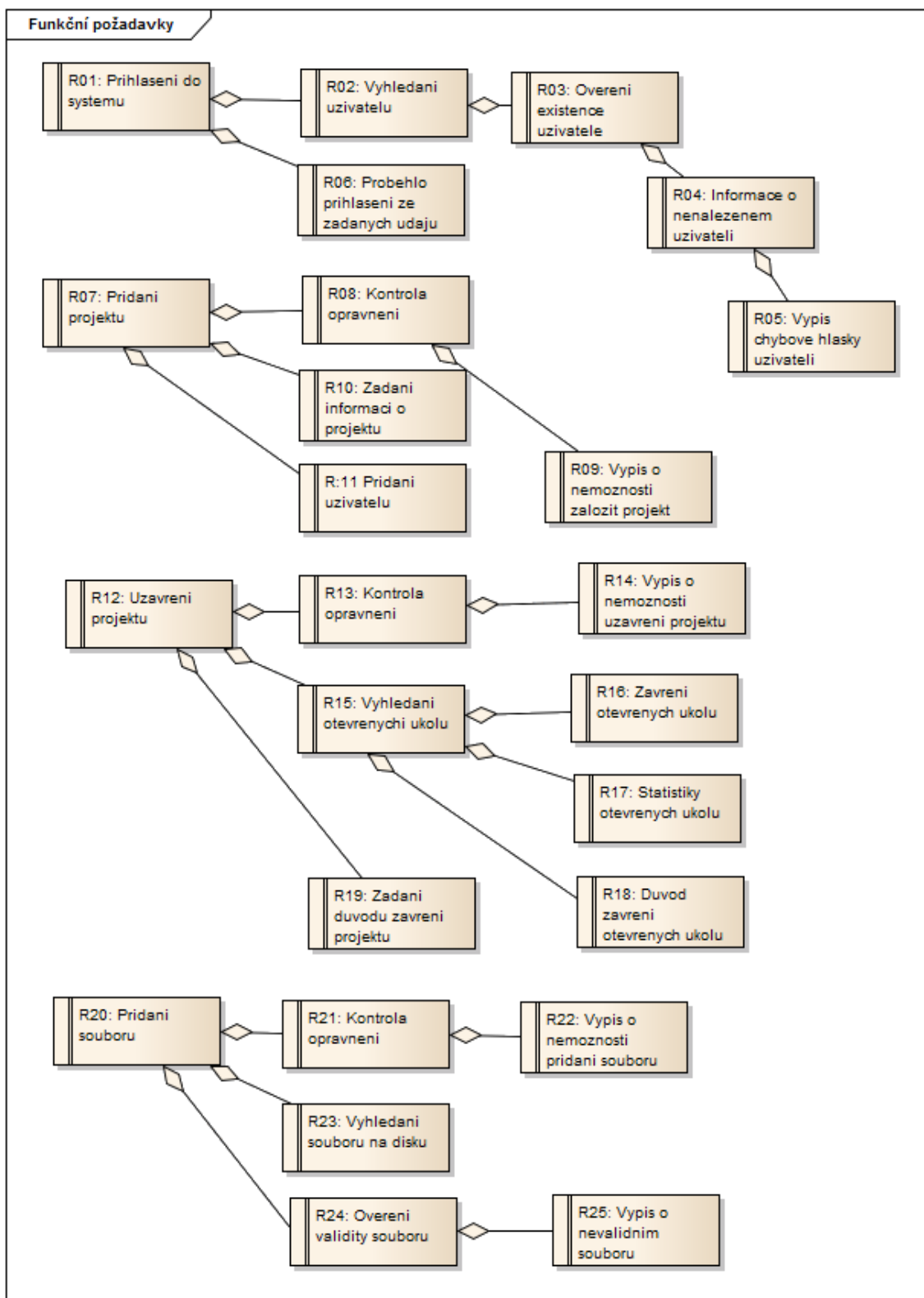
Vlastnosti a omezení služeb, jenž jsou systémem nabízeny. Jako časová náročnost, spolehlivost a bezpečnostní omezení. Často se vztahují na systém jako celek než na individuální části. Typické nároky kladené na nefunkční požadavky jsou především rychlost, velikost, jednoduchost na použití a spolehlivost [12]. Tyto informace jsou zobrazeny na obrázku 5.



Obrázek 5. Nefunkční požadavky - UML

3.2.2 Funkční požadavky

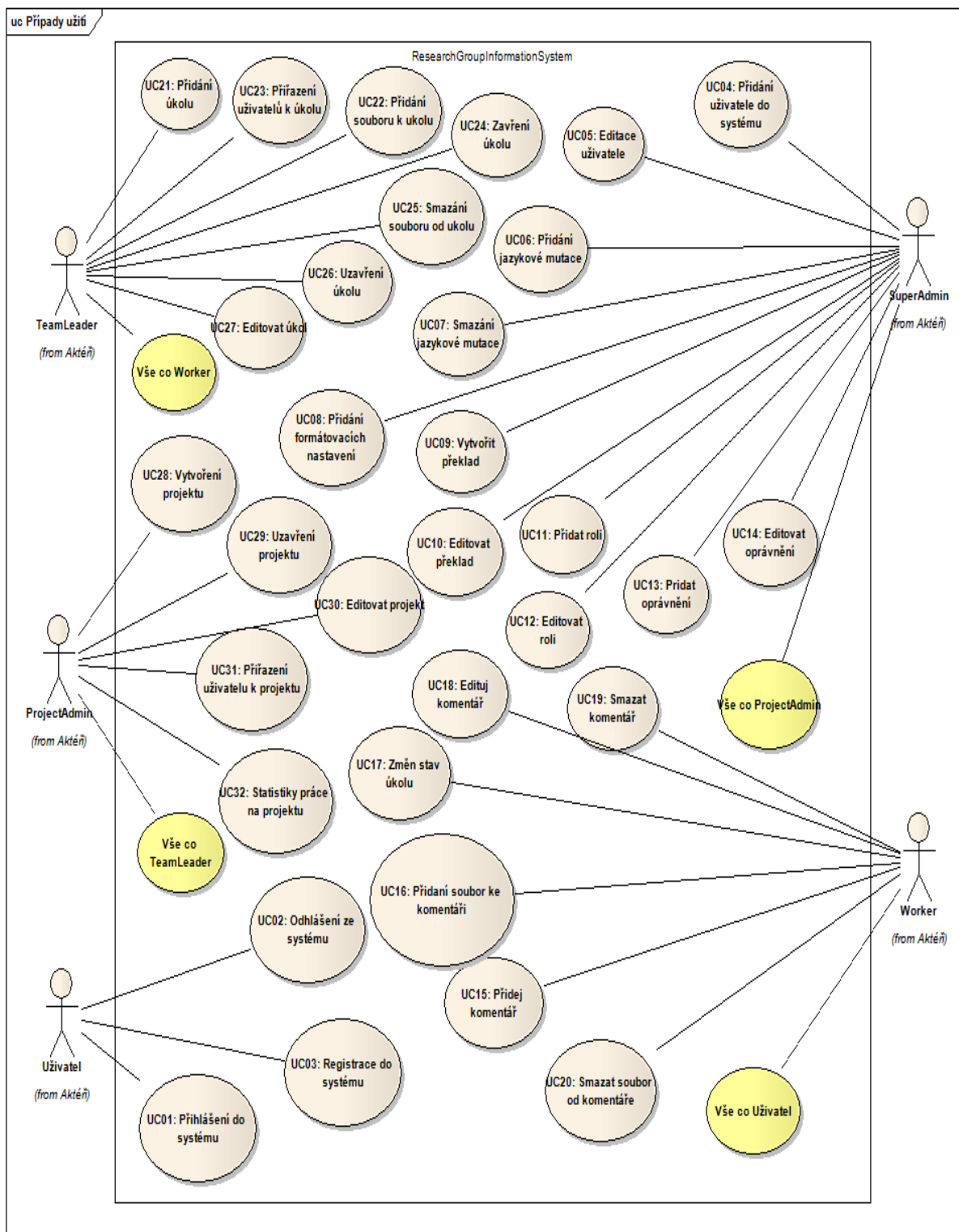
Výroky, objasňují co se musí udělat a identifikují se nutné úkony, aktivity a akce, které musí být vykonány. Analýza funkčních požadavků je použita jako základ funkce systému pro funkční analýzu [12]. Tyto informace jsou zobrazeny na obrázku 6.



Obrázek 6. Funkční požadavky - UML

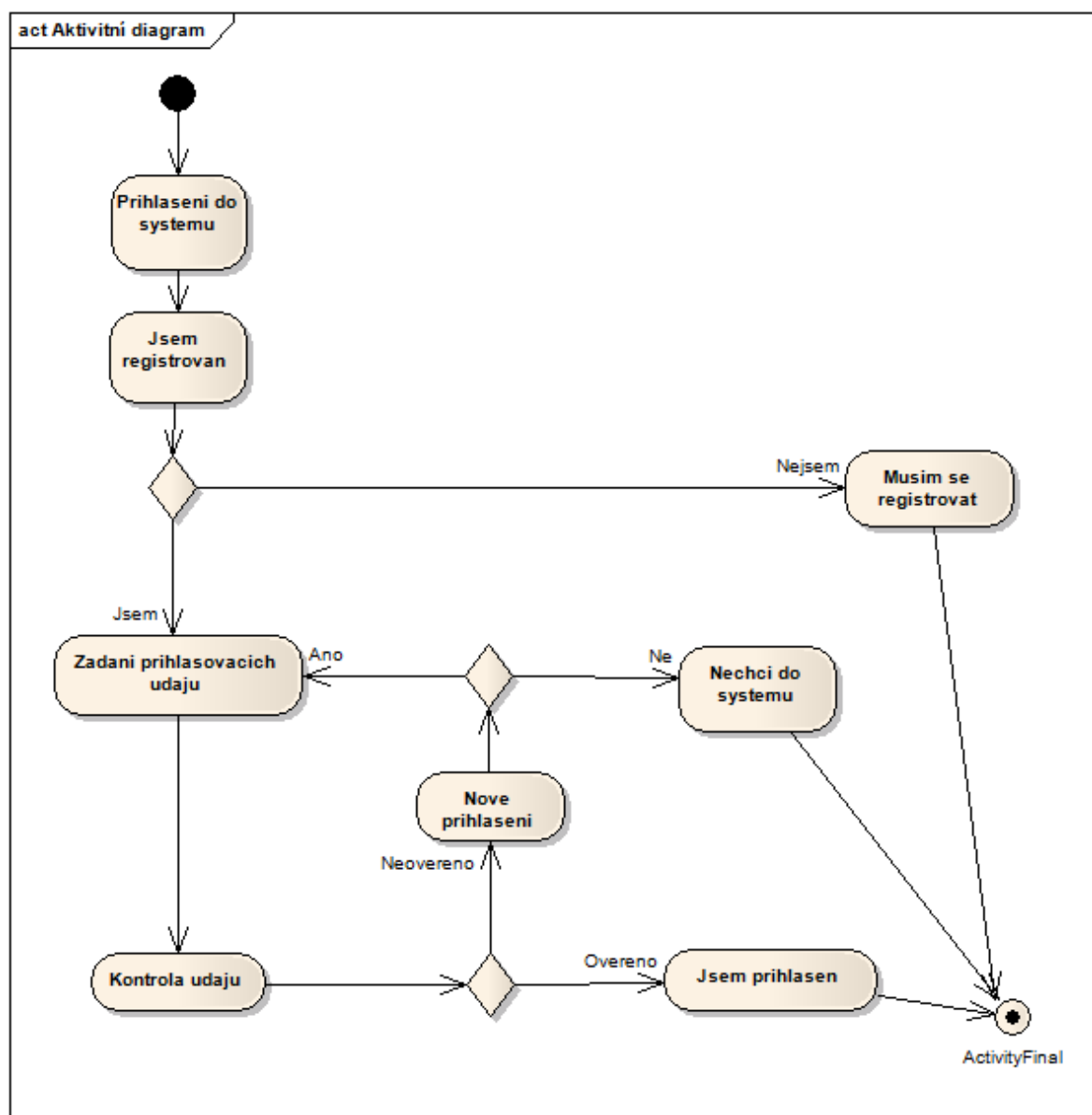
3.3 Diagram případů užití

Use case diagram nám říká, jaká je interakce mezi rolemi a systémem. Za rolí se může v tomto diagramu skrývat jak člověk tak i externí systém, jako na obrázku číslo 7.

