

WWW portál požadavků na rozvrh

WWW portal of requirements for timetable

Bc. Jaroslav Vaculík

Diplomová práce
2009



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2008/2009

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Jaroslav VACULÍK**
Studijní program: **N 3902 Inženýrská informatika**
Studijní obor: **Informační technologie**

Téma práce: **WWW portál požadavků na rozvrh**

Zásady pro vypracování:

1. Provedte literární průzkum z oblasti PHP a PostgreSQL týkající se dané problematiky.
2. Seznamte se s prací v programu Rozvrh a uveďte jeho stručnou charakteristiku.
3. Vytvořte aplikaci řešící obsluhu požadavků vyučujících na rozvrh.
4. Tato aplikace bude také umožňovat systémové moduly pro správu.
5. Vámi vytvořenou aplikaci umístěte na CD-ROM jako přílohu práce.

Rozsah práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. Lavin, L. (2009): PHP – objektově orientované, koncepty, techniky a kód. Grada, Praha, ISBN 978-80-247-2137-8
2. Kosek, J. (1999): PHP – tvorba interaktivních internetových aplikací. Grada, Praha, ISBN 80-7169-373-1
3. Wyke-Smith, C. (2006): CSS – Využijte kaskádové styly naplno! Computer Press, Praha, ISBN 80-251-1297-7
4. Momjian, B. (2003): PostgreSQL – Praktický průvodce. Computer Press, Praha, ISBN 80-722-6954-2
5. <http://www.postgresql.org/docs/> [28.1.2008] – PostgreSQL documentation
6. <http://www.cervený.sk/> [28.1.2008] – software Rozvrh v4.1

Vedoucí diplomové práce:

Ing. Karel Perůtka, Ph.D.

Ústav řízení procesů

Datum zadání diplomové práce:

20. února 2009

Termín odevzdání diplomové práce:

27. května 2009

Ve Zlíně dne 13. února 2009

prof. Ing. Vladimír Vašek, CSc.
děkan



doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Dokument obsahuje úvod do problematiky tvorby školních rozvrhů, stručnou charakteristiku programu Rozvrh pro tvorbu rozvrhů, návrh systému pro zadávání požadavků na rozvrh pomocí technologií PHP a PostgreSQL a popis tohoto systému.

Klíčová slova: školní rozvrh, požadavky, webový portál, PHP, PostgreSQL, framework, IS, Redakční systém, modul, databáze, UML

ABSTRACT

The document contains an introduction to the issue of making school schedules, brief characteristics of the application Rozvrh for the production schedules, proposal of system for requirements for schedule using technologies PHP and PostgreSQL and a description of this system.

Keywords: school timetable, requirements, web portal, PHP, PostgreSQL, framework, IS, Content management system, module, database, UML

Zde bych chtěl uvést poděkování vedoucímu mé práce Ing. Karlu Perůtkovi Ph.D. za ochotu, trpělivost a spolupráci při tvorbě této diplomové práce. Dále pak Ing. Martinu Kudláčkovi a Ing. Radomíru Chlupovi za cenné informace.

Nakonec bych chtěl poděkovat své přítelkyni za její psychickou podporu a trpělivost.

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.

Ve Zlíně

.....
Podpis diplomanta

OBSAH

ÚVOD	10
I TEORETICKÁ ČÁST	11
1 CÍLE	12
2 PROGRAM ROZVRH	13
2.1 CHARAKTERISTIKA PROGRAMU	13
2.2 TVORBA ROZVRHU	13
2.3 MOŽNOSTI IMPORTU/EXPORTU	14
3 SPECIFIKACE POŽADAVKŮ	16
3.1 OBECNÉ POŽADAVKY NA APLIKACI.....	16
3.1.1 Požadavky plynoucí přímo ze zadání	16
3.1.2 Požadavky potřebné pro běh aplikace	16
3.2 KONKRÉTNÍ POŽADAVKY.....	17
3.2.1 Přihlašování uživatelů	19
3.2.2 Editace osobních údajů.....	20
3.2.3 Zadávání požadavků.....	21
3.2.4 Editace požadavků.....	23
3.2.5 Editace vyučujících	24
3.2.6 Import/export dat.....	25
3.2.7 Zadávání deadlinů	26
3.2.8 Zasílání e-mailových notifikací.....	27
3.2.9 Správa systémových modulů.....	28
II PRAKTICKÁ ČÁST	29
4 ANALÝZA	30
4.1 VOLBA NÁSTROJŮ A TECHNOLOGIÍ	30
4.1.1 XHTML.....	30
4.1.2 CSS.....	31
4.1.3 PHP	31
4.1.4 JavaScript + jQuery	31
4.1.5 Apache s podporou SSL a mod_rewrite.....	31
4.1.6 XML.....	32
4.1.7 CSV	32
4.1.8 PostgreSQL	32
4.1.9 modul Ltree pro PostgreSQL	33
4.1.10 Python.....	33
4.1.11 Doxygen	33
4.1.12 SchemaSpy	34
4.1.13 Graphviz.....	34
4.1.14 Framework	34
4.1.15 Redakční systém.....	35

4.2	ANALÝZA FRAMEWORKU.....	35
4.3	ANALÝZA REDAKČNÍHO SYSTÉMU	38
4.4	ANALÝZA POŽADAVKŮ, REALIZACE PŘÍPADŮ UŽITÍ	39
4.4.1	Přihlašování uživatelů	40
4.4.2	Editace osobních údajů.....	41
4.4.3	Zadávání požadavků.....	42
4.4.4	Editace požadavků.....	44
4.4.5	Editace vyučujících	45
4.4.6	Import/export dat.....	45
4.4.7	Zadávání deadlinů	46
4.4.8	Zasílání e-mailových notifikací.....	46
4.4.9	Správa systémových modulů.....	47
5	NÁVRH SYSTÉMU	50
5.1	PŘÍSTUPOVÁ OPRAVNĚNÍ.....	50
5.1.1	Vyhodnocování přístupových pravidel.....	51
5.2	ČÍSELNÍKY	52
5.2.1	Akademický rok	52
5.2.2	Semestry	52
5.2.3	Členění	52
5.2.4	Formy studia.....	52
5.2.5	Ročníky	52
5.2.6	Studijní obory.....	53
5.2.7	Stavy časového plánu	53
5.2.8	Učebny.....	53
5.2.9	Budovy	53
5.2.10	Místa.....	53
5.2.11	Typy učeben	53
5.2.12	Typy studia	53
5.2.13	Detašovaná pracoviště.....	53
5.2.14	Typy zaměstnance	54
5.2.15	Zařazení	54
5.3	MODULY POŽADAVKŮ	54
5.3.1	Požadavky vyučujících.....	54
5.3.2	Požadavky předmětů	56
6	BEZPEČNOSTNÍ RIZIKA A JEJICH OŠETŘENÍ	57
6.1	ZJIŠŤOVÁNÍ HESEL HRUBOU SILOU	57
6.2	ODPOSLECH INFORMACÍ	57
6.3	SQL INJECTION.....	58
6.4	CROSS-SITE SCRIPTING (XSS).....	59
6.5	SESSION HIJACKING	60
6.6	SESSION FIXATION	60
	ZÁVĚR	61
	ZÁVĚR V ANGLIČTINĚ.....	62

SEZNAM POUŽITÉ LITERATURY.....	63
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	65
SEZNAM OBRÁZKŮ	67
SEZNAM PŘÍLOH.....	68

ÚVOD

Každá škola se setkává s problémem, jak efektivně sestavit rozvrh pro své studenty. Pro tyto účely vzniklo již poměrně mnoho aplikací jak jednodušších freeware tak i složitějších a komplexnějších placených software (např. aSc Rozvrhy, Katedra, ...), které řeší tuto problematiku a jsou schopny automatického generování rozvrhů na základě vstupních parametrů. Většina těchto aplikací je schopna bezproblémově fungovat na všech typech základních a středních škol. Problémy ale nastávají u větších institucí, jako jsou vysoké školy, zpravidla díky rozdílnému fungování těchto institucí a to hlavně v oblasti dostupnosti lidských zdrojů z řad vyučujících.

Zadání této práce vzniklo na základě požadavků rozvrhářů Fakulty aplikované informatiky UTB ve Zlíně. Na této fakultě se pro tvorbu rozvrhů používá program Rozvrh [1]. Při tvorbě rozvrhu však fakultní rozvrhář musí brát v potaz i skutečnosti, že vyučující nemají pevně stanovenou pracovní dobu, ale tato se přizpůsobuje jejich časovým možnostem. Proto zde vznikl požadavek na vytvoření systému, podpůrné aplikace při tvorbě rozvrhů, který by byl schopný na jednom místě udržovat informace o časových možnostech jednotlivých vyučujících, přehledně je zobrazovat a být dostupný online pro všechny vyučující.

I. TEORETICKÁ ČÁST

1 CÍLE

Jak plyne ze zadání, cíl této diplomové práce je vytvoření aplikace ve formě webového portálu řešícího obsluhu požadavků vyučujících na rozvrh. Při analýze a následném návrhu aplikace jsem se ale postupně setkal s drobnějšími vedlejšími cíli, jako je např. již pouhé přihlašování uživatelů do aplikace koncipované tak, aby bylo možné daného uživatele bezpečně identifikovat v rámci UTB. Po zvážení všech požadavků a cílů vyšlo najevo, že výsledná aplikace bude fungovat jako samostatný informační systém schopný shromažďovat, udržovat a spravovat data ve formě požadavků na rozvrh a k nim příslušných datových struktur.

Hlavním cílem je tedy *vytvoření informačního systému* schopného spravovat veškerá požadovaná data.

2 PROGRAM ROZVRH

Program Rozvrh [1], jehož autorem je RNDr. Lubomír Červený, primárně slouží k vytváření rozvrhů základních a středních škol. Nicméně aplikace splňuje všechny požadavky pro tvorbu rozvrhu i na jiných typech škol.

2.1 Charakteristika programu

Program je napsán pro OS Windows 98/2000/XP a kompatibilní systémy. Umožňuje poměrně jednoduché zadání všech dat potřebných pro tvorbu rozvrhu, to jsou seznamy školních budov, učeben, předmětů, vyučujících, tříd a studijních skupin. Při vytváření rozvrhu jsou pak brány v úvahu i individuální požadavky vyučujících a tříd. Mezi další vlastnosti této aplikace patří možnost práce jak se souhrnným tak i s týdenním rozvrhem jednotlivých tříd, předmětů, vyučujících ale i učeben. Lze také vytvářet vícetýdenní rozvrhy. Jsou zde implementovány funkce pro tvorbu rozvrhu ve více budovách s kontrolou času na přesun mezi budovami a funkce pro zpětnou kontrolu vytvořeného rozvrhu a jeho částí.

Aplikace není závislá na prvotním zadání dat, jinými slovy umožňuje přidat např. další učebny do již existujícího rozvrhu.

Program také vyniká možnostmi nastavením tisku veškerých zadaných údajů.

2.2 Tvorba rozvrhu

Před tvorbou rozvrhu je nutné zadat všechna potřebná data a vazby mezi nimi do programu.

Zadání základních dat spočívá v jednoduchém vypsání seznamů tříd, předmětů, vyučujících, učeben, skupin a budov. Ke každému záznamu je pak nutné zadat jeho zkratku, pomocí které se bude identifikovat.

Údaje o vyučování některé třídy (v případě vysoké školy studijního oboru a ročníku) se zadává vyplněním lístku pro danou třídu. Do jednoho lístku je možné přiřadit více studijních skupin, příp. pro každou studijní skupinu vytvořit samostatný lístek. V případě, že je některý předmět společný pro více tříd/studijních skupin, lze v lístku uvést zkratky jednotlivých tříd a skupin. Mezi lístky lze také vytvářet různé druhy vazeb jako např. že

některé předměty nelze vyučovat v po sobě jdoucích hodinách nebo že přednášku nelze mít v ten samý den jako laboratorní cvičení.

Další směrodatné údaje potřebné pro tvorbu rozvrhu se dají zadat vypsáním učební osnovy nebo v podobě úvazku vyučujícího či zadáním počtu hodin pro daný předmět a třídu. Jakým způsobem budou tato data zadávána, již záleží na konkrétním uživateli, který ze způsobů mu více vyhovuje, příp. lze postupy vhodně kombinovat. Tyto druhy vkládání dat a vazeb do systému pak umožňují lepší zpětnou kontrolu správnosti zadání díky více různým pohledům. Existuje zde také automatická kontrola pro kvalifikaci vyučujících v porovnání s jejich již zadanými úvazky.

Samotnou tvorbu rozvrhu jde ovlivnit a konfigurovat dle potřeb. Lze kombinovat automatické generování rozvrhu s ručním zadáváním hodin. Je možné postupovat ve více krocích, tedy např. umístit ručně předměty, které mají pevně stanoven časový plán, poté spustit automatické generování rozvrhu pro určitou skupinu lístků, které obsahují problematicky rozvrhovatelne předměty a nakonec nechat degenerovat rozvrh pro všechny lístky. Automatické generování rozvrhu je možné spustit i v případě, pokud požadujeme změnu pouze některých hodin v již existujícím rozvrhu. Ruční změna rozvrhu je zde umožněna pomocí funkcí drag&drop neboli jednoduchým přesouváním lístků v rozvrhu pomocí myši.

2.3 Možnosti importu/exportu

V programu lze všechny seznamy a vygenerované rozvrhy vyexportovat do formátu HTML pro snadné umístění na webovou prezentaci. Nově (dne 15. 9. 2008) byla také přidána možnost exportu dat do strojově čitelnějšího XML formátu, bohužel možnost importu v XML formátu zde chybí. Tímto se možnosti importu a exportu dat z programu končí. Aplikace Rozvrh sice umožňuje uživatelsky velice pohodlné zadávání a výběr dat z/do programu pomocí copy-paste neboli jednoduchým zkopírováním a vložením dat z/do programů MS Excel a MS Word, nicméně jedná se stále o ruční zadávání bez jakékoliv možnosti automatizace, tedy není tím zaručena žádná pevná struktura dat a bezpečnost přenosu dat z pohledu správnosti a platnosti přenášených údajů.