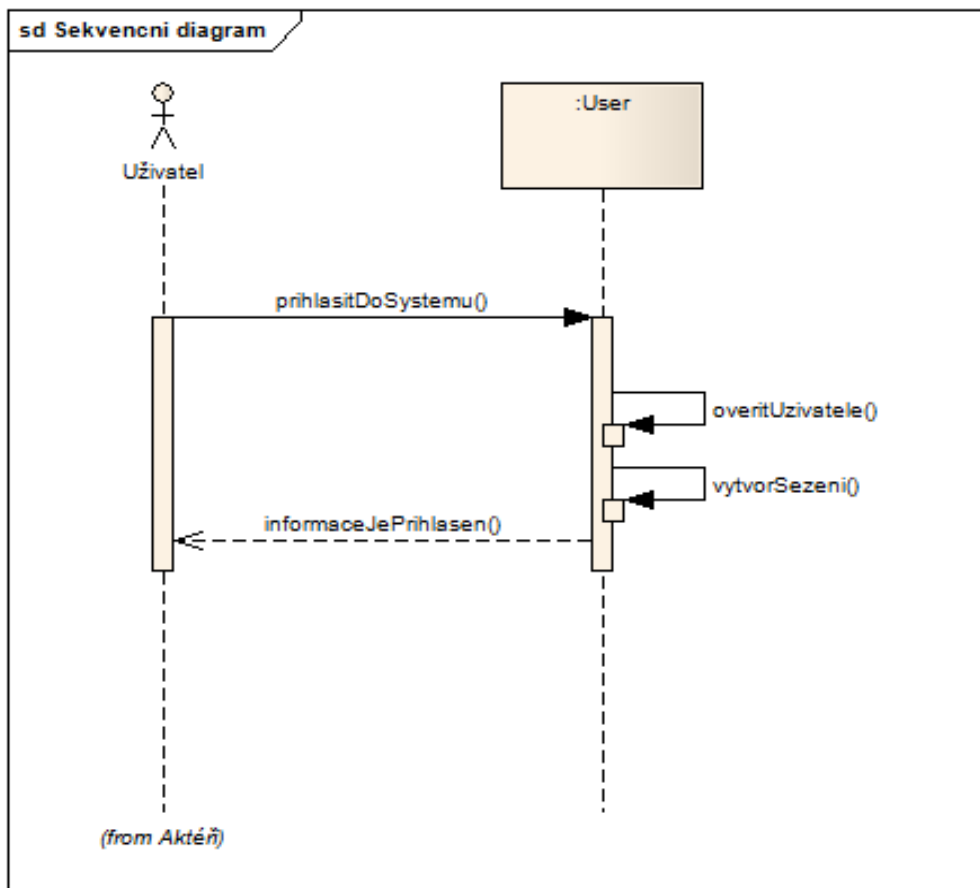


Obrázek 7. Případy užití - UML

3.4 Přihlášení



Obrázek 8. Přihlášení – Diagram aktivit - UML



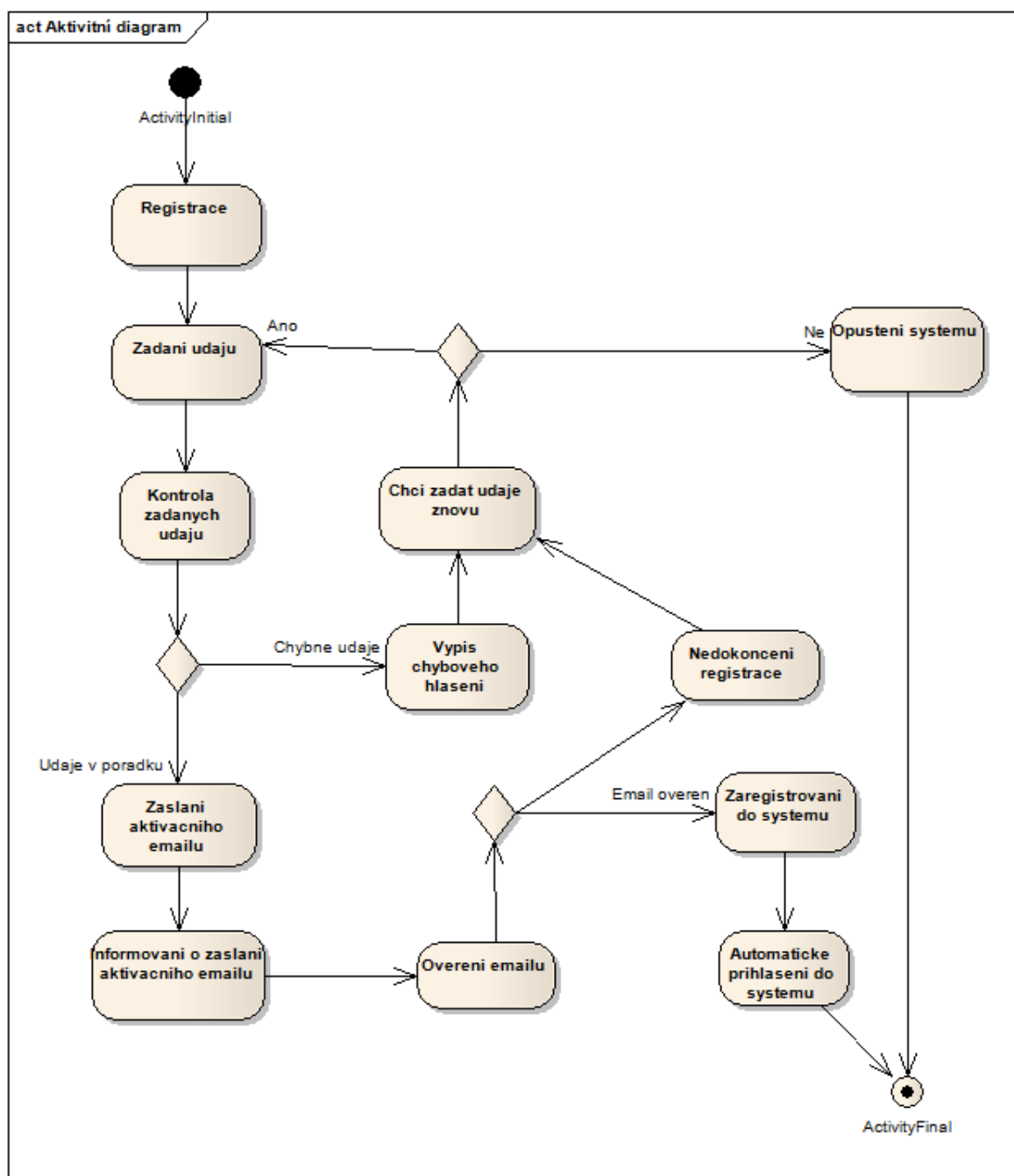
Obrázek 9. Přihlášení – Diagram sekvencí - UML

Tab. 1. Přihlášení - Scénář

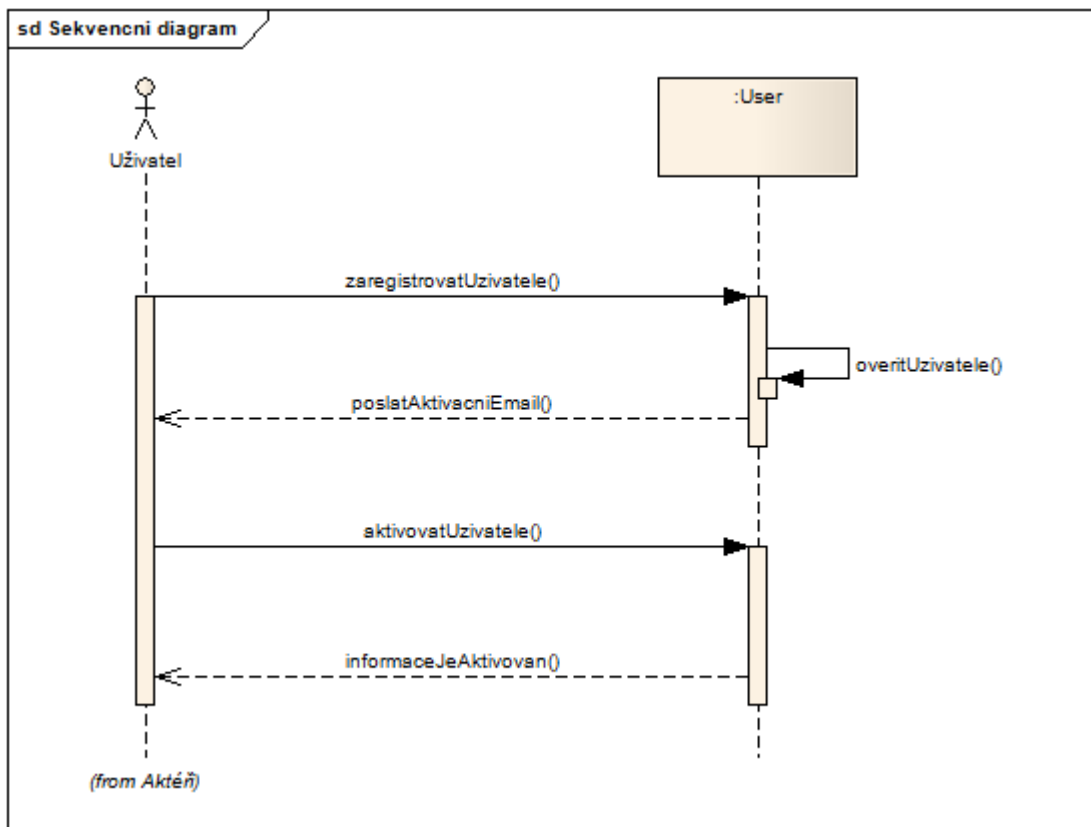
Aktér	Krok	Akce
Uživatel	1	Uživatel vyplní přihlašovací údaje
System	2	System zkontroluje z databáze existenci přihlašovacích údajů
System	3	Přihlašovací údaje existují a jsou zadány správně
Uživatel	4	Uživatel je přihlášen

Diagram tříd pro tento případ je vidět na obrázku 8, diagram sekvencí na obrázku 9 a scénář z use case diagramu je vidět v tabulce 1.

3.5 Registrace



Obrázek 10. Registrace – Diagram aktivit - UML



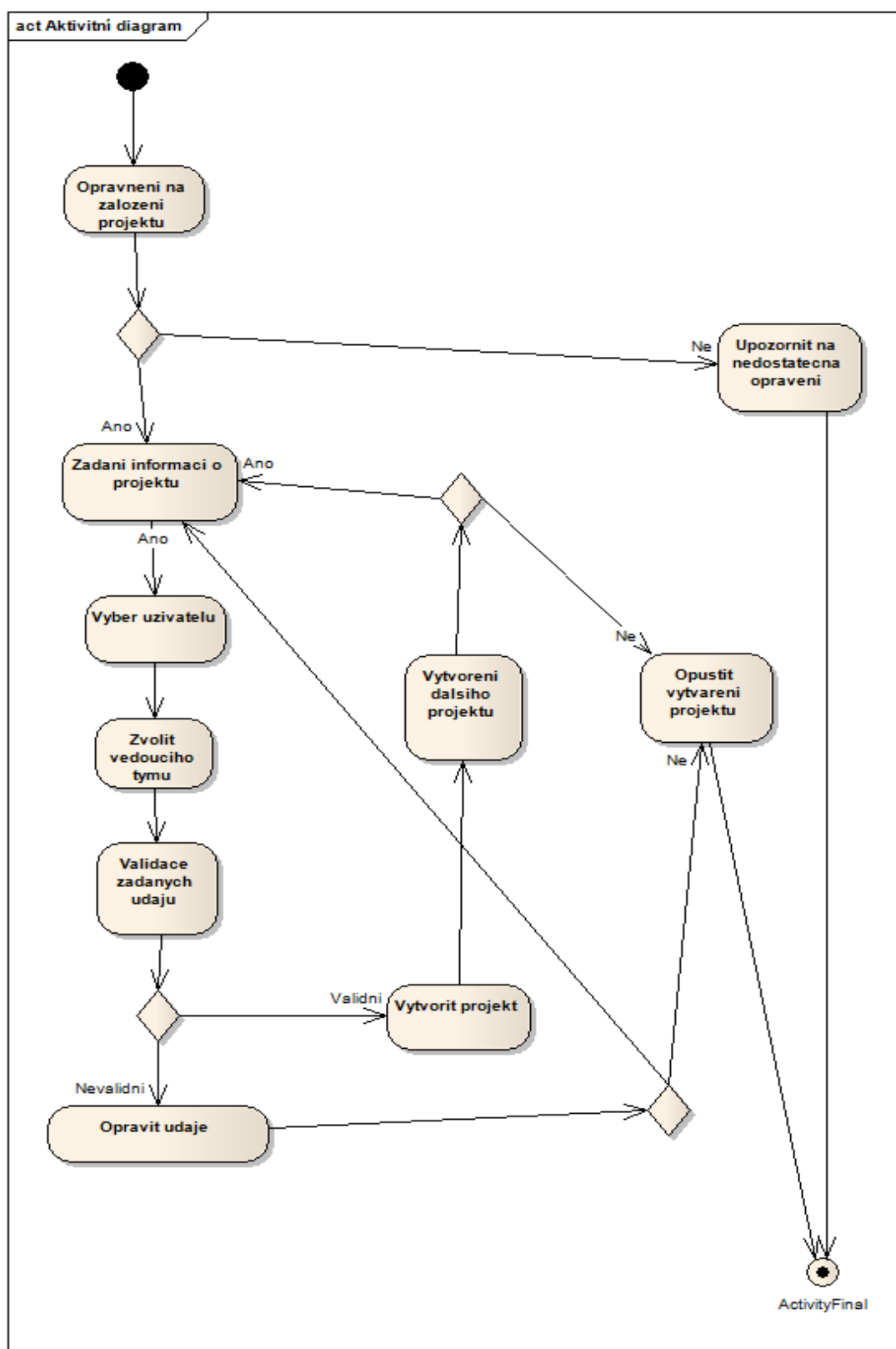
Obrázek 11. Registrace – Diagram sekvencí - UML

Tab. 2. Registrace - Scénář

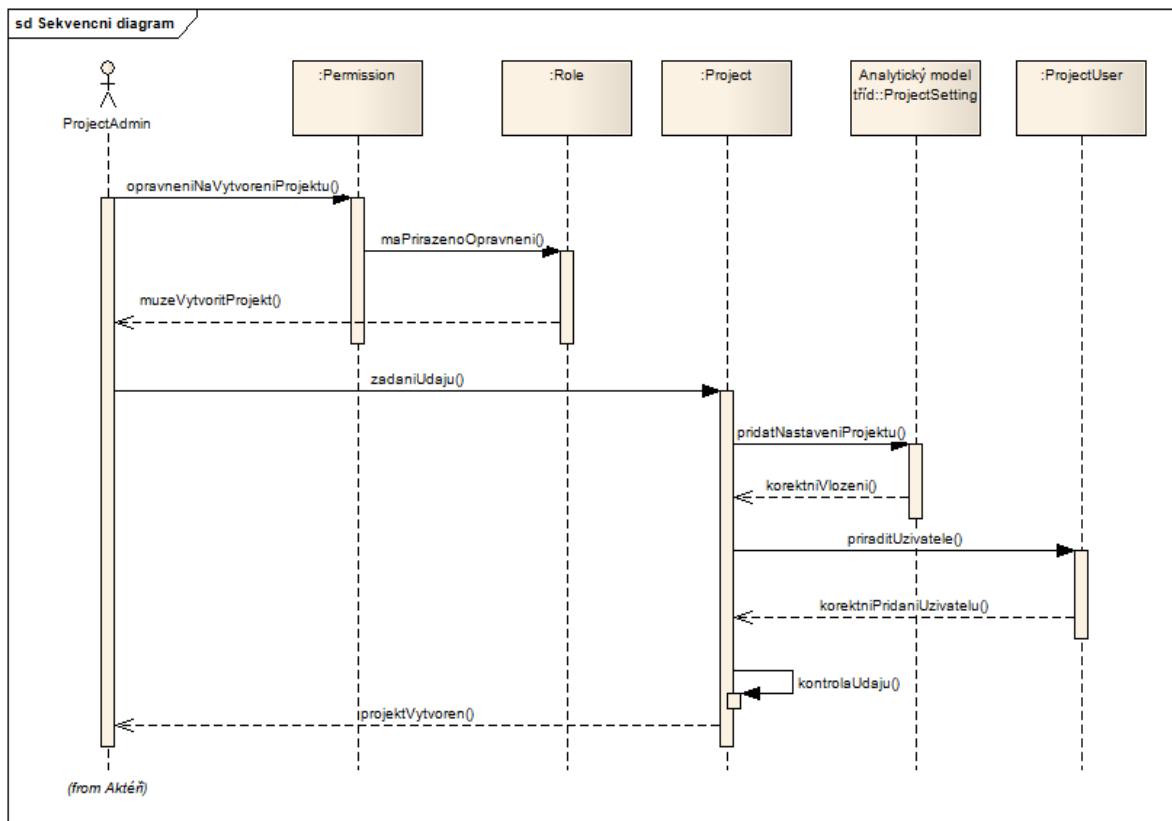
Aktér	Krok	Akce
Uživatel	1	Uživatel zadá registrační údaje
System	2	System systém ověří, zda tento uživatel v systému neexistuje
System	3	System pošle aktivační e-mail
Uživatel	4	Uživatel potvrdí aktivační e-mail
System	5	System aktivuje uživatelský účet a přihlásí uživatele

Diagram tříd pro tento případ je vidět na obrázku 10, diagram sekvencí na obrázku 11 a scénář z use case diagramu je vidět v tabulce 2.

3.6 Založit projekt



Obrázek 12. Založit projekt – Diagram aktivit - UML



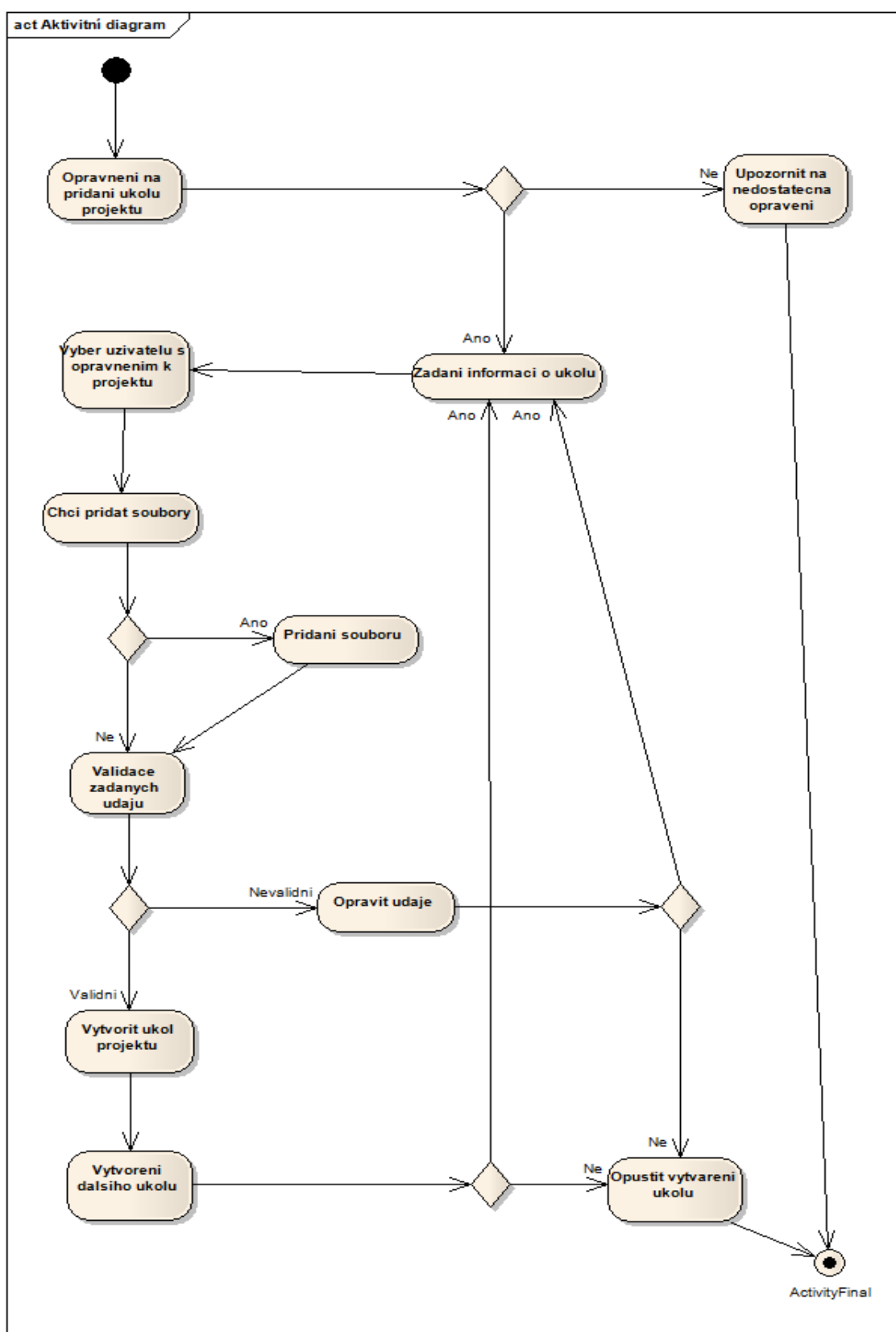
Obrázek 13. Založit projekt – Diagram sekvencí - UML

Tab. 3. Založit projekt - Scénář

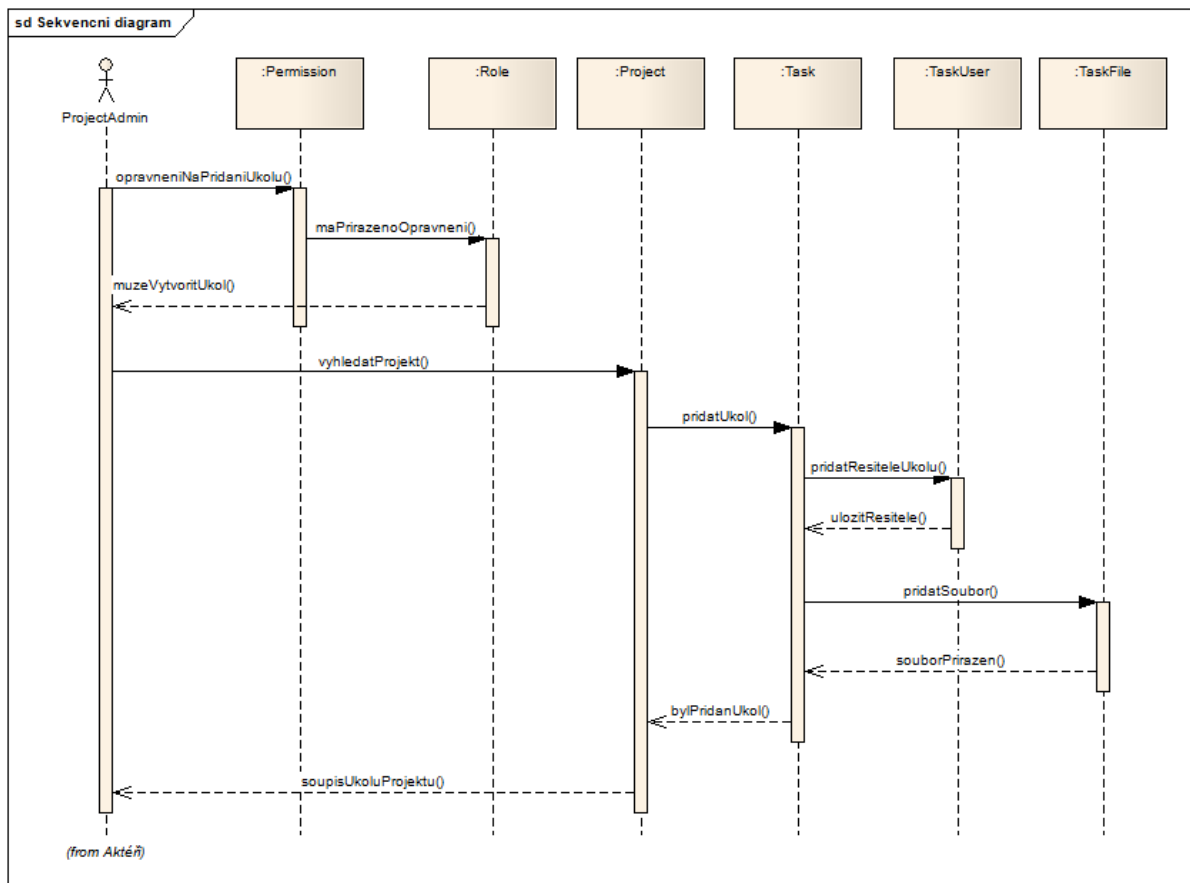
Aktér	Krok	Akce
Uživatel	1	Uživatel zvolí založit projekt
System	2	System ověří, zda má uživatel na požadavek právo
Uživatel	3	Uživatel zadá údaje o projektu
System	4	System uloží nový projekt a jeho nastavení
System	5	System přiřadí zvolené uživatele k projektu

Diagram tříd pro tento případ je vidět na obrázku 12, diagram sekvencí na obrázku 13 a scénář z use case diagramu je vidět v tabulce 3.

3.7 Přidat úkol



Obrázek 14. Přidat úkol – Diagram aktivit - UML



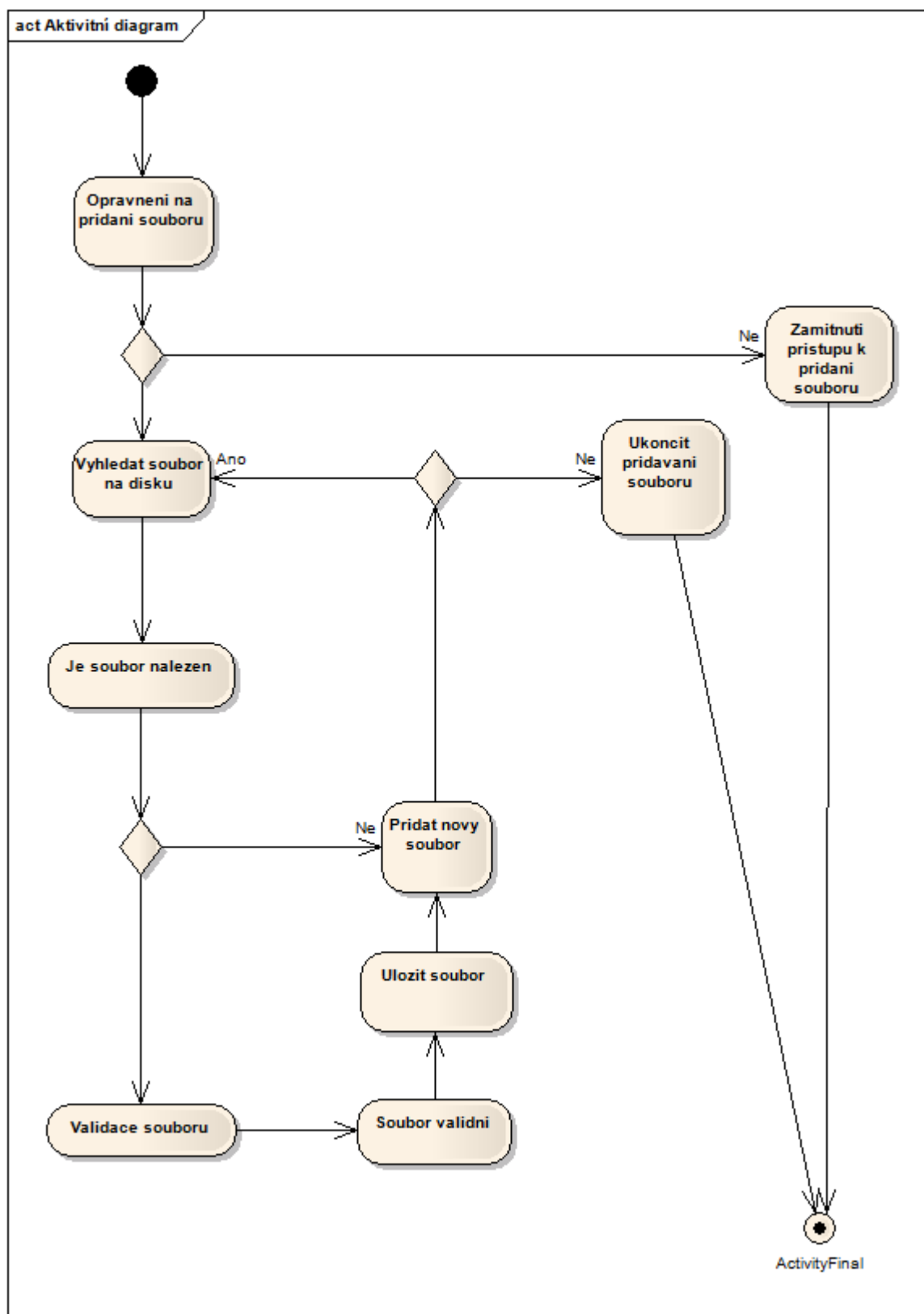
Obrázek 15. Přidat úkol – Diagram sekvencí - UML