Jednou z možností automatizace by byl přímý přístup do databáze programu, avšak program nepoužívá žádnou standardní relační, objektovou či jinou databázi, ale data si

udržuje ve vlastních datových strukturách přímo na pevném disku počítače. Je sice pravda, že tento způsob ukládání dat pak nevyžaduje instalaci dalších přídatných systémů pro správu a řízení dat, na druhou stranu snižuje přehlednost kontroly nad daty, ale to už je problém tvůrce programu.

3 SPECIFIKACE POŽADAVKŮ

3.1 Obecné požadavky na aplikaci

Požadavky na aplikaci můžeme rozdělit do dvou hlavních kategorií a to jsou

- Požadavky plynoucí přímo ze zadání
- Požadavky potřebné pro běh aplikace

3.1.1 Požadavky plynoucí přímo ze zadání

Jedná se o požadavky, které jsou vysloveny přímo v zadání diplomové práce.

- **Online dostupnost** – Již přímo v názvu práce je uvedeno „WWW portál“, hlavním kritériem aplikace by tedy měla být její online dostupnost pomocí internetové sítě. Je tomu tak proto, aby k aplikaci mohli přistupovat bezproblémově všichni její budoucí uživatelé.
- **Modularita** – Systém by měl poskytovat moduly pro správu. Tato část zadání vede k vytvoření plně modulárního systému, protože kombinace skriptů pro přímý výpis dat a modulární nadstavby pro správu dat by byla z hlediska dalšího rozvoje nepraktická a nepřehledná.

3.1.2 Požadavky potřebné pro běh aplikace

Tyto požadavky jsou buďto nepřímou odvozeny od zadání nebo vznikly proto, aby buďto zjednodušily práci s aplikací jak na straně běžného uživatele (tj. vyučujícího) tak i na straně administrátora systému, nebo jako rozšíření zpřehledňující celou aplikaci či zvyšující bezpečnost.

- **Přihlašování a správa uživatelů** – Aplikace by měla umožňovat bezpečné a snadné přihlašování uživatelů a správu údajů o uživateli.
- **Uživatelské skupiny, oprávnění** – Jednou z hlavních podmínek pro správnou funkčnost celé aplikace je přítomnost modulu pro správu uživatelských skupin. Do systému budou mít totiž přístup 4 různé skupiny uživatelů a každou tuto skupinu je třeba v systému bezpečně identifikovat. Uživatelé pak umožnit přístup pouze tam,

kam v rámci dané skupiny může, a umožnit mu provádět pouze akce povolené pro jeho uživatelskou skupinu.

- **Maximální jednoduchost** – Tvorba rozvrhů a zadávání požadavků na rozvrh není již triviální věc, proto by aplikace měla být z pohledu přihlášeného uživatele co nejjednodušší a snadno pochopitelná.
- **Intuitivnost ovládní** – Pro snadnou práci by měl být vytvořený systém intuitivní, tj. umožnit uživateli si snadno uvědomit, jak má v systému pracovat.

3.2 Konkrétní požadavky

V této části dokumentu je uveden textový popis toho, co by měl vyvíjený systém umět a jakými funkcemi disponovat.

Mezi základní funkce, které by v systému neměly chybět, patří přihlašování uživatelů s možností definice přístupových oprávnění, bude se totiž jednat o víceuživatelský systém, ve kterém budou mít různí uživatelé možnost spravovat různé systémové moduly. Systém by měl také umožňovat zadání a změnu osobních, a to především kontaktních údajů. Hlavní úlohou je pak zpracovávání požadavků na rozvrh, je tedy nutnost, aby zde byly funkce na zpracování a správu požadavků. Vůči systému zde bude existovat pět typů přístupů, jsou to:

- **nepřihlášený uživatel**
- **vyučující**
- **rozvrhář**
- **tajemník**
- **administrátor**

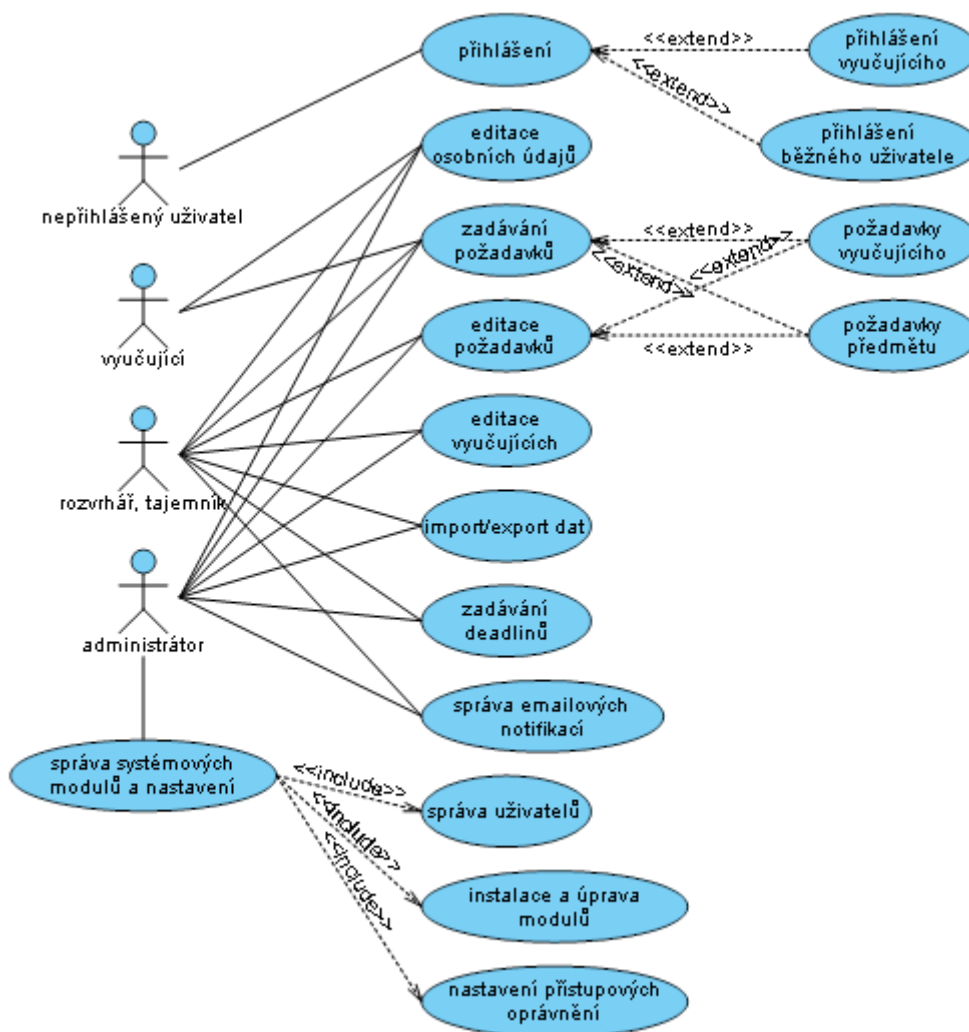
Z pohledu přístupových oprávnění a povinností jsou přístupy typu **rozvrhář** a **tajemník** vůči systému totožné, proto budou brány jako jeden typ přístupu. Takto získáváme čtyři typy přístupů. Tato členění se odborně nazývají role. Každá z rolí má v systému jinou úlohu a jiné možnosti přístupu. Pro role **rozvrhář**, **tajemník** a **administrátor** musí systém poskytovat navíc možnost úpravy (editace) záznamů o vyučujících a požadavcích na rozvrh a pak možnost zadávat konečná data pro zadání požadavků na rozvrh, tzv. **deadliny**. Spolu s tím souvisí funkce pro notifikaci vyučujících o povinnosti zadat své požadavky na rozvrh.

Nakonec bude mít role administrátor navíc na starost správu systémových modulů, tj. správu uživatelů a jejich přístupových oprávnění a instalace nových příp. úpravy již nainstalovaných modulů.

Role budou v návrhu a implementaci systému reprezentovány uživatelskými skupinami.

Obrázek níže (Obr. 1.) je tzv. *use case diagram* nebo use case model, česky diagram případů užití. Znárodňuje případy užití systému pro nalezené role.

Use case diagram je zobrazení dynamické (funkční) struktury systému z pohledu uživatele. Primárně je určen k definici chování systému, aniž by odhaloval jeho vnitřní strukturu. Jinými slovy je to soubor scénářů pro používání systému. Každý scénář pak obsahuje sekvenci (posloupnost) událostí, které v jeho rámci probíhají (včetně případných variant) a popis interakce (komunikace) mezi uživatelem (aktorem) a systémem.



Obr. 1. Diagram případů užití

3.2.1 Přihlašování uživatelů

Přihlašování bude probíhat dvěma způsoby. Z pohledu uživatele jsou tyto způsoby téměř totožné, ale z pohledu systému se jedná o dva různé scénáře, proto budou uvedeny oba.

První scénář se týká přihlašování běžného uživatele systému. Přihlášení bude probíhat pomocí formuláře, kde uživatel zadá své přihlašovací jméno a heslo a odešle formulář pro zpracování na straně aplikačního serveru. Systém ověří správnost zadaných údajů, a pokud se bude jednat o platné údaje, provede přihlášení uživatele.

Případ užití: Přihlášení běžného uživatele	ID: UC1
Účastníci: Běžný uživatel systému (nepřihlášený)	
Vstupní podmínky: Uživatel není přihlášen.	
Tok událostí: <ol style="list-style-type: none"> 1. Uživatel požádá systém o zobrazení přihlašovacího formuláře běžného uživatele. 2. Systém zobrazí formulář pro zadání přihlašovacích údajů. 3. Uživatel zadá své přihlašovací jméno a heslo a odešle formulář zpět systému. 4. Systém ověří správnost zadaných údajů. 5. Pokud jsou údaje správné, provede systém přihlášení uživatele. 	
Následné podmínky: Uživatel je přihlášen v systému.	

Druhý scénář pro přihlášení vyučujícího bude probíhat obdobně jenom s tím rozdílem, že přihlašovací formulář pro vyučující bude obohacen o zadání kontextu. Kontext je přidělen každému vyučujícímu v rámci fakulty, ve které je zaměstnán. Po zadání všech potřebných údajů pro přihlášení a odeslání dat, systému zkontroluje existenci uživatele dle zadaných parametrů pomocí LDAP technologie univerzitního serveru. Pokud ověření proběhne v pořádku, systém přihlásí vyučujícího pod speciálním uživatelským účtem pro vyučující. Informace o přihlášeném uživateli pak budou doplněna daty o vyučujícím dohledaným z databáze vyučujících, kterou bude systém udržovat. Pokud se data o vyučujícím v databázi nenajdou, bude vytvořen nový záznam o vyučujícím do této databáze.

Případ užití: Přihlášení vyučujícího	ID: UC2
Účastníci: Vyučující (nepřihlášený)	
Vstupní podmínky: Uživatel není přihlášen.	
Tok událostí: <ol style="list-style-type: none"> 1. Uživatel požádá systém o zobrazení přihlašovacího formuláře pro vyučující. 2. Systém zobrazí formulář pro zadání přihlašovacích údajů vyučujících. 3. Vyučující zadá své přihlašovací jméno, heslo a kontext a odešle formulář zpět systému. 4. Systém ověří existenci vyučujícího pro zadané přihlašovací údaje pomocí univerzitního serveru. 5. Pokud vyučující existuje, provede přihlášení pomocí speciálního uživatelského účtu pro vyučující. 6. Systém vyhledá přihlašujícího se vyučujícího ve vlastní databázi vyučujících, a pokud nenalezne záznam, pak vytvoří nový s vyučujícím. 7. Systém doplní informace o přihlášeném uživateli daty z databáze vyučujících. 	
Následné podmínky: Vyučující je přihlášen v systému.	

3.2.2 Editace osobních údajů

Editace osobních údajů bude přístupná všem uživatelům, tedy jak běžným uživatelům systému tak i vyučujícím. Rozdíl však může být v obsahu jednotlivých údajů pro editaci vyučujícího a editaci běžného uživatele. Vyučující zde bude mít povinné pole pro svolení k e-mailové komunikaci. Další rozdíly pak už nejsou tak relevantní a z pohledu užití nemají význam.

Případ užití: Editace osobních údajů	ID: UC3
Účastníci: Vyučující Rozvrhář, tajemník Administrátor	
Vstupní podmínky: Uživatel je přihlášen.	
Tok událostí: 1. Uživatel zažádá systém o zobrazení osobních údajů přihlášeného uživatele. 2. Systém zobrazí formulář s daty o přihlášeném uživateli. 3. Uživatel provede změnu svých údajů v zobrazeném formuláři a ten následně odešle. 4. Systém provede změnu údajů o uživateli.	
Následné podmínky: Uloženy změny osobních údajů přihlášeného uživatele.	

3.2.3 Zadávání požadavků

Požadavky na rozvrh budou dvojího druhu a to *požadavky vyučujících* a *požadavky předmětů*. Možnost zadávat požadavky budou mít všichni přihlášení uživatelé systému.

Zadání požadavků vyučujících bude probíhat tak, že uživatel zvolí akademický rok a semestr, pro který bude daný požadavek platný a zadá všechna potřebná data s možností vyplnění týdenního kalendáře vyučujícího. Rozdíl v zadávání z pohledu uživatelských oprávnění bude v tom, že přihlášený vyučující bude moci zadávat pouze své požadavky, zatímco ostatní běžní uživatelé systému budou povinni ke každému požadavku zadat vyučujícího, ke kterému se vkládáný požadavek vztahuje. Vkládání požadavků vyučujících bude probíhat dle následujícího scénáře.

Případ užití: Zadávání požadavků vyučujících	ID: UC4
Účastníci: Vyučující Rozvrhář, tajemník Administrátor	
Vstupní podmínky: Uživatel je přihlášen.	
Tok událostí: 1. Uživatel nastaví v systému výběr akademického roku a semestru (pokud se jedná o jiného uživatele než uživatele patřícího do skupiny „vyučující“, pak také nastaví volbu vyučujícího). 2. Uživatel vyplní týdenní kalendář zvoleného vyučujícího. 3. Uživatel zadá doplňující textové požadavky tak, že požádá systém o jejich vytvoření a vyplní poté požadované údaje.	
Následné podmínky: Do systému byly vloženy nové požadavky vyučujícího na rozvrh.	

Zadání požadavků předmětů bude probíhat také vybráním konkrétního akademického roku semestru. Uživatel poté vybere předmět, který je dostupný pro daný rok a semestr a vyplní požadované údaje.

Případ užití: Zadávání požadavků předmětů	ID: UC5
Účastníci: Vyučující Rozvrhář, tajemník Administrátor	
Vstupní podmínky: Uživatel je přihlášen.	
Tok událostí: 1. Uživatel nastaví v systému výběr akademického roku a semestru a předmětu dostupného pro daný rok a semestr. 2. Uživatel požádá systém o vytvoření nového prázdného požadavku na předmět. 3. Uživatel zadá požadované údaje pro požadavek na předmět do systému. 4. Systém uloží nový požadavek předmětu.	
Následné podmínky: Do systému byly vloženy nové požadavky předmětu.	

3.2.4 Editace požadavků

Editace požadavků vyučujících a požadavků předmětů bude zpřístupněna pouze rolím rozvrhář, tajemník a administrátor. Bude probíhat tak, že si uživatel vybere konkrétní požadavek a tomu změni nevyhovující parametry. U požadavků vyučujících pak bude při volbě konkrétního vyučujícího možnost změny i jeho týdenního kalendáře.

Editace požadavku vyučujícího bude probíhat podle následujícího scénáře:

Případ užití: Editace požadavků vyučujících	ID: UC6
Účastníci: Rozvrhář, tajemník Administrátor	
Vstupní podmínky: Uživatel je přihlášen.	
Tok událostí: <ol style="list-style-type: none"> 1. Uživatel provede výběr požadavku vyučujícího k aktualizaci. 2. Pokud uživatel při výběru požadavku zadá jako parametr filtrování záznamů konkrétního vyučujícího, systém automaticky zobrazí týdenní kalendář vyučujícího. 3. Uživatel může změnit týdenní kalendář vyučujícího, pokud je kalendář zobrazen. 4. Uživatel zažádá systém o editaci záznamu požadavku. 5. Systém zobrazí formulář pro aktualizaci požadavku. 6. Uživatel provede aktualizaci dat požadavku a odešle zobrazený formulář aktualizace. 7. Systém aktualizuje záznam požadavku vyučujícího. 	
Následné podmínky: Je aktualizován záznam o požadavku na rozvrh vyučujícího a týdenní kalendář vyučujícího.	

Editace požadavků předmětu:

Případ užití: Editace požadavků předmětů	ID: UC7
Účastníci: Rozvrhář, tajemník Administrátor	
Vstupní podmínky: Uživatel je přihlášen.	
Tok událostí: 1. Uživatel provede výběr záznamu požadavku předmětu pro aktualizaci. 2. Uživatel zažádá systém o editaci záznamu požadavku předmětu. 3. Systém zobrazí formulář pro aktualizaci požadavku předmětu. 4. Uživatel provede aktualizaci dat a odešle zobrazený formulář aktualizace. 5. Systém aktualizuje záznam požadavku předmětu.	
Následné podmínky: Je aktualizován záznam požadavku předmětu.	

3.2.5 Editace vyučujících

Editace vyučujících není nic jiného než aktualizace záznamů o vyučujících v modulu pro správu vyučujících. Systém tedy bude obsahovat mechanismus, jak vyhledat a zpřístupnit záznamy o vyučujících pro aktualizaci rolím „Administrátor“ a „Rozvrhář, tajemník“. Uživatel si pak jen vybere záznam k aktualizaci, upraví jej do potřebné podoby a systém aktualizuje daný záznam vyučujícího.

Případ užití: Editace vyučujících	ID: UC8
Účastníci: Rozvrhář, tajemník Administrátor	
Vstupní podmínky: Uživatel je přihlášen.	
Tok událostí: 1. Uživatel vyhledá záznam o vyučujícím, který chce aktualizovat. 2. Uživatel zažádá systém o editaci vybraného záznamu vyučujícího. 3. Systém uživateli zobrazí formulář pro aktualizaci vybraného záznamu vyučujícího. 4. Uživatel provede změnu údajů ve formuláři pro aktualizaci a formulář odešle. 5. Systém aktualizuje záznam vyučujícího dle parametrů zadaných uživatelem.	
Následné podmínky: Byl aktualizován záznam o vyučujícím.	

3.2.6 Import/export dat

Systém bude obsahovat funkce pro export dat jednotlivých modulů. Export konfigurovatelný a bude možné jej nastavit pro každý konkrétní modul zvlášť. Export bude probíhat do formátu CSV, tedy do formátu čitelného pro běžně používané tabulkové procesory *MS Excel* a *OpenOffice Calc*. Import dat do systému bude primárně omezený na ruční zadávání, systém však bude obsahovat možnost importu dat pro konkrétní modul pomocí CSV souboru s předem pevně definovanou strukturou. Pro usnadnění vkládání údajů o předmětech bude systém obsahovat možnost importu dat o předmětech z XML souboru vytvořeného exportem z informačního systému Stag používaného na UTB.

Případ užití: Export dat	ID: UC9
Účastníci: Rozvrhář, tajemník Administrátor	
Vstupní podmínky: Uživatel je přihlášen.	
Tok událostí: 1. Uživatel vybere modul, ze kterého budou exportována data. 2. Uživatel provede selekci požadovaných záznamů modulu pomocí filtru (vyhledávání) záznamů modulu. 3. Uživatel zadá příkaz pro export selektovaných záznamů. 4. Systém vygeneruje CSV soubor s požadovanými záznamy modulu a nabídne jej uživateli ke stažení.	
Následné podmínky: Systém vygeneroval CSV soubor s daty modulu.	

3.2.7 Zadávání deadlinů

Deadline je označení pro datum sloužící jako nejzazší termín pro zadání požadavků na rozvrh do systému. Toto datum se vztahuje ke konkrétnímu akademickému roku a semestru. Systému bude umožňovat spravovat číselník takových deadlinů a ke každému přiřadit akademický rok a semestr, ke kterému se vztahuje.

Případ užití: Zadávání deadlinů	ID: UC10
Účastníci: Rozvrhář, tajemník Administrátor	
Vstupní podmínky: Uživatel je přihlášen.	
Tok událostí: 1. Uživatel požádá systém o vytvoření nového deadlinu. 2. Systém zobrazí uživateli formulář pro vyplnění podrobných informací o deadlinu. 3. Uživatel zadá informace o deadlinu v podobě kalendářního data, akademického roku a semestru a formulář s daty odešle zpět systému. 4. Systém ověří platnost zadaných dat a uloží je jako nový deadline.	
Následné podmínky: Do systému byl přidán nový deadline.	

3.2.8 Zasílání e-mailových notifikací

E-mailové notifikace úzce souvisí se zadáváním deadlineů, každá notifikace bude totiž vázána na konkrétní deadline. Uživatel vytvoří novou notifikaci vyplněním předmětu e-mailu, přiřazením deadlineu a zadáním kalendářního data, kdy má být odeslána. Uživatel bude mít také možnost zadat podrobnější zprávu pro příjemce. Po vytvoření nové notifikace ji systém automaticky odešle dle zadaného kalendářního data. Před odesláním bude z notifikace vygenerováno tělo e-mailu doplněné o relevantní informace získané z detailu notifikace, tedy o konečné datum pro zadání požadavků a k jakému akademickému roku a semestru se toto datum vztahuje. Každou notifikaci bude možné odeslat pouze jednou. Odeslání bude umožněno také přímo na žádost uživatele. Systém bude také ukládat výsledek odeslání, tedy jestli bylo odeslání notifikace úspěšně provedeno nebo nikoli. Všechny e-mailové notifikace o povinnosti zadání požadavků na rozvrh budou zasílány všem vyučujícím, kteří ve svém uživatelském profilu dali souhlas s e-mailovou komunikací.

Případ užití: Zasílání e-mailových notifikací	ID: UC11
Účastníci: Rozvrhář, tajemník Administrátor	
Vstupní podmínky: Uživatel je přihlášen.	
Tok událostí: <ol style="list-style-type: none"> 1. Uživatel zažádá systém o vytvoření nové e-mailové notifikace. 2. Systém vytvoří nový záznam notifikace a zobrazí uživateli formulář pro zadání dat nové notifikace. 3. Uživatel vyplní předmět e-mailu notifikace, datum pro odeslání, deadline, ke kterému se notifikace vztahuje příp. i podrobnější zprávu pro příjemce e-mailové notifikace a formulář s daty odešle na zpracování systému. 4. Systém ověří platnost zadaných dat a aktualizuje záznam notifikace. 5. Je zadán příkaz k odeslání e-mailové notifikace buďto uživatelem anebo automaticky systémem ve stanovený čas pro odeslání. 6. Systém kontroluje podmínky pro odeslání a zašle e-mailovou notifikaci všem vyučujícím, kteří ve svém uživatelském profilu dali souhlas k e-mailové komunikaci. 	
Následné podmínky: Systém zaslal vyučujícím e-mailovou notifikaci o deadlineu pro zadání požadavků na rozvrh.	

3.2.9 Správa systémových modulů

Správa systémových modulů je přístupná pouze roli „Administrátor“. Pod tímto názvem se skrývají čtyři další termíny, jsou to:

- *Správa uživatelů*
- *Správa uživatelských skupin*
- *Instalace a úprava modulů*
- *Nastavení přístupových oprávnění*

Jde tedy o tři různé scénáře užití systému.

Správa uživatelů obsahuje změnu údajů o uživateli, tj. např. jeho jméno, přihlašovací jméno, heslo, atd. a možnost aktivace/deaktivace uživatelských účtů.

Správa uživatelských skupin obsahuje změnu názvu a podrobnějšího popisu skupiny a možnost aktivace/deaktivace uživatelské skupiny.

Instalace a úprava modulů znamená možnost přidání/odstranění příp. změnu již nainstalovaných modulů v systému. Uživatel tak může přidat nové potřebné moduly nebo upravit systémové nastavení stávajících.

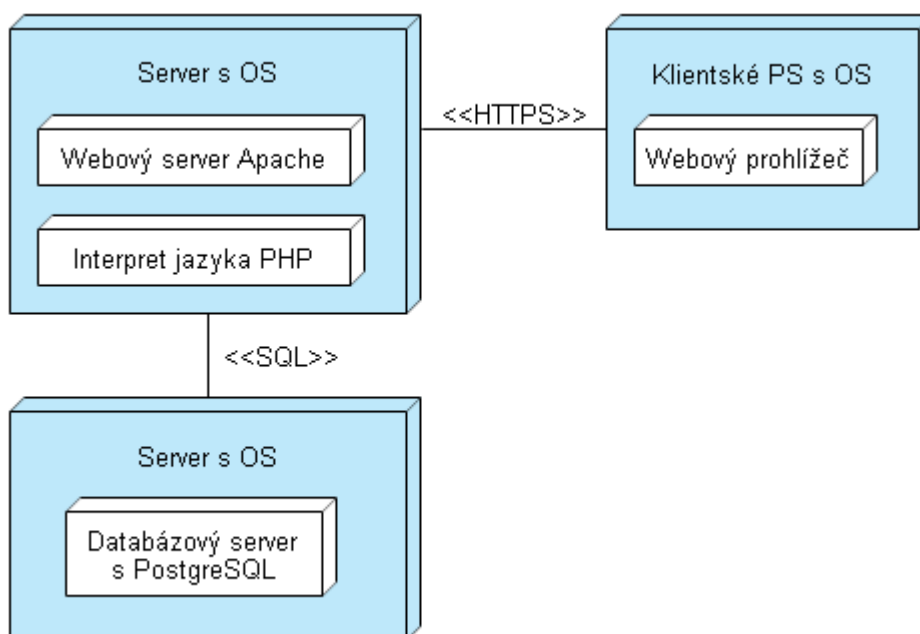
Nastavení přístupových oprávnění je velmi důležitá vlastnost systému. Každý uživatel totiž může mít v systému jiná privilegia. Jde především o bezpečnostní opatření pro zachování konzistence dat v databázi. Kdyby měl např. nezkušený uživatel systému stejná oprávnění jako administrátor, mohl by třeba nechtěně odstranit nebo změnit funkčnost u nainstalovaných modulů. Nastavení přístupových oprávnění bude prováděno vždy pro konkrétní uživatelské skupiny v kombinaci modul, prováděná akce, jazyková mutace a operace.

II. PRAKTICKÁ ČÁST

4 ANALÝZA

System bude modulární, proto bude vhodné při jeho návrhu a následné implementaci použít objektového návrhu. Každý modul pak bude moci představovat určitou třídu, která bude poskytovat funkcionalitu pro správu daného modulu. Objektový návrh má pak i tu výhodu, že je možné využívat dědičnosti jednotlivých tříd a tím přenášet již jednou implementované vlastnosti a funkce.

Předtím, než začnu s analýzou jednotlivých částí systému, bylo by vhodné vysvětlit, jakým způsobem bude probíhat komunikace a výměna dat mezi uživateli a systémem. Obrázek níže (Obr. 2.) představuje základní architekturu systému.



Obr. 2. Základní architektura systému

4.1 Volba nástrojů a technologií

Pro podrobnější analýzu je dobré si uvědomit, s jakými technologiemi bude probíhat vývoj systému a jaká omezení z tohoto plynou. V této podkapitole je uveden seznam nástrojů, které budou použity při vývoji.

4.1.1 XHTML

XHTML je zkratka z anglického eXtensible Hypertext Markup Language neboli volně přeloženo rozšiřitelný hypertextový značkovací jazyk. Jedná se o značkovací jazyk

vyvinutý W3C konsorciem pro tvorbu hypertextových dokumentů v prostředí WWW, proto bude tento jazyk použit jako prostředek pro prezentaci informací uživatelům systému.

4.1.2 CSS

Kaskádové styly (Cascading Style Sheets) jsou dnes velice rozšířeným prostředkem pro popis způsobu zobrazení dokumentů napsaných v jazycích HTML, XHTML a XML. Tento jazyk umožňuje změnit vzhled hypertextového dokumentu, aby se v něm mohli lidé lépe orientovat. Tato technologie umožňuje oddělit vzhled dokumentu od jeho informačního obsahu, je tedy vhodný i pro použití v této práci pro vyvíjený systém.