Tab. 4. Přidat úkol - Scénář

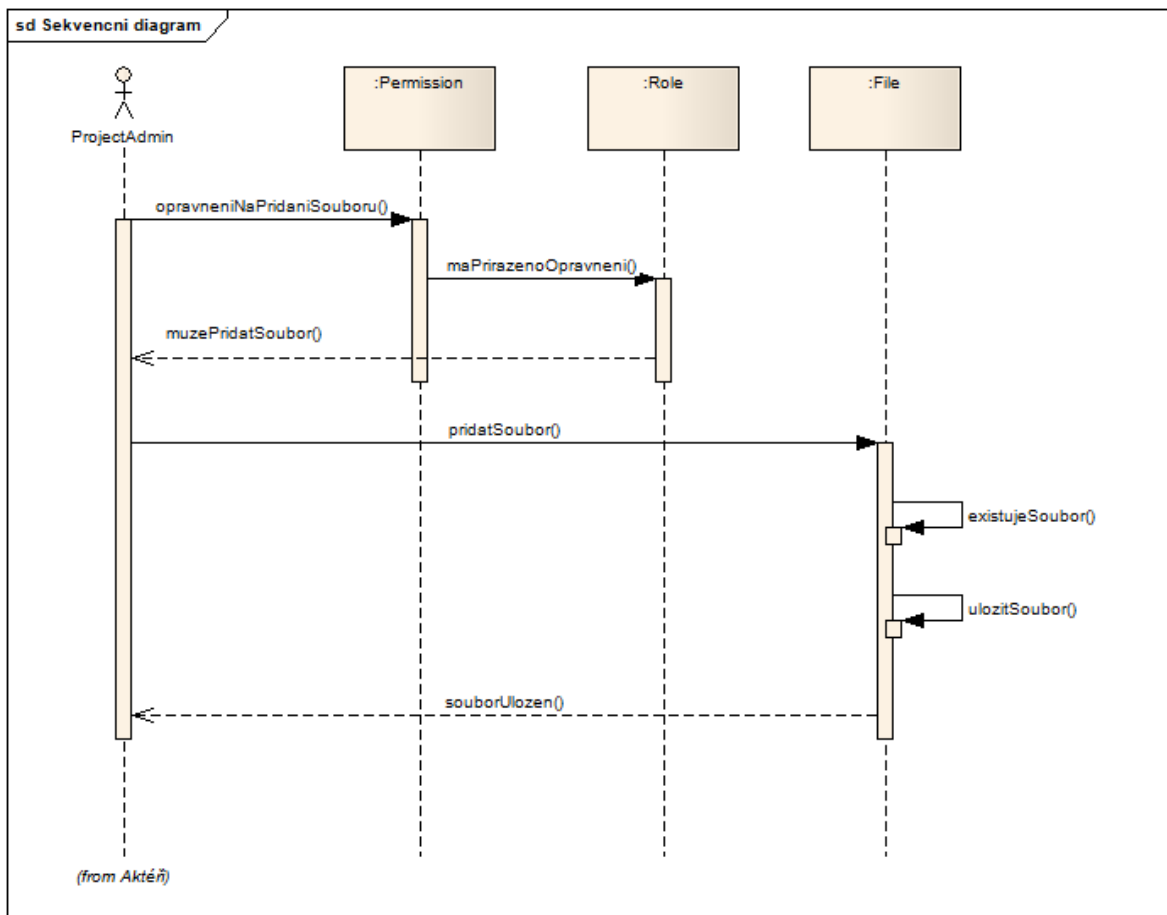
Aktér	Krok	Akce
Uživatel	1	Uživatel zvolí přidání úkolu
System	2	System ověří oprávnění uživatele
System	3	System vrátí seznam projektů, ke kterým má uživatel oprávnění
Uživatel	4	Uživatel vybere projekt
Uživatel	5	Uživatel vybere řešitele úkolu
System	6	System uloží řešitele k úkolu
Uživatel	7	Uživatel vybere soubory
System	8	System uloží vybrané soubory k úkolu

Diagram tříd pro tento případ je vidět na obrázku 14, diagram sekvencí na obrázku 15 a scénář z use case diagramu je vidět v tabulce 4.

3.8 Přidat soubor



Obrázek 16. Přidat soubor – Diagram aktivit - UML



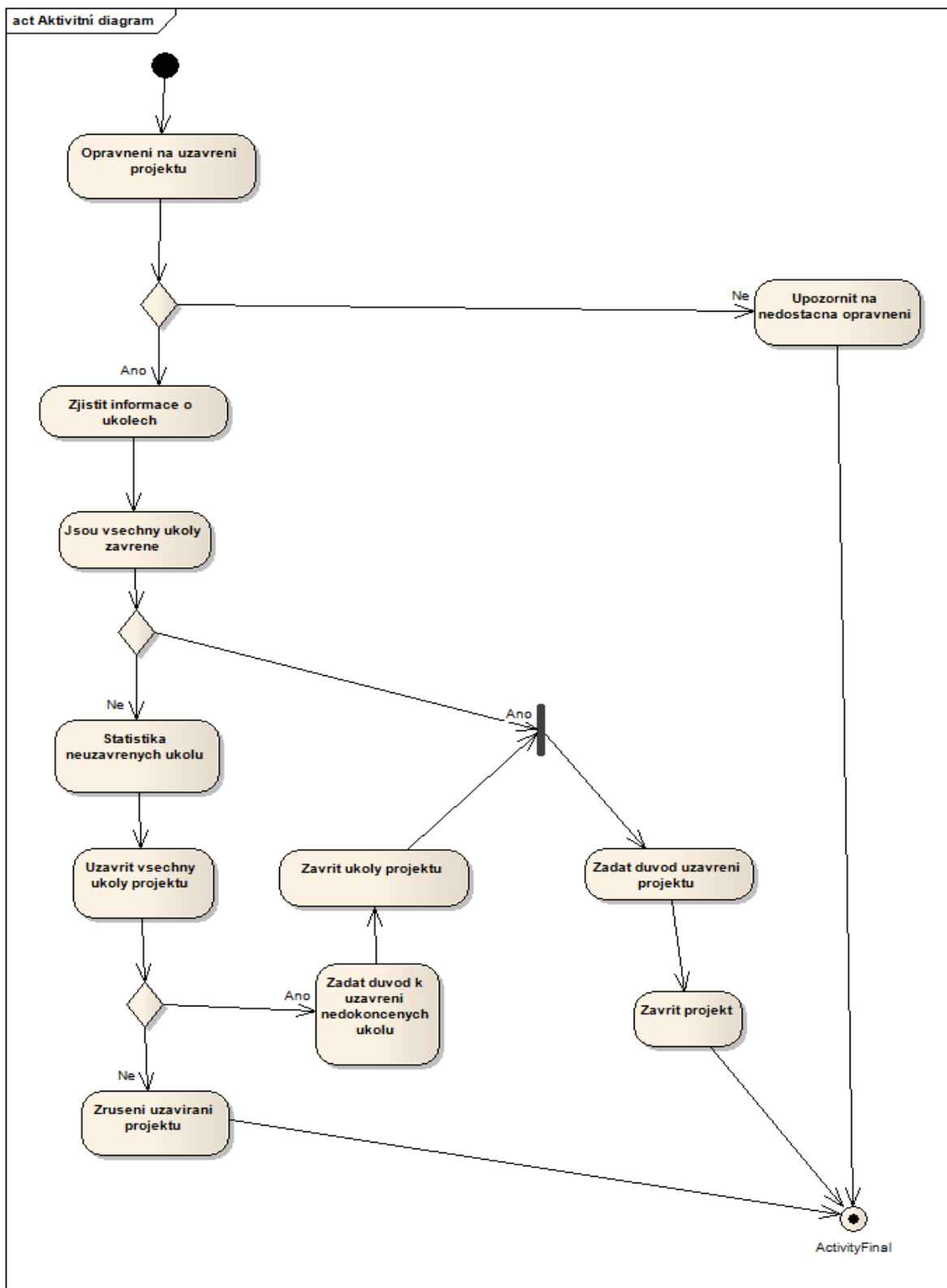
Obrázek 17. Přidat soubor – Diagram sekvencí - UML

Tab. 5. Přidat soubor - Scénář

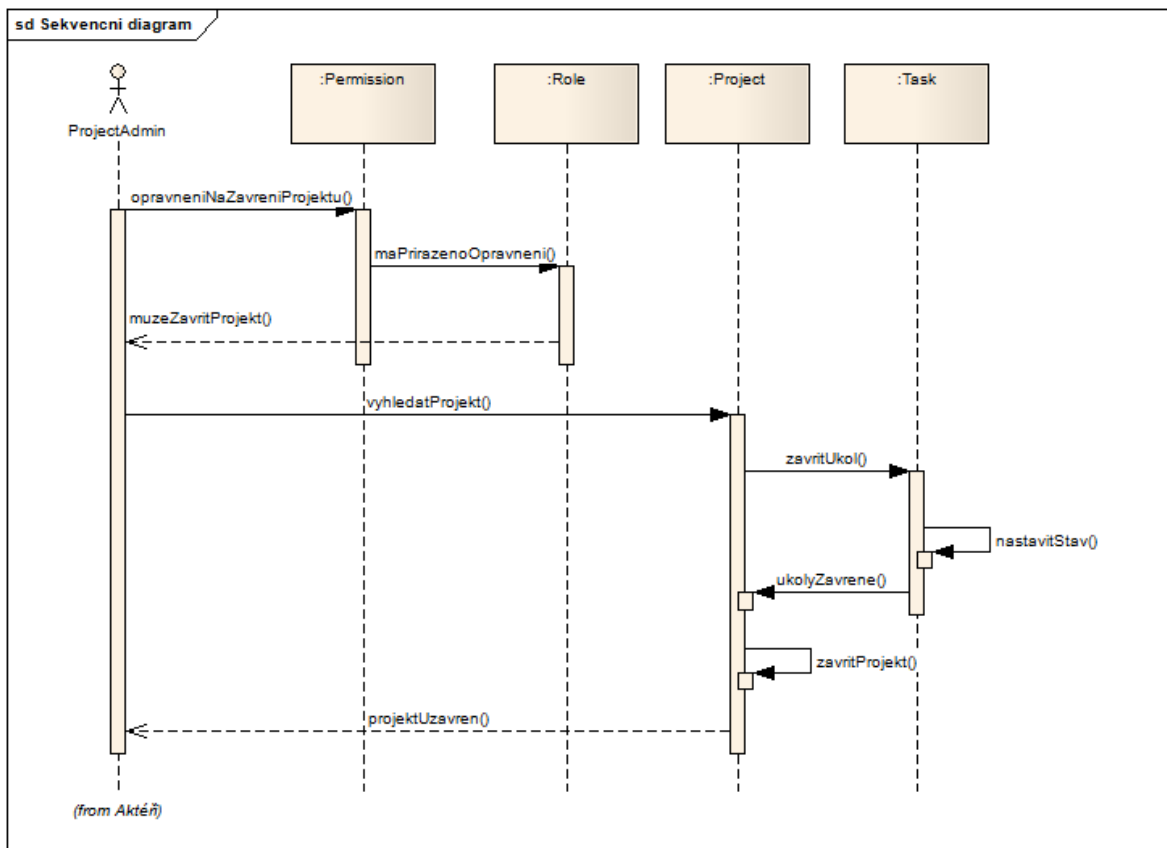
Aktér	Krok	Akce
Uživatel	1	Uživatel vybere přidání souboru
System	2	System ověří oprávnění uživatele
Uživatel	3	Uživatel vybere soubor
System	4	System ověří validitu souboru
System	5	System uloží soubor

Diagram tříd pro tento případ je vidět na obrázku 16, diagram sekvencí na obrázku 17 a scénář z use case diagramu je vidět v tabulce 5.