4.1.3 PHP

PHP (Hypertextový preprocesor) je skriptovací programovací jazyk, který lze spouštět i z příkazové řádky, nicméně je primárně určen pro generování dynamických internetových stránek. Hlavní výhodou jazyka je, že je hojně rozšířen na hostingových serverech, je dobře dokumentován a lze pro něj nalézt velkou spoustu již hotových projektů a ukázkových kódů. Výhody zde uvedené jsou pouze ty hlavní, výhod jazyka je samozřejmě mnohem více, proto je vhodný jako implementační jazyk systému.

4.1.4 JavaScript + jQuery

JavaScript je objektově orientovaný skriptovací programovací jazyk. V oblasti WWW se používá především pro „oživení“ stránky pomocí různých efektů a pro zvýšení funkčnosti prezentovaných dokumentů.

jQuery je JavaScriptový framework sloužící jako knihovna funkcí pro zjednodušení manipulace s (X)HTML dokumentem, práci s událostmi, AJAXem a zefektivnění vývoje webových aplikací. Pro tuto knihovnu je napsáno velké množství pluginů s otevřeným programovým kódem, které napomáhají při tvorbě DHTML dokumentů.

4.1.5 Apache s podporou SSL a mod_rewrite

Apache je název dnes velmi populárního webového serveru, který v současné době generuje většinu internetových stránek pro prohlížeče. Jedná se o multiplatformní produkt s otevřeným kódem pro Linux, BSD, Windows a další systémy.

SSL je zkratka anglického „*Secure Sockets Layer*“. Jedná se o protokol (vrstvu), která je vložena mezi transportní a aplikační vrstvu ISO/OSI modelu a poskytuje autentizaci a zabezpečení komunikace šifrováním mezi komunikujícími stranami.

Mod_rewrite je prepisovací engine v podobě modulu pro webový server Apache. Webový server předává tomuto modulu url adresy požadované jeho klienty a modul je vyhodnocuje pomocí skriptu s pravidly uloženým v souboru `.htaccess`. Pomocí těchto pravidel lze provádět prepisování url adres na jiné, předávat řízení patřičným skriptům, přesměrovávat, nastavovat stavové kódy apod.

4.1.6 XML

Značkovací jazyk XML byl vyvinut a standardizován W3C konsorciem. Jazyk je obecný a umožňuje vytvářet konkrétní značkovací jazyky pro různé účely a různá data. Jazyk byl vyvinut zejména pro výměnu a popis dat mezi aplikacemi a pro publikování dokumentů.

Jedná se o obecný formát popisu dat vhodný k výměně dat, tj. např. pro import či export dat do různých systémů.

4.1.7 CSV

Formát CSV neboli *Comma-Separated Values* (hodnoty oddělené čárkami) je jednoduchý souborový formát určený pro výměnu tabulkových dat. Soubor, který je ve formátu CSV, je sestaven z řádků, na kterých jsou jednotlivé položky odděleny čárkami. Položky pak mohou být uzavřeny do uvozovek, což umožňuje, aby položka obsahovala i samotný oddělovač příp. víceřádkovou hodnotu. Pokud se v hodnotě takové položky vyskytuje znak uvozovka, pak je tato zdvojena. Oddělovačem položek nemusí být nutně čárka, ale používá se i např. středník nebo tabulátor.

Jedná se o velice jednoduchý formát dat, který lze použít pro export či import dat z/do systému, pokud není potřeba složitě popisovat strukturu exportovaných/importovaných dat.

4.1.8 PostgreSQL

Objektově-relační databázový systém, který lze instalovat na všechny známější operační systémy. Tento databázový systém se vyznačuje podporou většiny datových typů standardů SQL92 a SQL99 a podporou datových typů BLOB a CLOB. Další klady jsou plná podpora

standardů ANSI-SQL 92/99, podpora poddotazů a transakčního zpracování, podpora integritních (primárních, cizích, testovacích, NOT NULL i unikátních) omezení, databázových triggerů, pohledů a množství druhů indexace. Největším kladek jsou však vlastnosti jako podpora v množství programovacích jazyků, zpracování výjimek a notifikací, různá rozšíření systému (např. prostorové rozšíření pro GIS) a integrovaná rozhraní pro možnost psaní vlastních procedur v jazycích Java, Perl, Python, Ruby, Tcl, C/C++ a PL/pgSQL, který je obdobný jako jazyk PL/SQL známý z databázového systému Oracle.

4.1.9 modul Ltree pro PostgreSQL

Jedná se o modul obsahující datové typy, metody indexované přístupové metody, operátory, funkce a dotazy pro manipulaci se stromovými strukturami. Umožňuje efektivní a rychlé vyhledávání v těchto strukturách. Je vhodný např. pro vytváření a řazení stromu kategorií.

Od verze PostgreSQL 8.3 je integrován do instalace tohoto databázového systému.

4.1.10 Python

Jedná se o interpretovaný objektově orientovaný programovací jazyk. Python je víceparadigmatický (neboli hybridní) jazyk, který umožňuje jednak objektově orientované programování, procedurální a částečně i funkcionální. Python se vyznačuje jednoduchostí, produktivností při psaní aplikací, přenositelnost programových kódů, množství rozšiřujících knihoven a v celku dobrá výkonnost (oproti PHP 3x až 5x vyšší).

Zejména z hlediska výkonnosti je tento jazyk vhodným řešením pro zpracování objemnějších XML dokumentů, se kterými mívá PHP, díky různým paměťovým a časovým omezením běhu skriptu, problémy.

4.1.11 Doxygen

Ke každému projektu by měla existovat dokumentace. V případě softwarových projektů by ke standardní dokumentaci měla existovat i dokumentace programová. Jde o dokumentaci, která popisuje jednotlivé funkce systému pro lepší orientaci vývojářů při opravách či rozvíjení systému.

Doxygen je volně dostupný univerzální nástroj, který dokáže při zachování jistých pravidel vygenerovat dokumentaci k programu přímo z jeho zdrojových kódů a to do mnoha různých formátů včetně HTML. Je určen pro použití v mnoha různých jazycích včetně PHP a Python, proto je vhodný pro tuto aplikaci jako nástroj pro generování programové dokumentace.

4.1.12 SchemaSpy

Konzolová aplikace napsaná v jazyce *Java*, která dokáže analyzovat metadata databázového schématu a vygenerovat jejich grafickou reprezentaci ve formátu zobrazitelným internetovým prohlížečem.

Tento nástroj je vhodný jako pomocná aplikace při tvorbě, opravách a rozšiřování systému, jelikož dokáže přehledně graficky znázornit strukturu databáze a závislosti mezi databázovými objekty.

4.1.13 Graphviz

Graphviz je volně šiřitelný software určený pro automatické generování grafů. Tento program bere na vstupu textové popisy grafů zapsané pomocí jednoduchého textového jazyka a s přihlédnutím na další vstupní parametry z nich automaticky generuje grafové závislosti.

Program je využívám generátorem programové dokumentace *Doxygenem* pro vizualizaci závislostí tříd a také aplikací *SchemaSpy* pro vizualizaci relací databázových objektů.

4.1.14 Framework

Pro implementaci systému je přinejmenším vhodné uvažovat o použití nějakého frameworku. Framework lze definovat jako nějakou softwarovou strukturu či aplikační rámec daného prostředí (zpravidla programovacího jazyka), který slouží jako podpora při programování a vývoji softwarových projektů. Tento rámec může obsahovat soubory knihoven, programů, návrhových vzorů, programových postupů případně jiných užitečných funkcionalit, které slouží k usnadnění a urychlení vývoje. Framework si bere za úkol odstínit návrháře a vývojáře od často se opakujících problémů, které by jinak museli řešit, a umožnit jim soustředit se na řešení samotného zadání.

Framework se dělí na dvě části, tzv. *frozen spots* a *hot spots* neboli pevné součásti a variabilní součásti frameworku. *Frozen spots* jsou neměnné součásti, které zůstávají neměnné při jakémkoliv použití frameworku, obsahují tedy základní vlastnosti, funkce, komponenty a vztahy mezi nimi, které daný framework nabízí. *Hot spots* jsou oproti tomu ty komponenty, jež je možné využít k odprogramování vlastní funkcionality.

V oblasti webových aplikací je dnes frameworků poměrně hodně, např. CakePHP, Prado, Symfony, Zend, atd. Výběr toho správného závisí zpravidla na druhu vyvíjené aplikace a způsobu použití. Pro většinu frameworků také existuje množství rozšíření a redakčních systémů umožňujících vývojáři se soustředit opravdu jen na jeho konkrétní úkol.

Další možností by bylo použití *redakčního systému*. Redakční systém je sám o sobě zpravidla postaven na nějakém frameworku a nabízí navíc další funkcionality pro správu dat.

4.1.15 Redakční systém

Pro implementaci požadovaného informačního systému se použití redakčního systému jako aplikační základny jeví jako velice vhodná volba. Redakční systémy ze své podstaty již totiž obsahují část potřebné funkcionality pro provádění úprav v systému a moduly pro správu uživatelů, uživatelských skupin a uživatelských oprávnění. Otázkou pak jen je, jaký redakční systém zvolit. Existují poměrně kvalitní redakční systémy jako jsou Joomla, Drupal, phpRS a další.

Jelikož se v oblasti této problematiky již delší dobu pohybují a to ne jen jako běžný uživatel, nýbrž jako programátor a vývojář, rozhodl jsem se vytvořit vlastní redakční systém, který by byl vystaven s použitím výše uvedených technologií a splňoval všechny požadavky kladené na vytvoření webového portálu pro zadávání požadavků na rozvrh. Redakční systém bude vyvinut na frameworku, který bude napsán natolik obecně, že umožní vytváření i jiných webových aplikací, bude jej tedy možné použít i na jiné projekty.

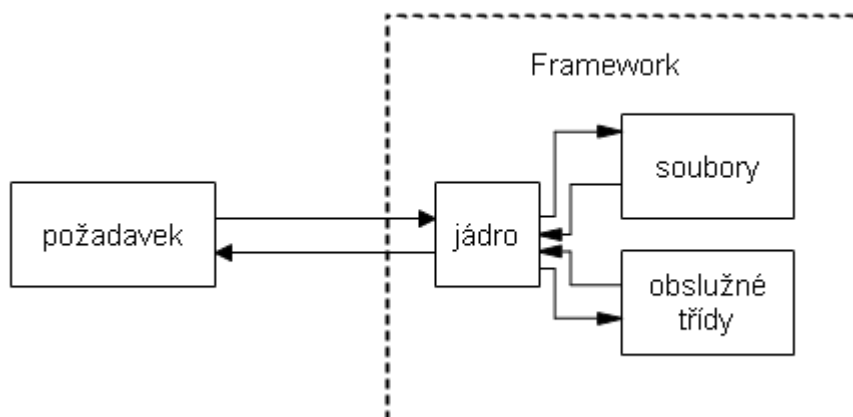
4.2 Analýza frameworku

V předchozí podkapitole při volbě požadavků jsem zvolil tvorbu vlastního frameworku. V této podkapitole se tedy budu věnovat analýze požadavků a funkcí, které musí framework splňovat a poskytovat.

Framework bude plně objektový v rámci možností jazyka PHP. Vedle vestavěných PHP funkcí bude poskytovat veškerou funkcionalitu k tvorbě obecných webových aplikací. Aplikace bude spouštěna vytvořením jádra frameworku (instance třídy *Core*) a voláním metody *run* pro spuštění běhu webové aplikace. Framework bude postaven na těchto principech:

- Bude existovat třída *Object* a všechny třídy vytvořené v rámci frameworku budou odvozeny z této třídy anebo z některého z jejich potomků. Toto omezení se nebude týkat třídy jádra *Core*, samotné třídy *Object* a tříd výjimek.
- Definice tříd a jsou dohledány a automaticky načteny ve chvíli, kdy vznikne první požadavek na vytvoření instance třídy. Není proto nutné includovat soubory s definicemi tříd, pokud daná třída splňuje náležitosti, které framework vyžaduje. Jádro frameworku také zajistí automatické načtení a provedení konfigurace nové instance třídy.
- Každá třída může obsahovat svoji vlastní konfiguraci, která je dědičná a je možné ji kdykoliv upravit, nebo doplnit.
- Všechny URL požadavky budou zachyceny a předány jádru, které zabezpečí vytvoření příslušné třídy pro obsluhu požadavku. Jádro bude tedy mapovat požadavky na obslužné třídy. Oproti klasickým PHP aplikacím teda nebudou požadavky předávány konkrétním skriptům. Výjimku bude tvořit předávání binárních souborů (např. obrázků, PDF dokumentů atd.).
- Výpisy varování a chyb (warnings, errors), které může PHP vypisovat při běhu aplikace, budou transformovány na výjimky a ty následně vyhodnocovány.
- Všechny zdrojové kódy budou v kódování UTF-8.

Obsluha URL požadavku bude probíhat dle schématu *Obr. 3*.



Obr. 3. Obsluha URL požadavku

Framework bude také poskytovat:

- Podporu více jazyků – toto bude realizováno pomocí lokalizační knihovny Gettext. Knihovna ani metody podpory více jazyků zde nebudou popsány, jelikož nejsou pro tuto diplomovou práci zapotřebí, jsou zde však uvedeny pro úplnou specifikaci vyvíjeného frameworku.
- Podporu pro ladění a testování aplikací.
- Podporu pro zaslání e-mailů.
- Podporu pro vytváření URL adres z textových řetězců.
- Načítání parametrů předaných aplikaci pomocí GET a POST.
- Základní obecnou předlohu pro implementaci webové aplikace a také předlohu pro vytváření nejznámějšího druhu webové aplikace – hypertextového dokumentu
- Funkcionalitu pro cachování a správu doplňkových souborů tříd. Bude se především jednat o CSS a JavaScriptové soubory.
- Třidu pro práci s databází, která bude přístupná všem objektům jako atribut třídy.
- Třidu pro práci s formuláři (zejména automatizovaný výpis prvků formuláře na základě předložených vstupních parametrů).
- Datové typy – každý datový typ bude vytvořena samostatná třída, která zajistí jeho správné vypisování ve formulářích a ošetření hodnot při ukládání do databáze.

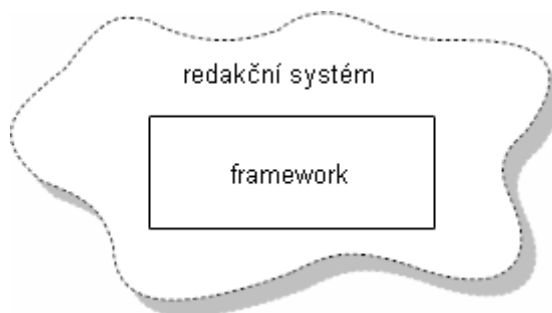
Toto byla stručná analýza vyvíjeného frameworku, aby měl čtenář hrubou představu o jeho možnostech.

4.3 Analýza redakčního systému

Zvolil jsem implementaci vlastního redakčního systému, v této podkapitole budou tedy popsány a analyzovány požadavky, které bude systém splňovat.

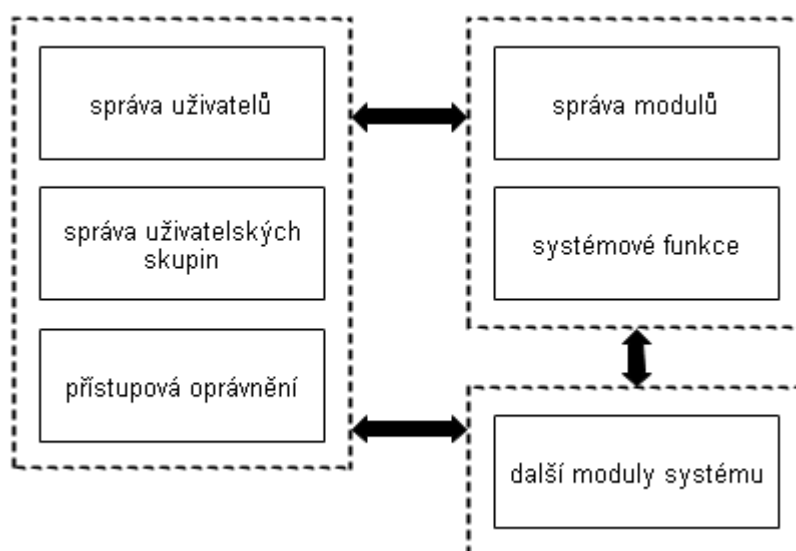
Redakční systém bude postaven na vyvíjeném frameworku a bude využívat jeho výhod (*Obr. 4.*). Samotný redakční systém bude tvořit:

- Instalace databáze redakčního systému obsahující soubor tabulek, datových typů, procedur, databázových triggerů a základních dat pro fungování systému.
- Podpora více jazyků – jednak bude využíváno rozhraní frameworku pro lokalizační knihovnu Gettext a pak bude implementována možnost lokalizací dat spravovaných moduly redakčního systému.
- Metody pro automatickou instalaci databáze systému dle nastavené konfigurace.
- Základní kostra modulu, která bude sloužit jako předloha pro vytváření modulů. Tato bude také realizovat běžné operace, které budou s moduly prováděny.
- Třída pro obsluhu tabulek a databázových datových typů modulu.
- Modul pro správu uživatelů.
- Funkce pro bezpečné přihlášení a odhlášení uživatele.
- Modul pro správu uživatelských skupin.
- Modul pro nastavování přístupových oprávnění uživatelským skupinám.
- Modul pro správu modulů (instalace, reinstalace, zobrazování v seznamu sekcí).



Obr. 4. Vztah redakčního systému a frameworku

Samotný návrh architektury redakčního systému je znázorněn na Obr. 5.



Obr. 5. Architektura jádra redakčního systému a modulů

4.4 Analýza požadavků, realizace případů užití

Před začátkem analýzy jednotlivých požadavků by bylo vhodné uvést nějaký komplexní pohled na strukturu aplikace např. ve formě diagramu tříd. Avšak s použitím frameworku a redakčního systému jsou moduly a relace mezi nimi generovány automaticky na základě pravidel definovaných v definici tabulek jednotlivých modulů. Samotný framework a redakční systém pak obsahují objekty, moduly a funkcionalitu, která je využívána nově definovanými moduly. Generovaná závislost tříd je pak pro grafické zobrazení poměrně složitá. Z těchto důvodů zde nebude uveden diagram tříd, který by znázorňoval všechny třídy vyvíjeného systému a vazby mezi nimi. Případně zájemce o celistvější pohled na strukturu aplikace odkážu na programovou dokumentaci vytvořenou programem *Doxygen*

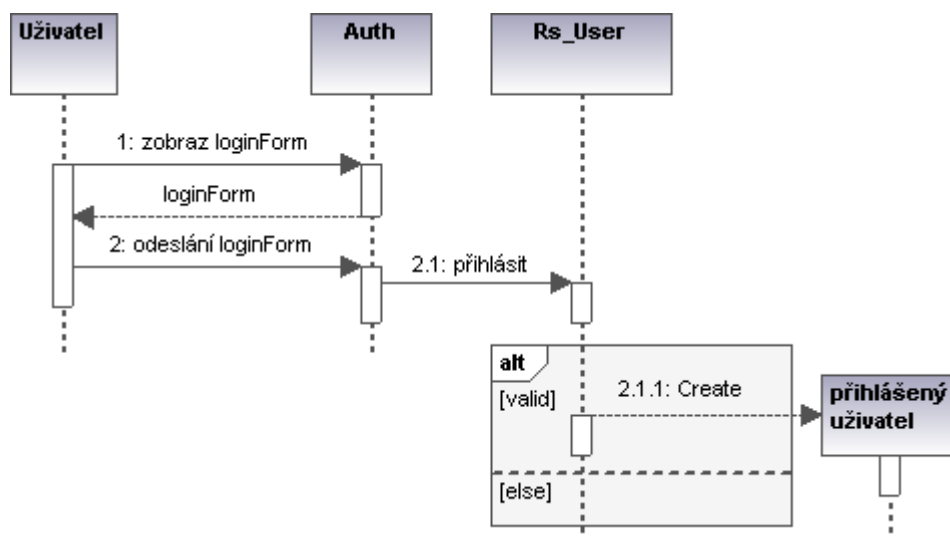
za pomoci generátoru grafů *Graphviz*, případně na vizualizaci struktury databáze vytvořené programem *SchemaSpy*.

Pro analýzu jednotlivých požadavků specifikovaných v podkapitole 3.2 zde bude uvedeno, jakým způsobem bude probíhat provádění scénářů požadavků v čase. Toto bude znázorněné pomocí sekvenčních diagramů.

4.4.1 Přihlašování uživatelů

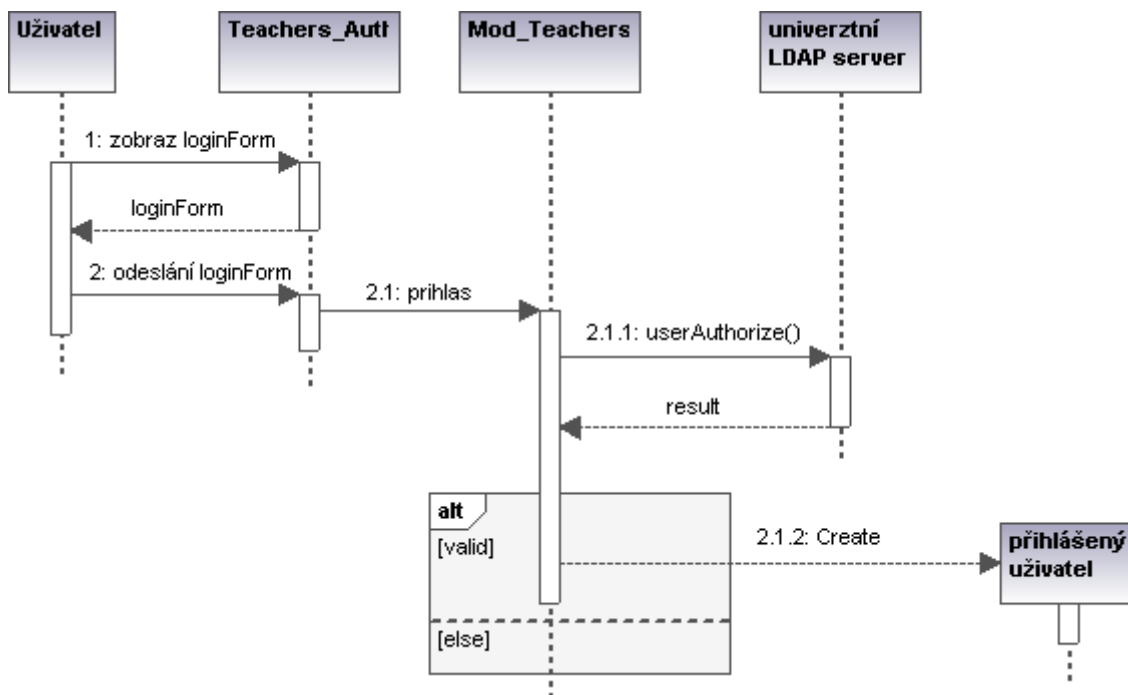
Přihlašování uživatelů bude dvojího druhu.

O **přihlašování běžných uživatelů** systému se bude starat třída *Auth*, která na základě zadaných přihlašovacích údajů uživatele provede přihlášení pomocí metod modulu pro správu uživatelů *Rs_User*. Modul *Rs_User* bude pak v systému reprezentován třídou stejného jména. Níže uvedený sekvenční diagram znázorňuje průběh přihlášení pro případ užití UC1.



Obr. 6. Přihlašování běžného uživatele

Přihlašování vyučujících bude obstarávat třída *Teachers_Auth* spolu s modulem pro správu vyučujících *Mod_Teachers*. Je zde analogie s přihlašováním běžných uživatelů. I tady bude modul *Mod_Teachers* s systému reprezentován třídou stejného jména. Navíc, aby byla zachována funkcionality pro přihlašování a správu uživatelů, bude tato třída potomkem třídy *Rs_User*. Ta pak bude ověřovat existenci uživatele na univerzitním LDAP serveru pomocí uživatelem zadaných přihlašovacích údajů. Realizace případu užití UC2 je pak znázorněna na obrázku *Obr. 7*.



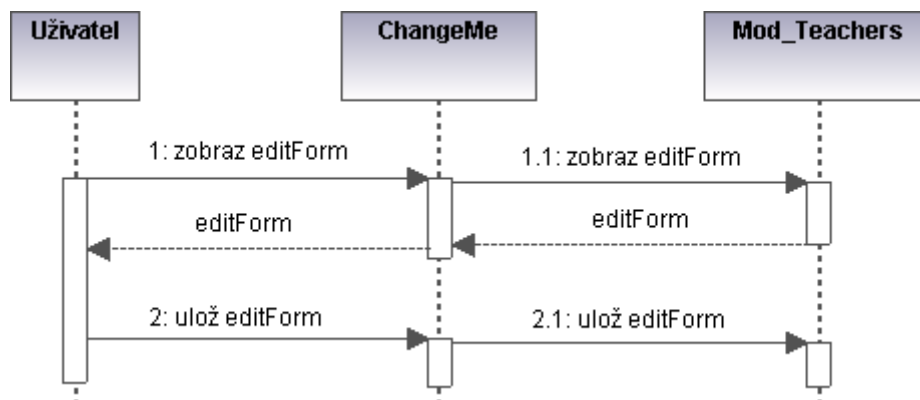
Obr. 7. Přihlašování vyučujícího

4.4.2 Editace osobních údajů

Editace osobních údajů je speciální případ pro zpracovávání záznamů modulu pro správu uživatelů a modulu pro správu vyučujících. Speciální v tom, že i v případě, že nemá daný uživatel oprávnění měnit záznamy v daných modulech, může mít oprávnění pro změnu svého vlastního záznamu.

Editaci osobních údajů bude zajišťovat univerzální modul *ChangeMe* (v systému reprezentovaný třídou *Rs_User_ChangeMe*). Modul bude univerzální v tom, že bude využívat vlastností předaného modulu (v našem případě modulu pro správu uživatelů nebo vyučujících). Bude tedy sloužit jako rozhraní pro užívání cizích modulů při editaci konkrétního záznamu modulu. Takový přístup bude mít za následek, že změna osobních údajů uživatele nebude vázána na konkrétní modul (*Rs_User*), nýbrž bude možné použít i jiného modulu (*Mod_Teachers*). Tím docílíme toho, že nebudeme muset v systému rozlišovat přístup uživatele na programové úrovni, ale až na úrovni konfigurace systému.

Pokud byl předchozí odstavec nejasný, situaci možná více zprůhlední sekvenční diagram pro případ užití UC3.



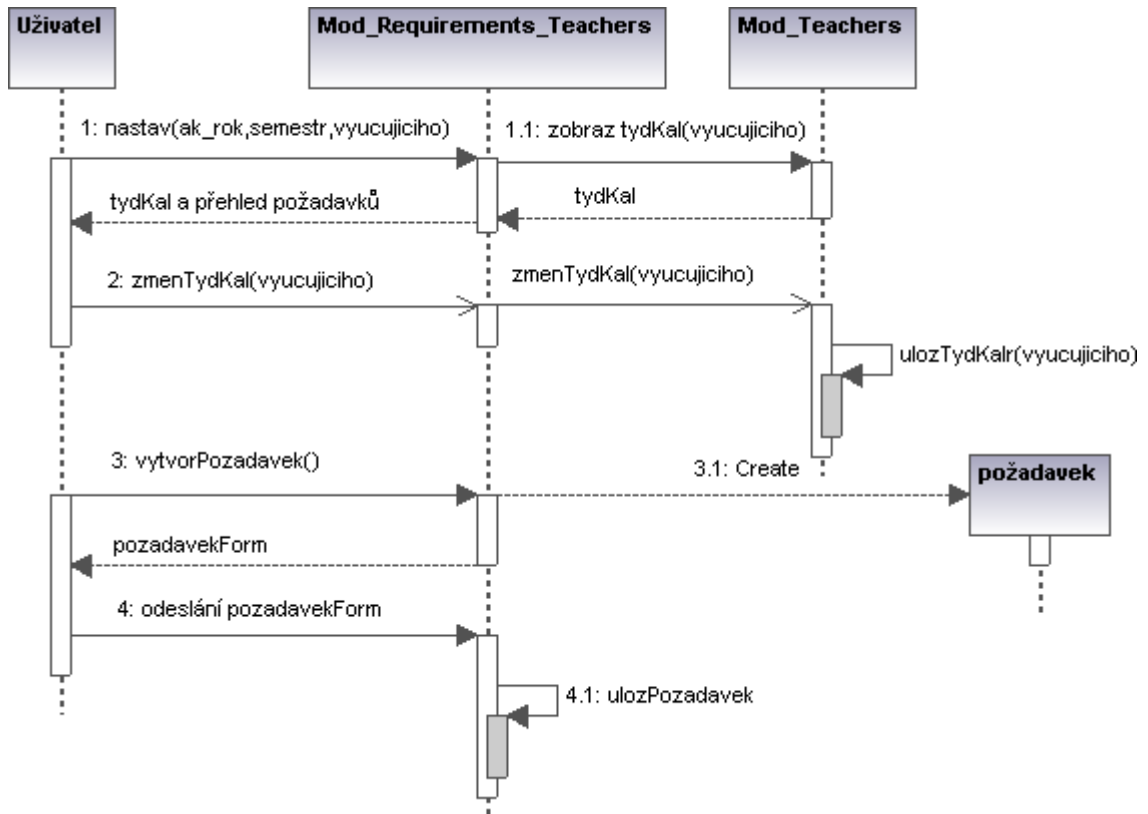
Obr. 8. Editace osobních údajů

4.4.3 Zadávání požadavků

Dle specifikace redakčního systému bude právě modul tvořit rozhraní pro zpracovávání vlastních záznamů. Zpracováváním se rozumí jakékoliv akce prováděné nad záznamy.