3.9 Zavřít projekt



Obrázek 18. Zavřít projekt – Diagram aktivit - UML



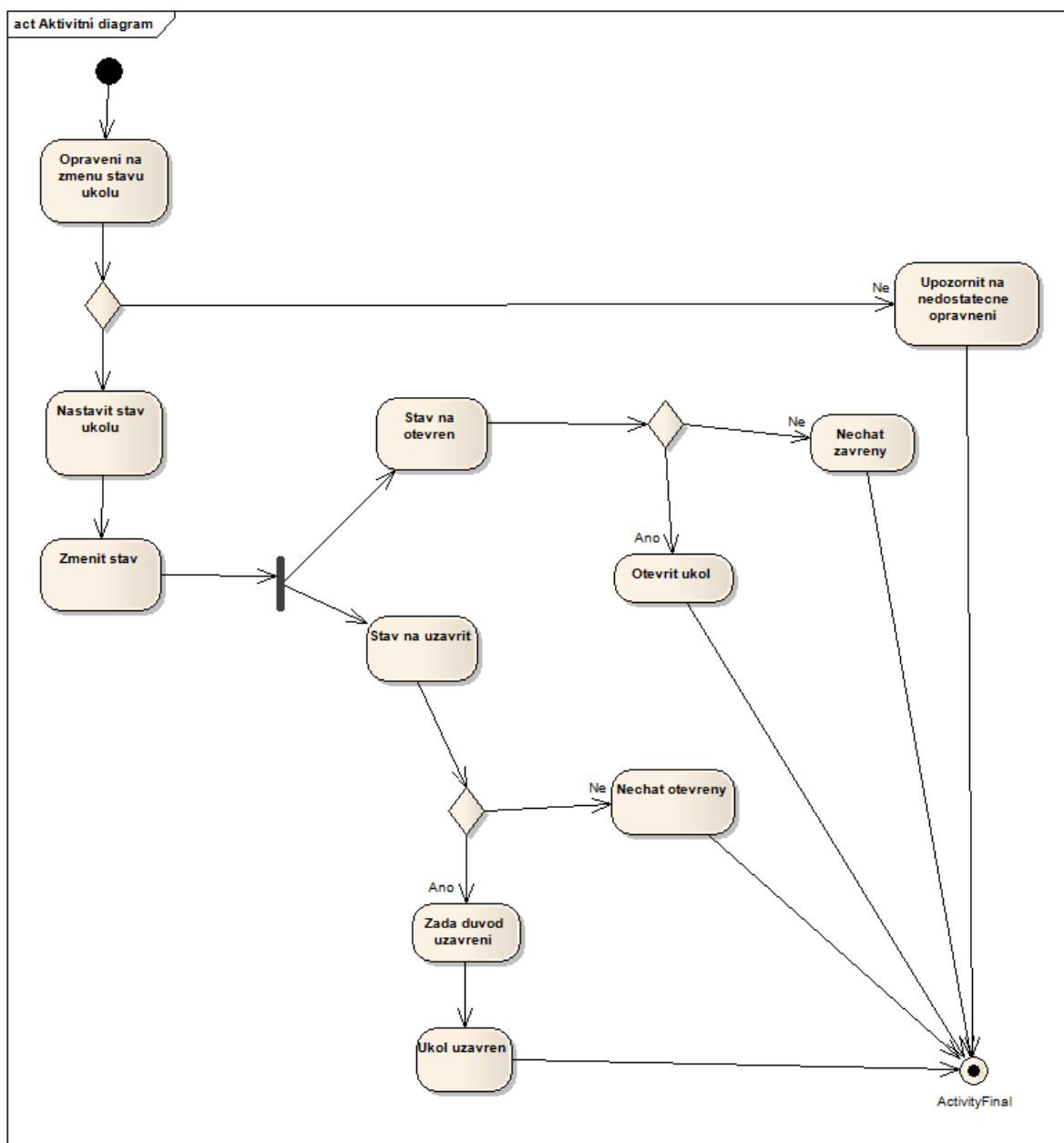
Obrázek 19. Zavřít projekt – Diagram sekvencí - UML

Tab. 6. Zavřít projekt - Scénář

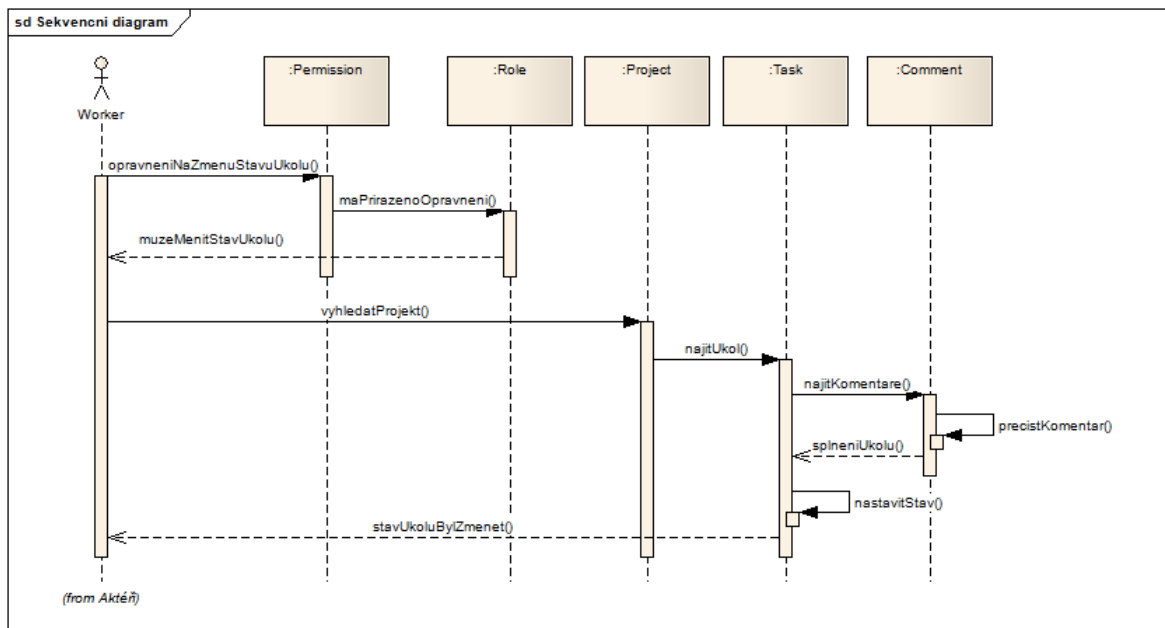
Aktér	Krok	Akce
Uživatel	1	Uživatel požádá o zavření projektu
System	2	System ověří zda má uživatel oprávnění k zavření projektu
Uživatel	3	Uživatel vybere projekt pro zavření
System	4	System vyhledá všechny otevřené úkoly projektu
System	5	System zavře všechny otevřené úkoly projektu
System	6	System zavře projekt

Diagram tříd pro tento případ je vidět na obrázku 18, diagram sekvencí na obrázku 19 a scénář z use case diagramu je vidět v tabulce 6.

3.10 Změnit stav úkolu



Obrázek 20. Změnit stav úkolu – Diagram aktivit - UML



Obrázek 21. Změnit stav úkolu – Diagram sekvencí - UML

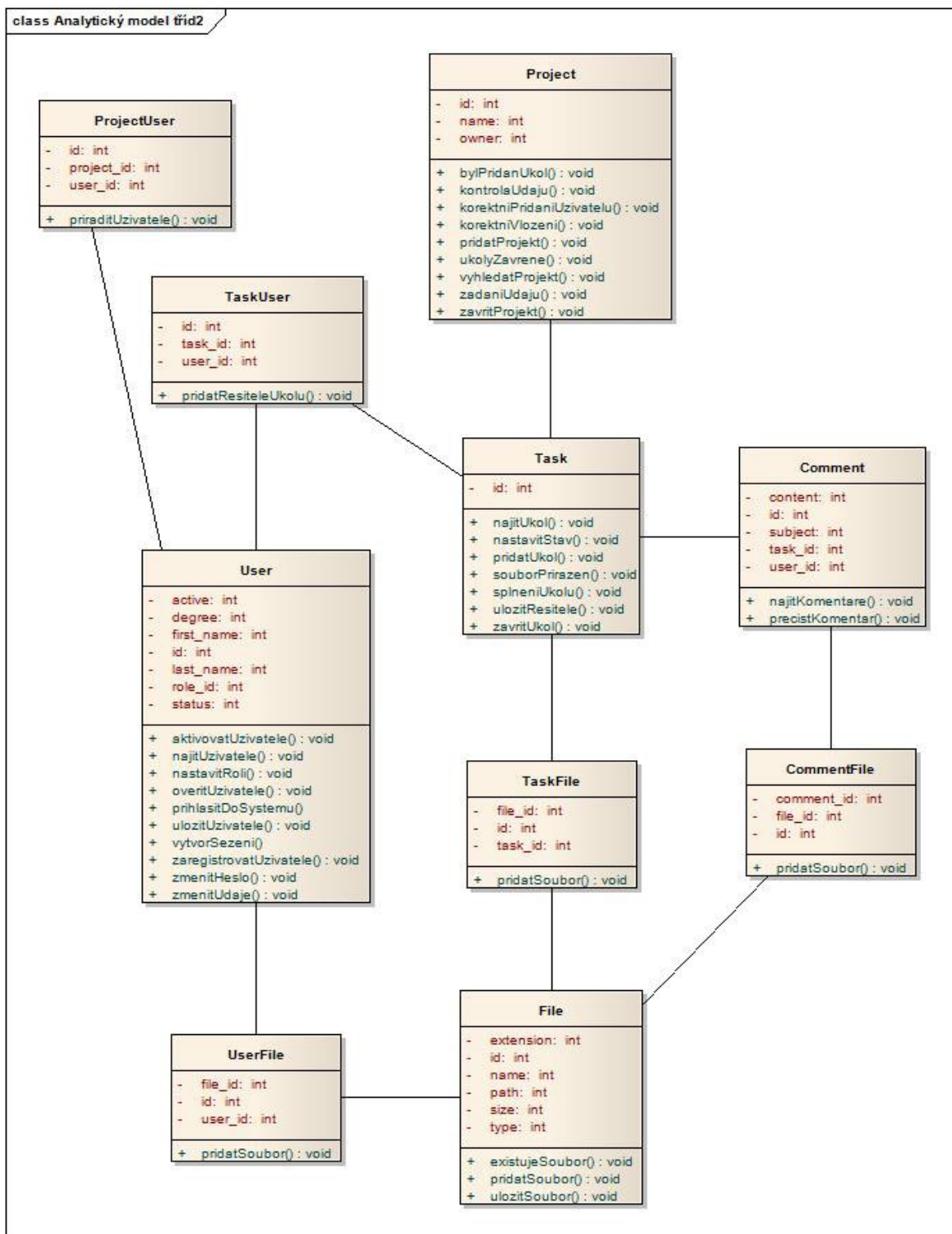
Tab. 7. Změnit stav úkolu - Scénář

Aktér	Krok	Akce
Uživatel	1	Uživatel zobrazí změnu stavu úkolu
System	2	System zobrazí formulář pro změnu stavu úkolu
Uživatel	3	Uživatel nastaví stav úkolu na uzavřený
System	4	System ověří validitu a zda je možné úkol uzavřít
System	5	System informuje všechny přiřazené uživatele o změně stavu
Uživatel	6	Uživatel si zobrazí informaci o změně stavu úkolu

Diagram tříd pro tento případ je vidět na obrázku 20, diagram sekvencí na obrázku 21 a scénář z use case diagramu je vidět v tabulce 7.

3.11 Diagram tříd

Znázorňuje statickou datovou strukturu a operace a souvislosti mezi objekty, jak je na obrázku číslo 22. Datové struktury kompletuje do tříd a ukazuje vazby mezi třídami [12].



Obrázek 22. Změnit stav úkolu – Diagram sekvencí - UML

3.12 Diagram sekvencí

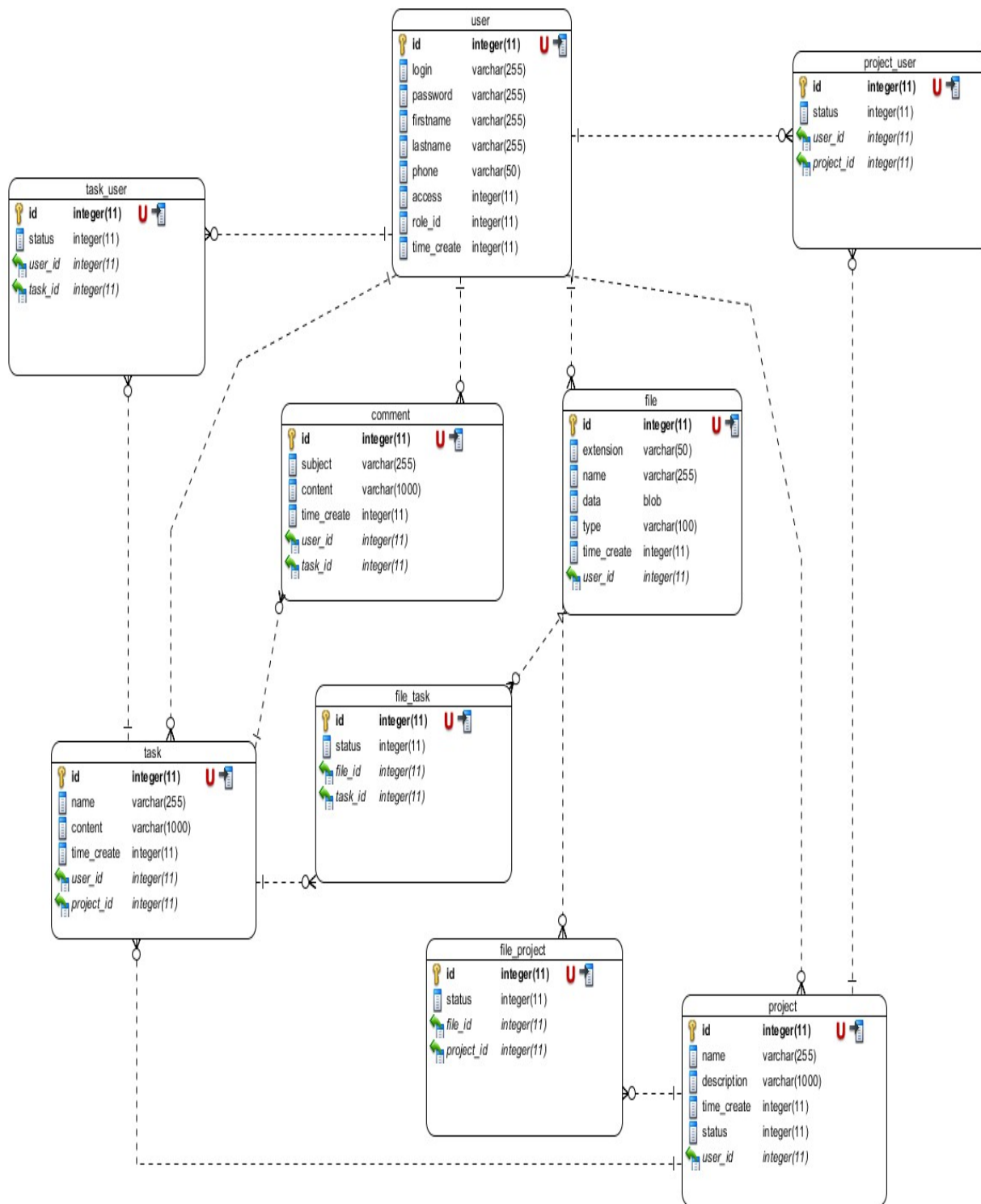
Sekvenční diagram je diagram interakce, který ukazuje, jak spolu procesy navzájem pracují a v jakém pořadí. Tento diagram ukazuje objekty interakce uspořádané v časové posloupnosti. Znárodnuje objekty a třídy zapojené do scénáře a sekvence zpráv, které jsou vyměňovány mezi objekty. Tyto zprávy jsou potřebné k provedení scénáře. Sekvenční diagramy jsou obvykle spojeny s diagramy užití a jsou někdy nazývány schémata událostí nebo scénáře událostí. Sekvenční diagram ukazuje různé procesy nebo objekty, které žijí současně, a které si vyměňují zprávy mezi nimi. Pořadí je určeno dle pozice zprávy. Čím je zpráva v diagramu výše, tím dříve se provádí. To umožňuje specifikaci jednoduchých scénářů [12].