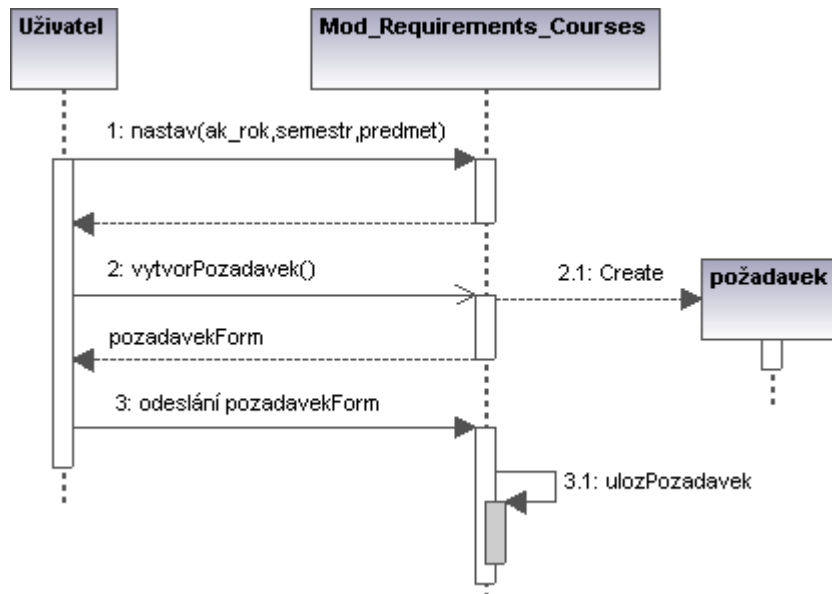
Jak je uvedeno v podkapitole 3.2.3 je zadávání požadavků dvojího druhu. Proto vzniknou dva moduly (tedy dvě třídy) *Mod_Requirements_Teachers* a *Mod_Requirements_Courses*, které budou poskytovat prostředky pro správu požadavků.

Obrázek *Obr. 9.* reprezentuje sekvenční diagram případu užití UC4 pro zadávání požadavků vyučujících.



Obr. 9. Zadávání požadavků vyučujících

Pro případ užití UC5 pak sekvenční diagram vypadá následovně.

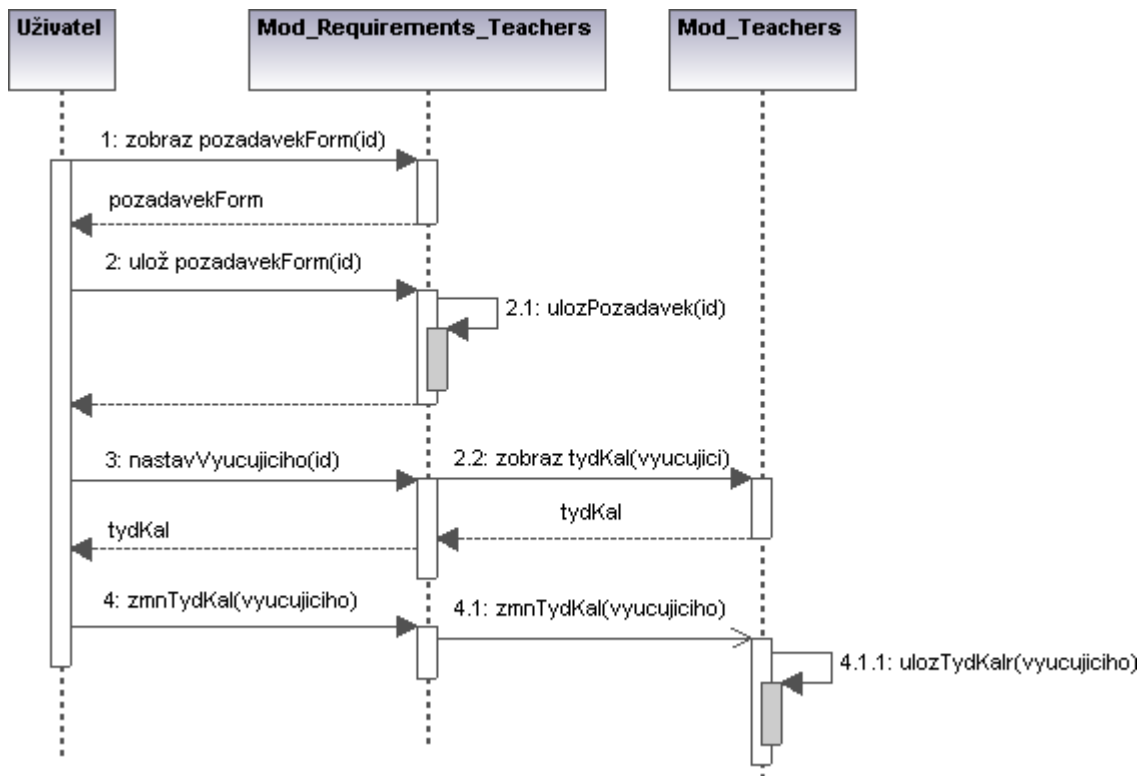


Obr. 10. Zadávání požadavků předmětů

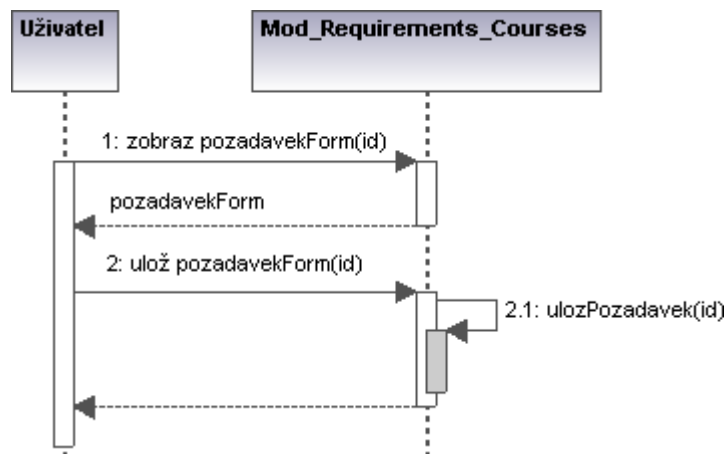
4.4.4 Editace požadavků

Stejně jako u zadávání požadavků je i editace dvojího druhu, editace požadavků vyučujících a požadavků předmětů. A stejně tak i funkce pro správu požadavků zajišťují moduly *Mod_Requirements_Teachers* a *Mod_Requirements_Courses*.

Realizace případu užití UC6 je na *Obr. 11*, pro případ užití UC7 pak na obrázku .



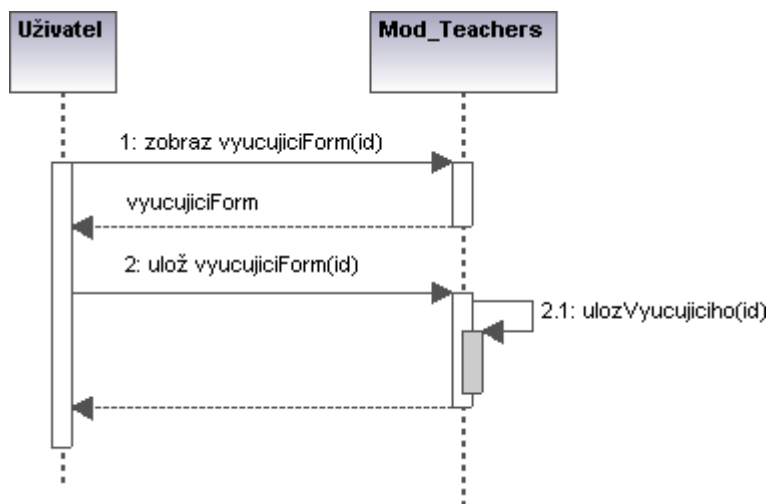
Obr. 11. Editace požadavků vyučujících



Obr. 12. Editace požadavků předmětů

4.4.5 Editace vyučujících

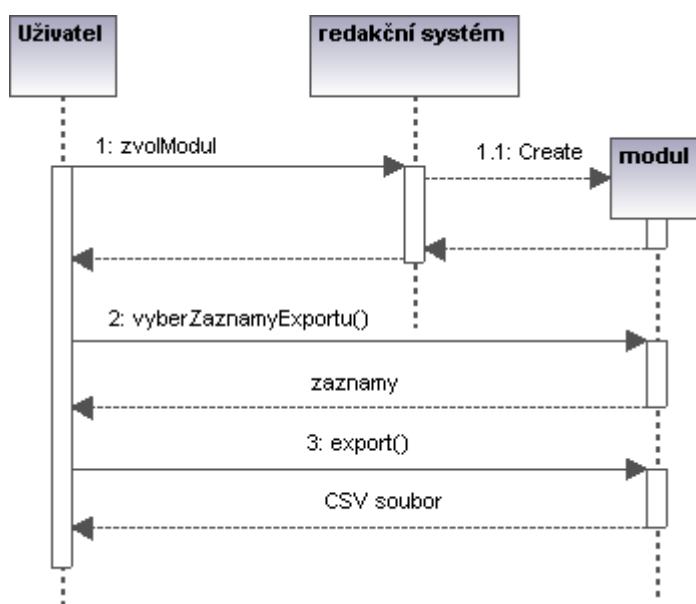
Operace se záznamy vyučujících budou prováděny pomocí modulu *Mod_Teachers*. Editace záznamu vyučujícího pro případ užití UC8 bude mít průběh dle *Obr. 13*.



Obr. 13. Editace vyučujících

4.4.6 Import/export dat

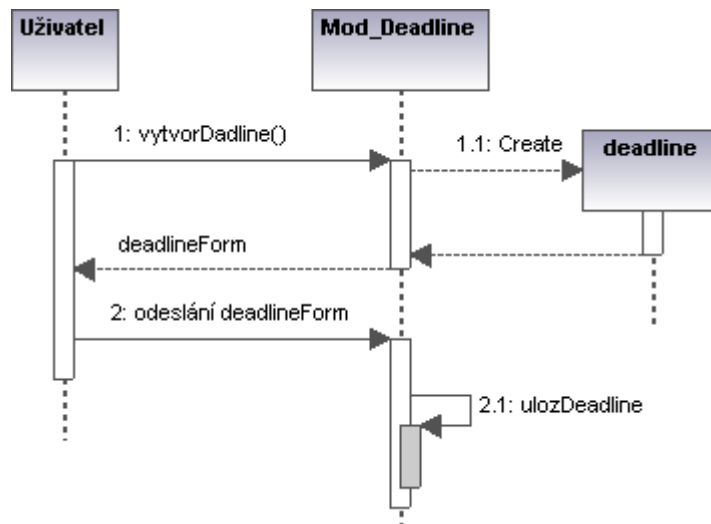
Export dat do CSV nebude zajišťovat konkrétní modul, bude to obecná vlastnost všech modulů. Export dat nějakého modulu bude tedy možný, pokud bude mít takový modul povolen a nastaven export ve své konfiguraci. Na obrázku *Obr. 14*. je pak vidět průběh exportu dat.



Obr. 14. Export záznamů modulu do CSV

4.4.7 Zadávání deadlinů

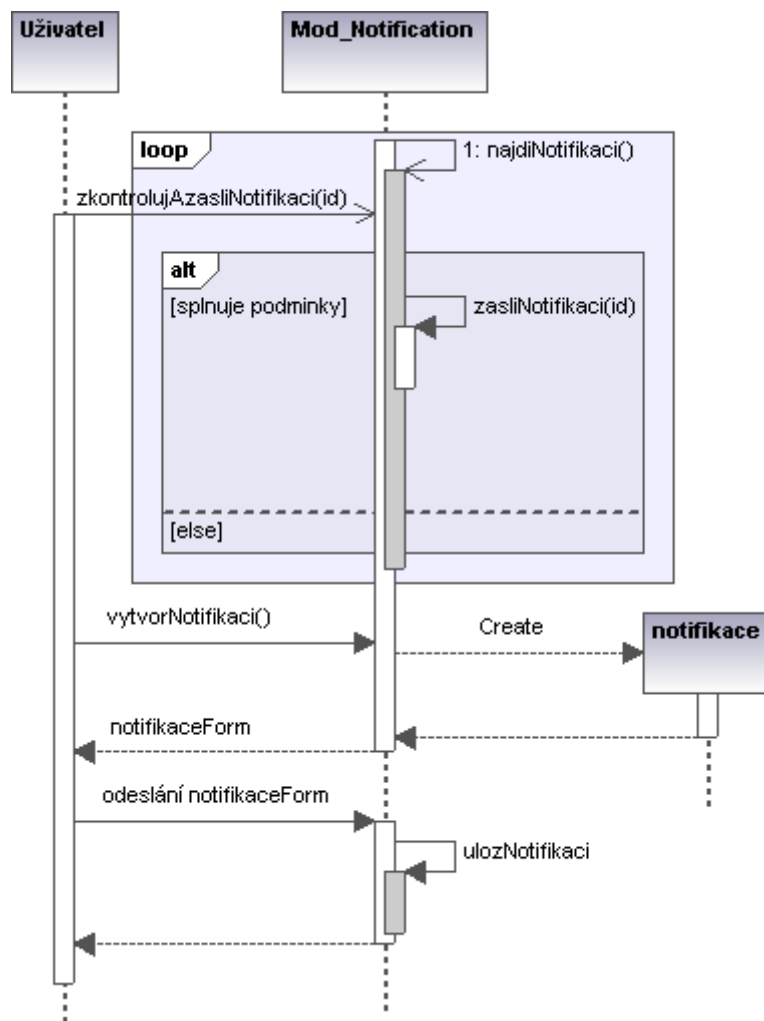
Pro správu deadlinů pro konečné zadávání požadavků bude vytvořen modul *Mod_Deadline*. Realizace případu užití UC10 pak vypadá následovně.