3.13 Diagram aktivit

V jazyce UML popisuje chování. Slouží k modelování procesů a logiky. Procesy jsou určeny sekvencí jednotlivých kroků. Sekvence kroků reprezentuje řídicí tok. Mezi kroky se mohou objevit symboly typu rozhodnutí, vstupní a výstupní podmínky a spojení. Dále se zde nachází události. Ty dělíme na příchozí a časové [12].

3.14 E-R diagram

Tento diagram se používá pro abstraktní znázornění dat. Entity-relationship diagram vytváří model datové části systému. Modelují se zde data, atributy a struktura. Souhrně se tomu říká entita. Entita představuje objekt reálného světa, k němuž chceme v systému uchovávat specifické informace. V diagramu je nadepsána podstatným jménem. Atributy entity jsou vlastnosti tohoto objektu, jak je na obrázku 23. Pokud se jedná o automobil, může to být barva, pokud o člověka, může to být věk. Další částí diagramu je relace (vztah), které je definován slovesem. Vztah je vazba vedená mezi dvěma entitami, o které chceme evidovat a uchovávat informace. Vztahová množina je množina sdružující vztahy stejného typu. Na ERD je zakreslena pojmenovanou hranou, která spojuje jednotlivé entitní množiny. Atribut představuje datový prvek v modelu [12].



Obrázek 23. E-R diagram - UML

4 IMPLEMENTACE

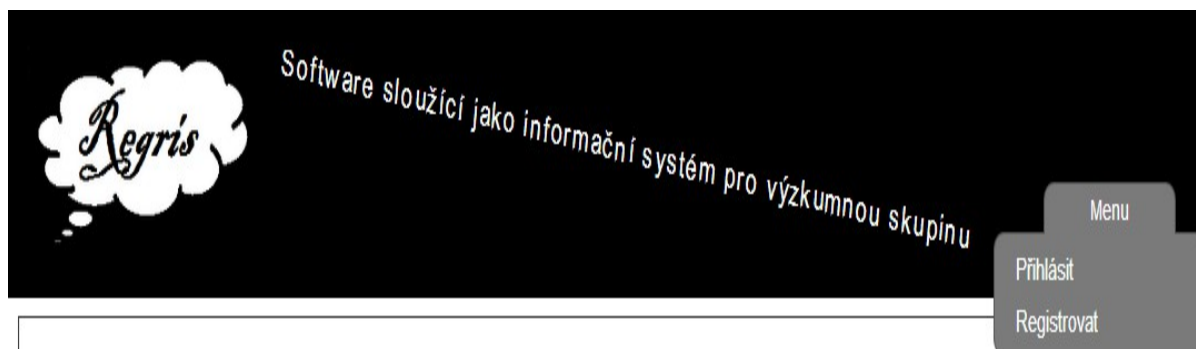
4.1 GUI

Grafické uživatelské rozhraní umožňuje ovládat počítač pomocí interaktivních grafických ovládacích prvků. Na monitoru počítače jsou zobrazena okna, ve kterých programy zobrazují svůj výstup. Uživatel používá klávesnici, myš a grafické vstupní prvky jako jsou menu, ikony, tlačítka, posuvníky, formuláře a podobně [13].

4.1.1 HTML

HyperText Markup Language, označovaný zkratkou HTML, je značkový jazyk pro hypertext. Je hlavním z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci dokumentů na internetu [13].

4.1.2 Hlavička – nepřihlášený



Obrázek 24. Hlavička - nepřihlášený - Implementace

Vrchní banner obsahuje logo, popisek a menu. Menu obsahuje dvě položky, jak je na obrázku 24. První z nich je Přihlásit se. Po přihlášení se uživateli zobrazí hlavní stránka aplikace. Druhá položka menu je Registrovat. Po registraci je uživatel vyzván k přihlášení do aplikace.

4.1.3 Registrace



Registrovat se

Login

Heslo

Jméno

Příjmení

E-mail

Telefon

Odeslat

Obrázek 25. Registrace uživatele - Implementace

Při registraci musí uživatel vyplnit všechny údaje, jak je na obrázku 25. Tím že se zaregistruje se automaticky stává nejvyšší autoritou, která může vytvářet projekty. Přidávat uživatele k projektům a vykonávat další administrátorské činnosti. Po registraci je se může uživatel přihlásit.

4.1.4 Přihlášení



Zadejte login a heslo

Login

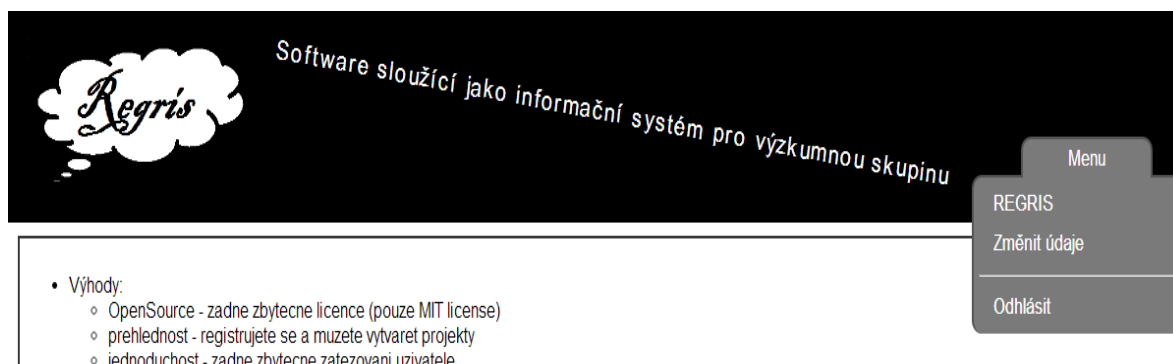
Heslo

Přihlásit

Obrázek 26. Přihlášení uživatele - Implementace

Po zadání přihlašovacích údajů, jak je na obrázku 26, je uživatel přesměrován na úvodní stránku aplikace.

4.1.5 Hlavička – přihlášený



Regris

Software sloužící jako informační systém pro výzkumnou skupinu

Menu

- REGRIS
- Změnit údaje
- Odhlásit

• Výhody:

- OpenSource - zadne zbytecne licence (pouze MIT license)
- prehlednost - registrujete se a muzete vytvaret projekty
- jednoduchost - zadne zbytecne zatezovani uzivatele

Obrázek 27. Hlavička – přihlášený - Implementace

Menu obsahuje tři položky. První z nich je REGRIS. Tím se uživatel dostane na úvodní stránku aplikace. Druhá položka menu je Změnit údaje. Po otevření tohoto odkazu se zobrazí editační okno uživatele. Poslední položka je Odhlásit. Tento odkaz uživatele odhlásí ze systému, jak je na obrázku 27.

4.1.6 Přidání projektu

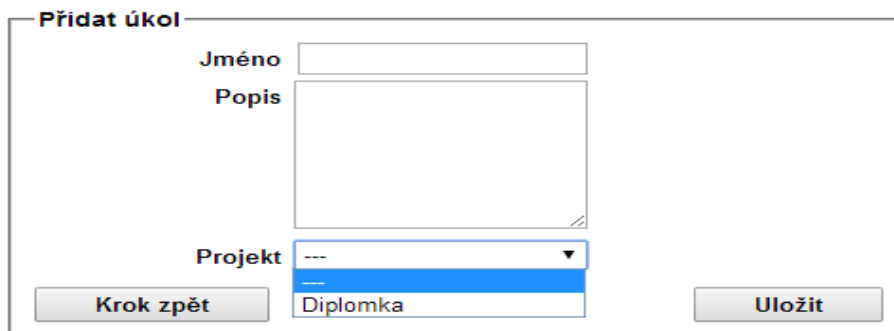


The screenshot shows a web form titled "Přidat projekt". It has two input fields: "Jméno" (Name) and "Popis" (Description). Below the fields are two buttons: "Krok zpět" (Previous step) and "Uložit" (Save).

Obrázek 28. Přidání projektu - Implementace

Přidání projektu je nejdůležitější věc v systému. Po vytvoření projektu je dále možné vytvářet uživatele a úkoly k tomuto projektu. Pro vytvoření projektu je důležité vyplnit všechny údaje, jak je na obrázku 28.

4.1.7 Přidání úkolu



The screenshot shows a web form titled "Přidat úkol". It has three input fields: "Jméno" (Name), "Popis" (Description), and "Projekt" (Project). The "Projekt" field is a dropdown menu with "Diplomka" selected. Below the fields are two buttons: "Krok zpět" (Previous step) and "Uložit" (Save).

Obrázek 29. Přidání úkolu - Implementace

Přidání úkolu probíhá stejně jako přidání projektu. Je nutné vyplnit všechny položky, které jsou uživateli nabídnuty, jak je na obrázku 29. Pokud se vytváří úkol z hlavní stránky, je nutné vybrat úkol ke kterému patří. Pokud se úkol přidává z detailu projektu, tak zde není výběrové poličko projektu. Po uložení je možné přidávat uživatele k vybraným úkolům.

4.1.8 Přidání uživatele k projektu

The image shows two screenshots of a web application interface for adding users to a project.

The top screenshot, titled "Přidat uživatele", shows a form with a dropdown menu labeled "Uložení uživatelé" containing three dashes. Below the dropdown are two buttons: "Krok zpět" on the left and "Uložit" on the right.

The bottom screenshot, titled "Vytvořit a přidat uživatele", shows a form with several input fields: "Login", "Heslo", "Jméno", "Příjmení", "E-mail", and "Telefon". Below these is a "Role" dropdown menu with three dashes. The dropdown is open, showing two options: "Úkolovač" and "Uživatel". Below the dropdown are two buttons: "Krok zpět" on the left and "Uložit" on the right.

Obrázek 30. Přidání uživatele k projektu - Implementace

Přidání uživatelů k projektu probíhá na dvou úrovních. První je výběr uživatelů z již existujících projektů, které vytvořil ProjectAdmin. Druhá úroveň je zaregistrování uživatelů do systému a přiřazení těchto uživatelů k projektu. Toto přidání funguje stejně jako registrace, jen je přidáné výběrové políčko s rolí, kterou chce ProjectAdmin přiřadit uživatelům. Každý uživatel může být v projektu jen jednou, jak je na obrázku 30.

4.1.9 Přidání uživatele k úkolu

The image shows a screenshot of the "Přidat uživatele" form. The dropdown menu "Uložení uživatelé" is open, showing a list of users. The first user is "uzivatel (uzivatel uzivatel)". Below the dropdown are two buttons: "Krok zpět" on the left and "Uložit" on the right.

Obrázek 31. Přidání uživatele k úkolu - Implementace

Přidání uživatele k úkolu probíhá výběrem ze všech uživatelů, které jsou přiřazeny k projektu. Tento výběr jen omezuje uživatele, kteří mohou vidět jen ty úkoly, které mají přiřazené nebo sami vytvořili, jak je na obrázku 31.

4.1.10 Přidání komentáře

Přidat komentář

Jméno

Popis

Úkol ---

Ukol 1

Obrázek 32. Přidání komentáře - Implementace

U komentáře je nutné vyplnit všechny položky. Všechny položky formuláře jsou povinné. Pokud je komentář přidáván z hlavní stránky, je nutné ve výběru úkolu přidat úkol. Pokud přidáváme komentář z náhledu úkolu, není toto nutné, jak je na obrázku 32.