Obr. 15. Zadávání deadlinů

4.4.8 Zaslání e-mailových notifikací

Pro emailové notifikace vznikne nový modul *Mod_Notification*. Ten bude, vedle standardních operací pro vytvoření nové notifikace, zadání předmětu notifikace, volbu deadlinu a dalších požadavků uvedených v podkapitole 3.2.8, implementovat také automatizované zaslání notifikací a možnost ručního zaslání notifikace přímo uživatelem. Automatizace zaslání notifikací bude provedena pomocí funkce modulu, která bude pravidelně spouštěna jako plánovaná úloha programem Cron.



Obr. 16. Zasilání e-mailových notifikací

4.4.9 Správa systémových modulů

Pro správu systémových modulů zde nebudu uvádět žádné sekvenční diagramy, pouze textový popis. Hlavním důvodem je, že v podkapitole 3.2.9 nejsou uvedeny žádné případy užití, další důvod je ten, že se jedná o popisy interakcí systému s uživatelem, které již byly modelovány v předchozích podkapitolách a jsou analogické. Jedná se zejména o obecné chování systému při vytváření nových záznamů modulů a editaci záznamů.

Jak bylo uvedeno v podkapitole se specifikací požadavků správy systémových modulů, správa se skládá ze čtyř hlavních částí:

- *Správa uživatelů*
- *Správa uživatelských skupin*
- *Instalace a úprava modulů*

- ***Nastavení přístupových oprávnění***

Pro analýzu tohoto požadavku by bylo vhodné konkretizovat pojem „*správa*“ na „*administrace*“. K těmto částem systému bude mít totiž přístup pouze člen skupiny „*Admin*“, tedy administrátor.

Administrace uživatelů bude realizována modulem *Rs_User* reprezentovaného v redakčním systému stejnojmennou třídou. Ta bude poskytovat jak základní operace vytvoření, aktivace/deaktivace a smazání uživatelského účtu, tak i operace pro změnu detailů daného uživatelského účtu (např. změnu přihlašovacích údajů nebo e-mailu atd.).

Pro ***administraci uživatelských skupin*** bude vytvořen modul *Mod_User_Groups*. Ten bude funkčně stejný jako modul *Rs_User* pro administraci uživatelských účtů, rozdíl bude pouze ve struktuře zpracovávaných dat (uživatelské skupiny nebudou obsahovat např. login a heslo jako uživatelské účty).

U ***administrace instalace a úprav modulů*** se jedná o speciální případ modulu. Ten totiž neoperuje nad prostými daty jako většina modulů, ale obstarává instalaci, reinstalaci a systémová nastavení pro ostatní moduly. To bude realizováno modulem (třídou) *Rs_Sections*, který bude zajišťovat správnost instalace jednotlivých modulů a jejich systémových nastavení. Tento modul bude mít ještě vedlejší úkol, budou jím vytvářeny položky menu redakčního systému, které se budou zobrazovat přihlášeným uživatelům. Pro možnosti lepšího nastavení tohoto menu bude modul umožňovat zadávat jednotlivé položky jako:

- *skryté* – položka se nebude v menu vypisovat
- *systémové* – položka bude vypisována pouze v menu administrace systému
- *položky menu* – bude sloužit pouze jako položka menu, nikoliv jako modul

Administrace přístupových oprávnění, jak bylo řečeno ve specifikacích požadavků, je jedna z nejdůležitějších částí systému zajišťující bezpečný chod celé aplikace. Je tedy nezbytně nutné provést důkladnou analýzu a při následném návrhu pečlivě postupovat, aby bylo riziko chyb co nejmenší.

Rozhodl jsem se, že tato část systému bude fungovat na principu firewallu. ***Firewall*** je hardwarové nebo softwarové zařízení, které dokáže filtrovat komunikaci mezi dvěma a více body v síti (zpravidla mezi počítačem a internetem). Obecně jsou dva typy firewallů,

prvním je proxy server, a druhý pracuje na základě paketového filtru a právě tato varianta bude zvolena jako jádro pro zjišťování přístupových oprávnění. Paketový filtr obsahuje sadu pravidel. Na příchozí paket jsou pak tato pravidla aplikována, a pokud jim paket vyhovuje, je propuštěn přes firewall, jinak je zahozen. V případě přístupových oprávnění bude fungování obdobné. Bude vytvořen modul *Rs_Acl*, který bude sloužit pro definování sad přístupových pravidel. Dále budou vytvořeny podpůrné moduly *Rs_Action* pro definování prováděných akcí jako editace nebo mazání a *Rs_Condition* pro definování podmínek pro nastavení platnosti daných akcí, např. že akce „přidávání“ záznamů bude platná vždy, nikdy, pouze pro vlastní záznamy atd. Pak bude modul přístupových pravidel využívat systémové tabulky *rs_langs*, ve které budou definovány jazykové mutace platné v kontextu redakčního systému, bude tak možné např. zakázat editaci záznamů v dané jazykové mutaci. Pak bude modul *Rs_Acl* závislý na modulu správy sekcí *Rs_Sections*, tak bude možné přidělit přístupové pravidlo konkrétní sekci. Nakonec, jak už bylo uvedeno ve specifikaci požadavků, zde bude závislost na konkrétní uživatelské skupině.

Další podrobnější informace o řešení přístupových oprávnění jsou vysvětleny v kapitole 5.

5 NÁVRH SYSTÉMU

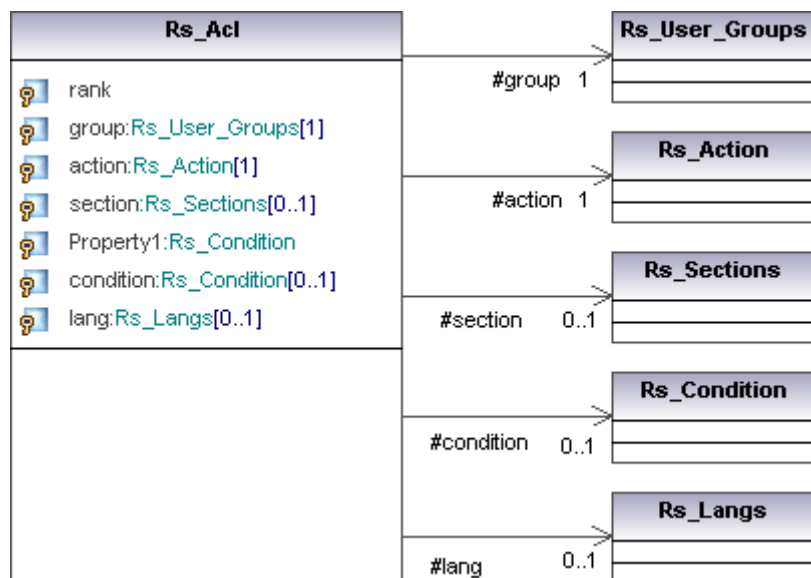
V předchozích kapitolách byla provedena specifikace a analýza požadavků portálu pro zadávání požadavku na rozvrh. Cílem této kapitoly bude provedení návrhu realizace aplikace na základě analýzy z předchozí kapitoly.

Budou zde uvedeny a rozebrány pouze důležité části implementovaného systému. Podrobný návrh realizace všech částí systému by přesáhl rozsah tohoto dokumentu. Zájemce o podrobnější informace odkazují tímto na programovou dokumentaci.

5.1 Přístupová oprávnění

Dle analýzy v podkapitole 4.4.9 v části „Administrace přístupových oprávnění“ byla provedena realizace mechanismu firewallu.

Každé přístupové pravidlo obsahuje povinné a volitelné položky. Povinné položky jsou uživatelská skupina a akce, ke kterým se pravidlo vztahuje, nepovinné pak všechny ostatní. Pokud není některá z nepovinných položek zadána, je tato brána, že pravidlo platí pro všechny její možné hodnoty. Níže uvedený diagram tříd znázorňuje situaci.



Obr. 17. Diagram tříd přístupových oprávnění

Pro zachování přehlednosti nejsou v diagramu záměrně atributy a metody referovaných tříd ani metody třídy *Rs_Acl*. Dále je tu drobná nepřesnost, je zde uvedena třída *Rs_Langs*, která však v systému fyzicky neexistuje. Tato třída v diagramu pouze reprezentuje tabulku s povolenými jazykovými mutacemi.

Ve třídě *Rs_Acl* je navíc atribut *rank*, ten určuje pořadí přístupového pravidla při vyhodnocování.

5.1.1 Vyhodnocování přístupových pravidel

Před začátkem vyhodnocování přístupu je předpoklad, že *uživatel není oprávněn*. Při vyhodnocování je nutné předem znát podmínky pro vyhodnocování, tj. zadání sekci, zobrazenou jazykovou mutaci webu, prováděnou akci, příp. záznam, se kterým má být akce provedena a vlastníka tohoto záznamu. Poté jsou vybrána všechna přístupová pravidla pro daný vstup a procházena od posledního k prvnímu. První, který bude vyhovovat prováděné akci, je pak brán jako požadovaný výsledek. Z tohoto pravidla získáme podmínku přístupu a aplikujeme ji v závislosti na tom, jestli je přihlášený uživatel vlastník záznamu, se kterým chce manipulovat anebo patří alespoň do stejné uživatelské skupiny. Z podmínky pak získáme přímý výsledek TRUE nebo FALSE podle toho, jestli má uživatel oprávnění anebo ne.

Přístupová práva - test

Sekce	Podmínka	Akce	Jazyk	Úpravy
*	skupinové	*	*	  
*	nikdy	Mazání	*	  
Formy studia	vždy	Mazání	Česky	  

Obr. 18. Příklad nastavení přístupových oprávnění

Na obrázku *Obr. 18*. Je uveden příklad nastavení přístupových oprávnění pro uživatelskou skupinu „test“. Pravidla jsou čtena od shora dolů, tj. dle jejich pořadí (pozor, vyhodnocování pravidel je prováděno v opačném směru, tedy od spodu nahoru). První pravidlo způsobí, že uživatel skupiny „test“ získá oprávnění provádět libovolné akce v libovolných jazykových mutacích ve všech sekcích redakčního systému pro všechny záznamy, jejichž majitelé jsou ve skupině „test“. Druhé pravidlo vak uživateli zakazuje provádět akci mazání záznamů pro všechny sekce všech jazykových mutací. Třetí pravidlo povoluje akci mazání v sekci „Formy studia“, pokud mazaný záznam bude v české mutaci.

5.2 Číselníky

Pro správnou funkci zadávání požadavků bylo nutné vytvořit množství číselníků. Číselníky jsou jakési soubory dat, které lze pomocí relací přiřadit k nějakému záznamu. Tyto číselníky byly vytvořeny z části při vstupní analýze, některé v důsledku vývoje jako rozšíření, zobecnění funkcionality nebo na základě vstupních omezení. Některé číselníky byly pak přebrány z IS Stag, pro vyšší kompatibilitu s tímto systémem.

Názvy číselníků jsou voleny tak, aby co nejlépe vystihovaly data, která obsahují. Pro jednotlivé číselníky bude uvedena jejich stručná charakteristika. Číselníky ve skutečnosti obsahují větší spektrum dat, než je uvedeno v jejich popisech, zpravidla jsou to názvy položek, zkratky a další parametry. Ty ale nejsou důležité pro pochopení významu číselníků, proto zde nebudou uvedeny a případné zájemce o technické detaily odkážu na programovou dokumentaci.

5.2.1 Akademický rok

Jde o jednoduchou kolekci záznamů představujících akademické roky.

5.2.2 Semestry

Opět jednoduchá kolekce, obsahující soubor semestrů.

5.2.3 Členění

Číselník členění představuje stromovou strukturu. Jednotlivé záznamy představují součásti univerzity. V kořenových uzlech jsou záznamy reprezentující fakulty (FT, FAI, ..), v listech pak konkrétní ústavy (UAI, UEM, ...).

5.2.4 Formy studia

Kolekce forem studia (prezenční, kombinované, ...).

5.2.5 Ročníky

Jedná se o kolekci ročníků (I. bakalářský, II. Bakalářský, I. Magisterský, atd.). Tento číselník je již trochu složitější a to v tom, že obsahuje relaci na číselník typů studia (5.2.12).

5.2.6 Studijní obory

Představuje soubor studijních oborů, například můj obor je V/IT neboli pátý ročník oboru Informační technologie.

5.2.7 Stavý časového plánu

Pomocný číselník pro nastavení možných stavů časového plánu. Časovým plánem se zde rozumí týdenní kalendář, který vyplňuje vyučující. Stavý se pak přiřazují jednotlivým hodinám v týdnu. Každý stav má svou informační hodnotu, např. že v tu danou hodinu vyučující nemůže vyučovat apod.

5.2.8 Učebny

Kolekce záznamů, které představují učebny pro výuku předmětů. Každá učebna má svůj typ (laboratoř, posluchárna, kancelář, atd., relace na Typy učeben) a existuje v nějaké budově (relace na Budovy).

5.2.9 Budovy

Jedná se o soubor budov, kterými univerzita disponuje. Každá budova tohoto číselníku se nachází na nějakém konkrétním místě (relace na Místa).

5.2.10 Místa

Číselník míst, možná lépe řečeno adres, ve kterých jsou lokalizovány univerzitní budovy.

5.2.11 Typy učeben

Kolekce typů učeben (Jazyková učebna, Kreslírna, ...).

5.2.12 Typy studia

Kolekce typů studia (bakalářské, magisterské, ...).