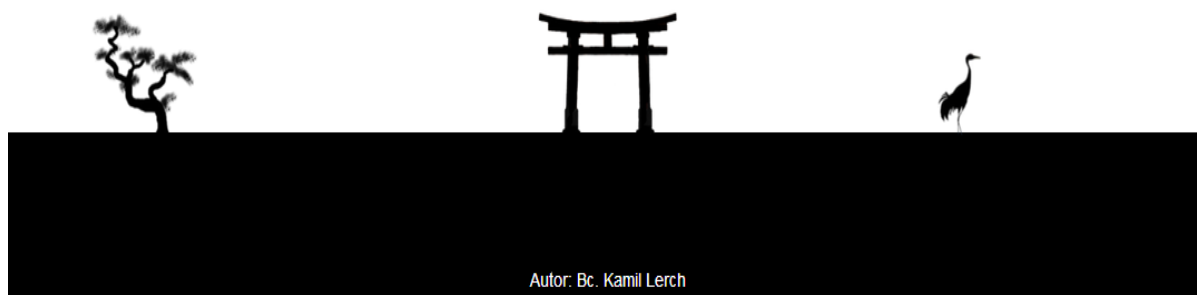
4.1.11 Detail projektu

Jméno projektu: Diplomka
Popis projektu: První projekt pro vytvoření diplomky
Datum vytvoření: 18.4.2014 0:06:22



Přiřazené úkoly

Jmeno	Obsah	Přiřazený uživatel	Čas vytvoření	Uživatelé	Komentáře	Stav	
Ukol 1	print screen pridani uzivatelu k u...	Test Test	18.4.2014 0:12:52	0	4	Nový/Otevřený	
Ukol 2	Jit spat	Test Test	18.4.2014 0:27:05	0	2	Nový/Otevřený	
Vstat	Nezaspat jak vcera	Test Test	18.4.2014 0:27:36	0	0	Nový/Otevřený	







Obrázek 33. Detail projektu - Implementace













Detail projektu obsahuje informace o projektu, možnost k projektu přidávání souvisejících souborů a přehled všech přiřazených úkolů a jejich stavů. Úkoly je odsud možné editovat, zavírat, nahlížet do nich nebo k nim přidávat uživatele, jak je na obrázku 33.

4.1.12 Náhled stránky Regris







Správa projektů

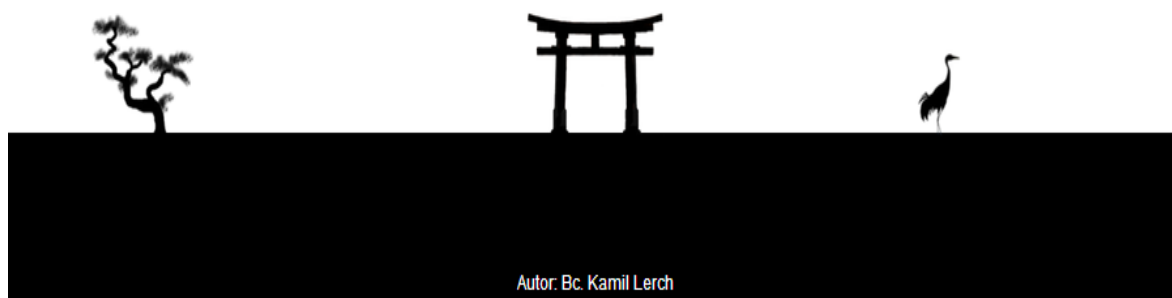
Jmeno	Popis	Uživatelé	Úkoly	Čas vytvoření	
Diplomka	První projekt pro vytvoření diplomky	1	3	18.4.2014 0:06:22	   

Správa úkolů

Jmeno	Obsah	Přiřazený uživatel	Čas vytvoření	Uživatelé	Komentáře	Stav	
Ukol 1	print screen pridani uzivatelu k u...	Test Test	18.4.2014 0:12:52	0	4	Nový/Otevřený	   
Ukol 2	Jít spat	Test Test	18.4.2014 0:27:05	0	2	Nový/Otevřený	   
Vstat	Nezaspat jak vcera	Test Test	18.4.2014 0:27:36	0	0	Nový/Otevřený	   

Správa komentářů

Jmeno	Popis	Vytvořil	Úkol	Čas vytvoření	
Koment 1	No tak zatím se snazim stihat a snad se ...	Test Test	Ukol 1	18.4.2014 0:28:55	
Koment 2	Spat bych mel jit	Test Test	Ukol 2	18.4.2014 0:28:05	
Koment 3	Uz nevim co psat	Test Test	Ukol 2	18.4.2014 0:29:11	
Koment 4	Jeste dva komntare	Test Test	Ukol 1	18.4.2014 0:29:45	
Koment 5	Jeste jeden	Test Test	Ukol 1	18.4.2014 0:30:04	
Koment 6	Uz hotovo	Test Test	Ukol 1	18.4.2014 0:30:19	



Obrázek 34. Hlavní stránka aplikace Regris - Implementace

Hlavní stránka aplikace, která se uživateli zobrazí po přihlášení. Pro každou roli je jiná. V případě, že uživatel není ProjectAdmin, nezobrazuje se možnost spravovat projekty. Do

projektů je možné jen nahlížet, ale nikoliv je editovat. Uživatelé se dále zobrazují jen úkoly, které má přiřazené, nebo je vytvořil. Z komentářů se zobrazují jen takové, které uživatel sám vytvořil, jak je na obrázku 34. Na celou historii řešení úkolu je nutné se podívat přes úkol.

4.1.13 Detail úkolu

Jméno projektu: Diplomka
Popis projektu: První projekt pro vytvoření diplomky
Datum vytvoření projektu: 18. 4. 2014 0:06:22

Jméno úkolu: Úkol 1
Popis úkolu: print screen přidání uživatele k úkolu
Datum vytvoření úkolu: 18. 4. 2014 0:12:52

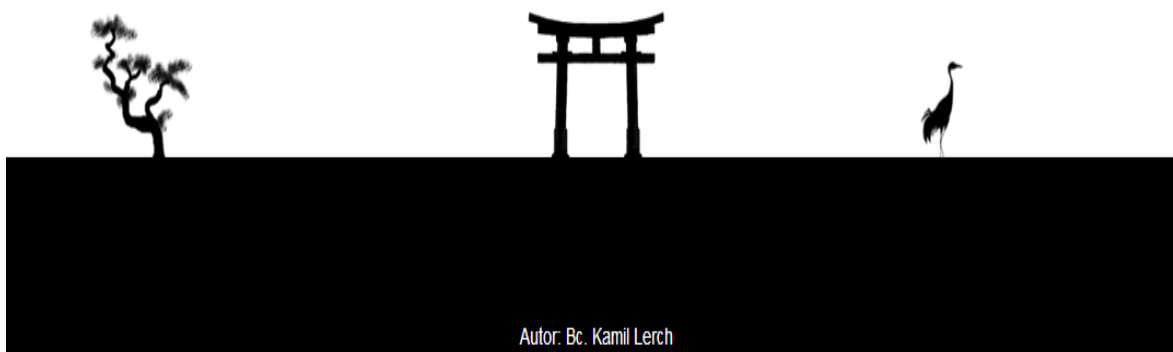


Komentáře k úkolu



Jméno	Popis	Vytvořil	Úkol	Čas vytvoření	
Koment 1	No tak zatím se snažím stíhat a snad se mi t...	Test Test	Úkol 1	18. 4. 2014 0:28:55	
Koment 4	Jeste dva komentare	Test Test	Úkol 1	18. 4. 2014 0:29:45	
Koment 5	Jeste jeden	Test Test	Úkol 1	18. 4. 2014 0:30:04	
Koment 6	Uz hotovo	Test Test	Úkol 1	18. 4. 2014 0:30:19	

Krok zpět



Obrázek 35. Detail úkolu - Implementace

Detail úkolu obsahuje informace o úkolu. Přináší také možnost přidávat souboru a obsahuje seznam všech komentářů seřazených od nejstaršího po nejnovější. Komentář je možné editovat, jen pokud se jedná o komentář uživatele, který jen vytvořil, jak je na obrázku 35.

4.2 Nástroje

4.2.1 PhpStorm

JetBrains PhpStorm poskytuje inteligentní editor pro PHP, HTML a JavaScript pomocí analýzy kódu on-the-fly, prevenci chyb a automatizovaných refaktorování pro PHP a JavaScript kód. Pro vytvoření aplikace byla použita verze 7.1.3 tohoto vývojového prostředí. Tento editor je jedním z nejlepších pro vývoj PHP aplikací a to nejen pro funkce, které podporuje, ale hlavně pro rychlost a možnosti nastavení.

4.2.2 WAMP

WampServer je vývojové prostředí Windows. To umožňuje vytvářet webové aplikace s Apache2, PHP a databází MySQL. Vedle PhpMyAdmin umožňuje snadno spravovat své databáze. Toto prostředí dokáže automaticky nainstalovat vše potřebné pro vývoj. Dále je možné tento server dále ladit bez nebezpečí, že by se poškodily konfigurační soubory.

5 VÝSLEDKY

System byl implementovaný dle analýzy a umožňuje uživateli provádět operace se systémem, které byly dle analýzy požadovány. V případě, že je uživatel registrovaný a přihlášený, může vstoupit do systému. System ověřuje identitu uživatele při každém vstupu na stránky. Všechno nastavování je vytvořeno jako co nejjednodušší pro uživatele. Složitý systém se spoustou vyskakovacích oken by se stal pro uživatele nepřehledným. Proto existuje stránka, kde uživatel vidí vše podstatné na jednom místě. Uživateli je dle nastavené role povoleno vytvářet projekty, úkoly, uživatele, komentáře a přidávat soubory. Dále je možné tyto entity modifikovat, propojovat je a kombinovat. Tyto případy, že uživatel může provádět dané akce, jsou otestovány. Dále jsou otestovány modifikace URL (předávání parametrů, které jsou očekávány) a jiné prevence proti hrozbám. U obyčejných uživatelů se jedná o nechtěnou změnu parametrů, jako je třeba přepsání parametrů a ověření, že uživatelé mohou vidět jen k sobě přiřazené úkoly. Dále je otestována možnost správného obalování parametrů u dotazů z důvodu použití Injectionu na zjištění struktury databáze a výpisu dat z ní nebo vlomení JavaScriptu přímo do stránky. Toto jsou případy hrozeb, na které je brán největší zřetel. Jiné případy otestování funkčnosti neproběhly. Tyto případy se otestovaly v průběhu ladění programu.

5.1 Testování

Byly provedeny testy, které měly odhalit možnosti systému před zavedením do normálního používání. Všechny testy jsou jen orientační, ale výsledky je možné brát jako ověření funkčnosti, kterou potřebuje normální uživatel užívající systém pro organizaci práce ve výzkumné skupině. Všechny ostatní nedostatky a doplňující funkce je možné po analýze a návrhu doprogramovat. Důležité je však prověřit kompletní systém, jestli jsou řešení správná a nedošlo k chybě.

5.1.1 Registrace

Provedený test

Účelem tohoto testu je ověřit, zda je systém odolný vůči změnám parametrů při registraci. Pokud uživatel například bude chtít vyšší oprávnění a zkusí nastavit některé parametry, ať už změnou URL nebo parametru.

Výsledek:

Ať už se uživatel bude snažit podvrhnout identifikační číslo role nebo jiné parametry na které systém čeká. Vždy jsou data filtrovaná a nastavena dle požadavků systému.

5.1.2 Změna osobních údajů**Provedený test**

Účelem tohoto testu je ověřit, jestli je systém odolný vůči změnám parametrů při změně osobních údajů. Například pokud uživatel bude chtít vyzkoušet vložit například změnu hesla a nebo přihlašovací jméno.

Výsledek:

Uživateli není umožněno měnit zásadní věci jako je přihlašovací jméno nebo heslo. I kdyby se uživateli povedlo vložit neviditelné elementy a podvrhne stejné pojmenování parametrů, nepovede se mu je uložit. Po odeslání se tyto data nemění.

5.1.3 Přidání projektu**Provedený test**

Tento test zkoumá jestli je možné podvrhnout uživatelovo číslo ve vytváření projektu. Uživatelovo číslo je velice důležité pro další operace jako je přiřazení úkolu k projektu nebo uživatelů.

Výsledek:

Ať už se uživatel bude snažit podvrhnout své identifikační číslo, nebo jiné aby zanesl nějaký zmatek do projektů jiného uživatele, tak se mu toto jednaní nepovede. Tato operace není povolena. Uživatel může vytvářet a editovat jen projekty, které vytvoří on sám.

5.1.4 Editace projektu**Provedený test**

Základní problém tohoto testu je v tom, že uživatel vidí parametr pro editaci. Jedná se o identifikační číslo projektu. Je důležité aby každý uživatel mohl editovat jen své projekty.

Výsledek:

I přesto že uživatel změní číslo projektu, je uplatňovaná restrikce na vlastnictví takového

projektu uživatelem. Díky této vlastnosti není možné editovat projekt jiného uživatele.

5.1.5 Přidání uživatele

Provedený test

Tento test zkoumá jestli je možné přiřadit uživatele k jinému projektu a s vyšším oprávněním. Dalším podvržením by mohlo být použití nějakého již vytvořeného uživatele k cizímu projektu.

Výsledek:

Ať už se uživatel bude snažit změnit parametry formuláře, nikdy mu to nedovolí uložit cizího uživatele. Toto je řešeno systémem nezobrazování a neukládání parametrů přímo z formuláře. Systém tyto přednastavené hodnoty získává až na straně serveru. Uživatel může vytvářet a přidávat jen uživatele v již vytvořených projektech nebo přímo nové a to ke konkrétnímu projektu.

5.1.6 Přidání úkolu

Provedený test

Tento test zkoumá jestli je možné vytvořit úkol v jiném projektu. Toho by bylo možné docílit změnou parametru, který říká k jakému projektu se úkol přiřazuje. Díky přidání cizího projektu by bylo možné zjistit přiřazené uživatele z tohoto projektu.

Výsledek:

Ať už se uživatel bude snažit podvrhnout číslo projektu, systém mu to nedovolí uložit a tím nedojde k narušení integrity dat.

5.1.7 Editace úkolu

Provedený test

Test jestli je možné upravit cizí úkol.

Výsledek:

Ani po změně parametru ani po vložení výběrových polí nebylo možné dostat se do cizího úkolu a změnit jméno nebo popis úkolu. Systém hlídá vlastnictví a přiřazené uživatele k úkolům.

5.1.8 Přidání komentáře

Provedený test

Tento test zjistí jestli můžou přidávat komentáře jen uživatelé, kteří jsou přiřazeni k úkolům nebo i jiné uživatele v systému.

Výsledek:

System dokáže rozpoznat jestli je uživatel, který komentář přidává přiřazený k úkolu a následně dovolí tomuto uživateli přidat komentář. Není možné nijak podvrhnout komentář. Políčko s výběrem úkolů je testováno na vlastnictví uživatelem který vkládá komentář.

5.1.9 Editace komentáře

Provedený test

Tímto testem by se měl odhalit problém s editací komentáře. Komentář by měl editovat jen člověk co je přiřazený k projektu.

Výsledek:

Ať už se uživatel bude snažit podvrhnout číslo komentáře, pokud tento komentář je v úkolu který mu nepatří není možné jej upravit.

5.1.10 Zavření úkolu

Provedený test

Úkol by mělo být možné uzavřít jen přiřazenými uživateli k projektu. Nemělo by být možné uzavírat úkoly napříč projekty.

Výsledek:

Podvržení parametru nedovolí uživateli uzavřít úkol kterému není řešitelem. To znamená že pokud uživatel nemůže editovat úkol, nemůže mu nastavit status že je zavřený.

5.1.11 Zavření projektu

Provedený test

Zavření projektu by mělo být umožněno jen správci projektu, který zároveň kontroluje, jestli jsou úkoly splněny a je možné bez obav zavřít projekt.

Výsledek:

Uživatelé, kteří jsou sice přiřazeni k projektu jej však nemůžou zavřít. Je to proto, že jen správce projektu rozhoduje o úspěšném nebo neúspěšném vyřešení projektu.

5.1.12 Náhled úkolu**Provedený test**

V náhledu úkolu je možné přidávat soubory, editovat a nahlížet do komentářů. Je zde spousta informací a formulářů které mohou být zneužity. Proto by zde měly být jasná pravidla pro celou tuhle stránku.

Výsledek:

Pravidla pro ovládací prvky platí stejná jako na hlavní stránce aplikace po přihlášení. Výhodou je soupis informací pro konkrétní úkol a dále možnost vložení souboru přímo k úkolu.

5.1.13 Náhled projektu**Provedený test**

V náhledu projektů je možné přidávat soubory, editovat a nahlížet do úkolů. Je zde také hodně formulářů, přes které by se mohli uživatelé snažit dostat do systému a ovlivnit jiné uživatele. Proto zde fungují ještě větší omezení než u náhledu úkolu.

Výsledek:

I v tomto případě platí stejná pravidla pro ovládací prvky jako na hlavní stránce aplikace po přihlášení. Výhodou je soupis informací pro konkrétní projekt. Také zde je možné přidávat soubory ale navíc je možné přidávat přímo i úkolu. Po vytvoření úkolů je možné jim přidávat uživatele, soubory a komentáře.

ZÁVĚR

Výsledný systém je prototyp, který implementuje výše uvedené požadavky na informační systém pro výzkumnou skupinu. Systém implementuje registraci a vytváření uživatelů, přidávání projektů, jejich dělení na úkoly a komentování těchto úkolů. Tyto úkony jsou všechny potřeba pro komunikaci skupiny.

Skupina se dělí do několika podskupin. Tyto podskupiny jsou modelovány rolami. V systému jsou čtyři role, které jsou pojmenovány takto: SuperAdmin, ProjectAdmin, TaskAdmin a Worker. Každá role má jiné možnosti jak nakládat se systémem. SuperAdmin je hlavní správce celé aplikace a nemá žádná omezení. Druhou rolí je ProjectAdmin. Uživatel s touto rolí se stává správcem svých projektů. Dalo by se říci že je manažerem výzkumné skupiny. Může do svých projektů registrovat a přiřazovat další uživatele. Skupina projektů tvoří jednu velkou a uzavřenou skupinu. To znamená že tuto aplikaci může využívat více výzkumných skupin, které o sobě nemusí vůbec vědět. TaskAdmin a Worker jsou řešiteli úkolů. Worker může úkoly jen řešit, editovat a uzavírat je. Zato TaskAdmin může jednotlivé úkoly i vytvářet a zadávat podřízenému Workerovi.

Má přehledné a jednoduché uživatelské rozhraní pro snadnější procházení a vkládání informací. Systém využívá základních požadavků na systém. Díky své jednoduchosti jej dokážou pochopit a používat i méně informačně gramotní uživatelé.

Systém neimplementuje možnost použití multijazyčných mutací. Z tohoto důvodu se sice všude v systému počítá s překlady, ty ale byly vytvořeny jen do češtiny. V případě využívání aplikace cizojazyčnými uživateli, je možné implementovat překlady do jiných jazyků a poskytnout je jako placenou službu.

Dalším plánovaným rozšířením systému je možnost komunikace s uživateli přes email. Tato vlastnost je velice výhodná pro upozorňování na změny. Ať už se jedná o samostatné změny v projektech, úkolech a komentářích nebo o registraci a přidání uživatelů do projektů. Systém zatím upozorňuje emailem pouze na registraci uživatelů, ale pouze v testovacím módu.

Aplikaci je dále možné rozšířit o další vylepšení. Jedná se většinou o moderní metody spravování aplikace a co největší možnost nastavení uživatelem. Tím se rozumí například možnost nastavení vlastního vzhledu celé aplikace nebo jejich jednotlivých částí. Tím však může vzniknout problém pro běžné uživatele. Dá se říci že tímto se uspokojí požadavky jen zkušených uživatelů, vývojářů a programátorů, kteří si už za svoje mnohaleté užívání počítačů a na nich nainstalovaných aplikací zvyklí mít možnost přizpůsobit si aplikaci co

nejvíce svému osobnímu vkusu.

Další možností směřování aplikace by mělo být testování v běžném provozu a postupné přidávání nových funkcí dle požadavků uživatelů. Tyto požadavky však musí být vždy přidány do stávající aplikace.

Největšími výhodami systému jsou právě jednoduchost a přehlednost. Systém je volně šířitelný a dostupný na GitHubu. Pro používání systému není nutné se registrovat na stránkách třetích stran ani jim poskytovat jakékoliv data. Systém může běžet lokálně, na firemním serveru nebo vzdáleném hostingu. Jedná se o webový projekt, takže je zaručena multiplatformnost. Pro funkci systému jsou potřeba dnes již běžné požadavky na webové aplikace tj: MySQL server, PHP a webový server a pro práci se systémem webový prohlížeč. Systém byl otestován na několik webových prohlížečích například: Firefox, Google Chrome nebo Internet Explorer.

SEZNAM POUŽITÉ LITERATURY

- [1]. PATTON, Ron. Testování softwaru. Vyd. 1. Praha: Computer Press, 2002. 314 s. ISBN: 80-7226-636-5.
- [2]. KRUCHTEN, Philippe. The Rational Unified Process An Introduction. 2nd edition. Boston: Addison-Wesley, 2000. 298 s. ISBN: 0-201-70710-1.
- [3]. PRESSMAN, Roger S a Richard HUNTER. Software engineering: a practitioner's approach. 7th ed. New York: McGraw-Hill Higher Education, c2010, xxviii, 895 p. McGraw-Hill systems design. ISBN 00-733-7597-7.
- [4]. KERR, James M a Richard HUNTER. Inside RAD: how to build fully functional computer systems in 90 days or less. New York: McGraw-Hill, c1994, xiv, 213 p. McGraw-Hill systems design. ISBN 00-703-4223-7.
- [5]. BECK, Kent. Extrémní programování. Vyd. 1. Praha: Grada, 2002, 158 s. ISBN 80-247-0300-9.
- [6]. CLARKE, Justin a Richard HUNTER. SQL injection attacks and defense: a practitioner's approach. 7th ed. Burlington, MA: Syngress Pub., c2009, xix, 473 p. McGraw-Hill systems design. ISBN 978-159-7494-243.
- [7]. VIEGA, John, Matt MESSIER a Pravir CHANDRA. Network security with OpenSSL. 1st ed. Sebastopol, CA: O'Reilly, c2002. ISBN 05-960-0270-X.
- [8]. CONVERSE, Tim a Joyce PARK. PHP5 and MySQL bible. [3rd ed.]. Indianapolis, IN: Wiley, c2004. ISBN 07-645-5746-7.
- [9]. NIXON, Robin. Learning PHP, MySQL, JavaScript, and CSS. 2nd ed. Sebastopol, CA: O'Reilly, 2012, xxi, 556 p. ISBN 14-493-1926-2.
- [10]. GOLDING, David. Beginning CakePHP: from novice to professional. New York : distributed by Springer-Verlag: Apress, c2008, xix, 319 p. ISBN 14-302-0977-1.
- [11]. KUTNOHORSKÁ, Jana. Výzkum v ošetrovatelství. 1. vyd. Praha: Grada, 2009, 175 s. Sestra. ISBN 978-802-4727-134.
- [12]. ROSENBERG, Doug a Matt STEPHENS. Use case driven object modeling with UML: theory and practice. [New ed.]. New York: Distributed to the book trade worldwide by Springer-Verlag, c2007, xxxi, 438 p. Sestra. ISBN 978-159-0597-743.
- [13]. HLAVENKA, Jiří a Robert Banh TECHNICAL REVIEWER. Vytváříme WWW stránky a spravujeme moderní web site. 6. vyd. Brno: Computer Press, 2002, 354 s. Wrox beginning guides. ISBN 80-7226-748-5.

- [14]. PUNTAMBEKAR, A.A. Software Engineering. 1. vyd. Amit Residency, 412, Shaniwar Peth, Pune - 411 030, India: Technical Publications Pune, 2008. ISBN 9788184313963.
- [15]. PITT, Chris. Pro PHP MVC. New York: Distributed to the book trade worldwide by Springer Science Business Media, c2012, xxv, 471 p. Expert's voice in open source. ISBN 978-143-0241-652.
- [16]. KUNSTOVÁ, Renata. Skupinová spolupráce, správa a řízení oběhu dokumentů. Vyd. 1. Praha: Vysoká škola ekonomická, Fakulta informatiky a statistiky, 1999, 80 s. ISBN 80-707-9647-2.

SEZNAM OBRÁZKŮ

Obrázek 1. Vodopádový přístup.....	13
Obrázek 2. Prototypový přístup.....	14
Obrázek 3. Spirální přístup.....	16
Obrázek 4. Model-View-Controller Diagram.....	23
Obrázek 5. Nefunkční požadavky - UML.....	32
Obrázek 6. Funkční požadavky - UML.....	33
Obrázek 7. Případy užití - UML.....	34
Obrázek 8. Přihlášení – Diagram aktivit - UML.....	35
Obrázek 9. Přihlášení – Diagram sekvencí - UML.....	36
Obrázek 10. Registrace – Diagram aktivit - UML.....	1
Obrázek 11. Registrace – Diagram sekvencí - UML.....	2
Obrázek 12. Založit projekt – Diagram aktivit - UML.....	3
Obrázek 13. Založit projekt – Diagram sekvencí - UML.....	4
Obrázek 14. Přidat úkol – Diagram aktivit - UML.....	5
Obrázek 15. Přidat úkol – Diagram sekvencí - UML.....	6
Obrázek 16. Přidat soubor – Diagram aktivit - UML.....	7
Obrázek 17. Přidat soubor – Diagram sekvencí - UML.....	8
Obrázek 18. Zavřít projekt – Diagram aktivit - UML.....	9
Obrázek 19. Zavřít projekt – Diagram sekvencí - UML.....	10
Obrázek 20. Změnit stav úkolu – Diagram aktivit - UML.....	11
Obrázek 21. Změnit stav úkolu – Diagram sekvencí - UML.....	12
Obrázek 22. Změnit stav úkolu – Diagram sekvencí - UML.....	13
Obrázek 23. E-R diagram - UML.....	15
Obrázek 24. Hlavička - nepřihlášený - Implementace.....	16
Obrázek 25. Registrace uživatele - Implementace.....	17
Obrázek 26. Přihlášení uživatele - Implementace.....	17
Obrázek 27. Hlavička – přihlášený - Implementace.....	17
Obrázek 28. Přidání projektu - Implementace.....	18
Obrázek 29. Přidání úkolu - Implementace.....	18
Obrázek 30. Přidání uživatele k projektu - Implementace.....	19
Obrázek 31. Přidání uživatele k úkolu - Implementace.....	19
Obrázek 32. Přidání komentáře - Implementace.....	20

Obrázek 33. Detail projektu - Implementace.....	20
Obrázek 34. Hlavní stránka aplikace Regris - Implementace.....	21
Obrázek 35. Detail úkolu - Implementace.....	22

SEZNAM TABULEK

Tab. 1. Přihlášení - Scénář.....	36
Tab. 2. Registrace - Scénář.....	2
Tab. 3. Založit projekt - Scénář.....	4
Tab. 4. Přidat úkol - Scénář.....	6
Tab. 5. Přidat soubor - Scénář.....	8
Tab. 6. Zavřít projekt - Scénář.....	10
Tab. 7. Změnit stav úkolu - Scénář.....	12

SEZNAM PŘÍLOH

Příloha P 1: OBSAH CD

PŘÍLOHA P 1: OBSAH CD

- / obsahuje MVC framework CakePhp
- /documents obsahuje verze diplomových prací, obrázky obsaženy v diplomové práci, soubor s návrhy v Enterprise Architect, k tomu lite verzi Enterprise Architect
- /app/Model obsahuje implementované řešení databázové vrstvy
- /app/Controller obsahuje implementované řešení pro řízení a zpracování dat od uživatelů do databáze a opačným směrem
- /app/View obsahuje implementované řešení pro zobrazení dat uživatelům a získání dat od uživatelů pomocí formulářů.
- /diplomovaPrace.pdf soubor s diplomovou prací ve formátu pdf.