5.2.13 Detašovaná pracoviště

Podobně jako místa (5.2.10) definují lokaci. V tomto případě však ne lokaci konkrétních budov, ale detašovaných pracovišť.

5.2.14 Typy zaměstnance

Číselník pro lepší specifikaci pozice zaměstnance (interní, externí, ...).

5.2.15 Zařazení

Další číselník pro bližší specifikaci konkrétního zaměstnance. Označuje např. zda-li daný zaměstnanec učí studenty kombinovaného studia, nebo jestli dojíždí apod.

5.3 Moduly požadavků

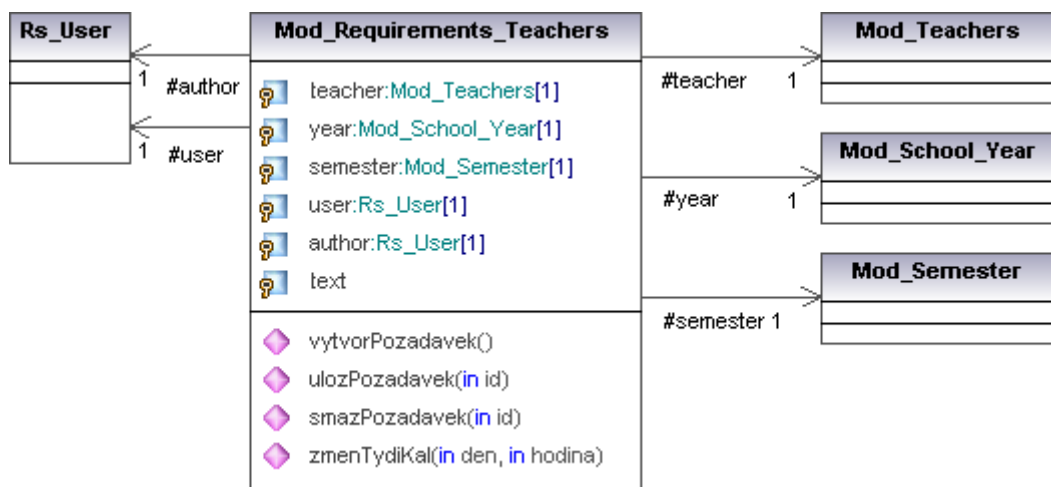
Zde je uveden návrh struktur pro zpracovávání požadavků vyučujících a požadavků předmětů. Pro zachování jednoduchosti a přehlednosti zde nebudou uvedeny některé implementační detaily, jako jsou např. atributy referovaných tříd a zděděné vlastnosti a atributy v diagramech tříd.

5.3.1 Požadavky vyučujících

Modul (třída) *Mod_Reuitements_Teachers* zajišťuje obsluhu požadavků vyučujících na rozvrh. Požadavek vyučujícího na rozvrh lze rozdělit na dvě části a to:

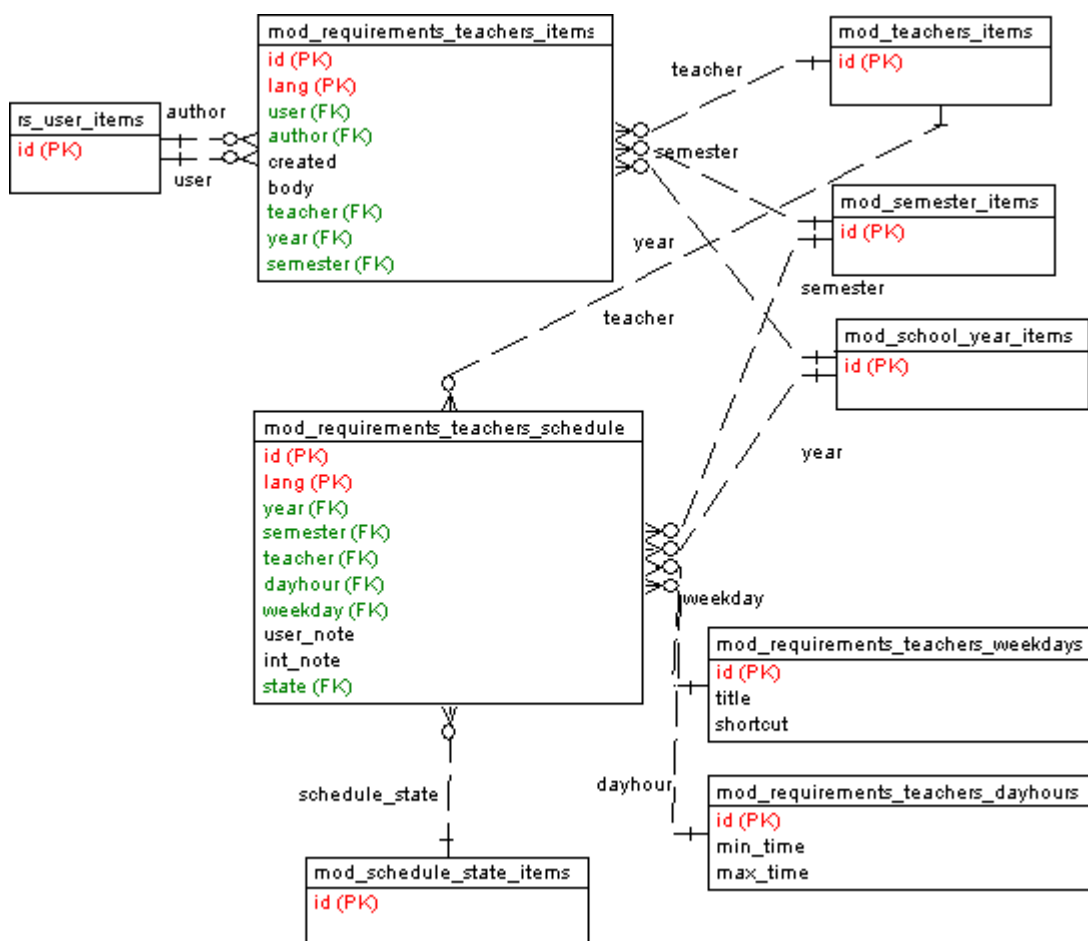
- *týdenní kalendář vyučujícího* a
- *obecný textový požadavek* sloužící jako doplnění týdenního kalendáře

Na obrázku *Obr. 19.* je znázorněn modul *Mod_Reuitements_Teachers* v diagramu tříd s jeho nejbližším okolím.



Obr. 19. Diagram tříd pro požadavky vyučujících

Jak je vidět, samotný diagram tříd pro popis modulu úplně nestačí. Je zde sice patrné, že modul poskytuje metody pro tvorbu, aktualizaci a odstranění textového požadavku a také aktualizaci týdenního kalendář vyučujícího, ale již není vidět pozadí části modulu pro uchovávání týdenního kalendáře. Je to proto, že struktury pro uchovávání těchto dat jsou součástí třídy samotné. Níže uvedený obrázek znázorňuje přímo databázové vazby mezi jednotlivými tabulkami a již je dosti výstižný pro ucelení pohledu na tento modul i přesto, že byly opět pro přehlednost vynechány mnohé systémové sloupce tabulek.



Obr. 20. Schéma databázových tabulek modulu *Mod_Requirements_Teachers*

Oproti digramu tříd zde figurují další čtyři tabulky. První je tabulka *mod_schedule_state_items* modulu *Mod_Schedule_State*. Jde o tabulku s číselníkem stavů časového plánu popsaného v podkapitole 5.2.7. Další dvě tabulky *mod_requirements_teachers_weekdays* a *mod_requirements_teachers_dayhours* jsou vnitřní tabulky modulu *Mod_Requirements_Teachers*. První obsahuje pevně daný číselník dnů v týdnu a druhá také pevně daný číselník hodin dne. Poslední tabulka

mod_requirements_teachers_schedule slouží právě k uchování informací o týdenním kalendáři vyučujícího.

5.3.2 Požadavky předmětů

Modul pro zadávání a správu požadavků předmětů se jmenuje *Mod_Requirements_Courses*. Pro každý předmět lze obecně zadat více požadavků. Každý požadavek se však vztahuje k jednomu konkrétnímu předmětu. Pro zadání požadavku předmětu je nejprve nutné zvolit předmět, ke kterému se bude požadavek vztahovat. Jelikož je každý předmět vázán na akademický rok a semestr, kdy je vyučován, a fakultu/ústav, kde je vyučován, je třeba při výběru předmětu nejdříve zvolit fakultu/ústav, akademický rok a semestr pro usnadnění hledání předmětu. Předmětů pro daný rok je totiž poměrně hodně. Při zadávání požadavku pak uživatel kromě textové poznámky může specifikovat:

- učebnu, kterou by si přál pro výuku
- typ místnosti – seminární, s diaprojektorem, atd.
- místo výuky – město
- druh předmětu – povinně volitelný, nepovinně volitelný, atd.
- seznam alternativních učeben, které jsou pro výuku vhodné
- typ studia, pro které bude předmět vyučován (prezenční, kombinované)
- počet studentů
- rozsah přednášek, laboratoří a cvičení, pokud není zadán
- vyučujícího/vyučující předmětu
- předmět, se kterým má výuka probíhat souběžně, příp. studijní obor/obory, pro než má výuka probíhat současně

6 BEZPEČNOSTNÍ RIZIKA A JEJICH OŠETŘENÍ

6.1 Zjišťování hesel hrubou silou

Jedním z nejznámějších způsobů nabourávání se do systému je zjišťování přihlašovacích údajů hrubou silou. Jde o způsob, kdy útočník zkouší náhodně případně s použitím nějaké heuristiky hesla tak dlouho, dokud jej systém nepřihlásí.

Zabezpečení

Jako zabezpečení proti tomuto útoku jsem implementoval do instalace databáze redakčního systému databázový zámek. Ten znemožní přihlašování po dobu deseti minut, pokud došlo k deseti po sobě neúspěšných pokusech přihlášení z dané IP adresy.

6.2 Odposlech informací

Odposlech je jednou z možností, jak mohou nepovolené osoby získat citlivé informace, které jsou přenášeny mezi webovou aplikací a klientem. Komunikace probíhá zpravidla přenosem formulářových dat. V případě redakčního systému jsou nejcitlivější data přihlašovací jméno a heslo uživatele, kritickým místem pak přihlašování do systému nebo změna přihlašovacích údajů.

Formulářová data jsou přenášena dvěma způsoby, buďto metodou *GET* anebo *POST*. Při přenosu metodou *GET* je velice snadné odposlechnout veškerá formulářová data, ta jsou totiž přenášena přímo v URL adrese. Útočníkovi tedy stačí jen z hlavičky požadavku získat žádanou URL adresu. Tato metoda proto není vhodná pro přenos citlivých údajů. Druhou variantou je metoda *POST*. Zde jsou formulářová data přenášena v těle požadavku a tak nejsou přímo vidět v URL adrese požadavku. Metoda *POST* byla proto zvolena jako jedno z bezpečnostních opatření pro přenos citlivých dat.

Samotné posílání formuláře pomocí *POSTu* však není dostatečně silné opatření. „Šikovný“ útočník totiž může odposlouchávat přímo zasílané pakety komunikace a z nich získat potřebná data. Komunikační protokol HTTP totiž zasílá data v čitelné podobě.

Zabezpečení

První druh zabezpečení je použití *POST* metody při zasílání citlivých údajů. Pro další zabezpečení jsem pak použil vynucenou komunikaci pomocí protokolu HTTPS, který veškerou komunikaci mezi serverem a klientem šifruje.

6.3 SQL injection

Cílem útoku označovaným pojmem SQL injection je snaha o pozměnění výsledků SQL dotazů podvržením vstupních dat. Je využíváno toho, že většina SQL dotazů je generována dynamicky za běhu aplikace s použitím zadaných parametrů. Jedná se vlastně o „vsunutí“ cizího nežádoucího SQL kódu do generovaného dotazu.

Tento typ útoku je zpravidla realizován pomocí formuláře, jehož data jsou zpracovávána a ukládána do databáze s nedostatečně ošetřenými vstupy. Útočník pak získává prakticky neomezenou kontrolu na databázi.

Příklad

Mějme SQL dotaz „*SELECT nazev FROM tab_barvy WHERE poradi = \$poradi;*“, kde proměnná *\$poradi* obsahuje hodnotu zadanou uživatelem. Pokud by vstup od uživatele byl ve skutečnosti řetězec „*1; DROP TABLE tab_barvy;*“ a vložil by se takto neošetřen do generovaného SQL dotazu, byla by po vykonání tohoto dotazu odstraněna tabulka *tab_barvy* z databáze.

Zabezpečení

Zabezpečení systému proti tomuto typu útoku je dvojího druhu.

Prvním je **používání automatické validace dat** vkládaných do databáze pomocí instancí tříd datových typů. Každá třída datového typu obsahuje metodu *dbFormat(\$val)*, která zkontroluje platnost a formu dat pro daný datový typ a vrátí vstupní hodnotu *\$val* naformátovanou pro bezpečné vložení do databáze. Například třída datového typu pro integer *Type_Int* při volání zmíněné metody povolí vložení hodnot *5*, *8976*, *null*, *'NULL'*, ale pro vstup *'a'* je vyvolána výjimka *TypeMismatch*. Pomocí validace datovými typy je v systému ošetřena většina vstupních dat.

Druhá možnost zabezpečení je používána všude tam, kde není použito nebo nelze použít automatickou validaci. Jde hlavně o ruční skládání SQL dotazů. Toto zabezpečení spočívá v převedení každé hodnoty, která má být součástí SQL dotazu, na řetězec, transformací pomocí vestavěné PHP funkce *pg_escape_string()* a obalení jednoduchými uvozovkami. To způsobí, že se data v dotazu chovají jako jeden celek a jsou pak také brána jako jedna hodnota.

6.4 Cross-site scripting (XSS)

Jedná se o jeden z nejznámějších způsobů útoku. Jedná se opět o vložení nežádoucího kódu jako u SQL injection, v tomto případě ale přímo do zobrazovaného hypertextového dokumentu.

Vkládaný kód je většinou kód JavaScriptový. Možnosti vkládání nežádoucího kódu jsou tři:

- **lokální**, označovaný též **DOM based** – jedná se o neošetřené přenesení proměnné z URL adresy do JavaScriptu
- **non-persistent** neboli též **reflected** – podobně jako v prvním případě jde o neošetřený vstup parametru z URL adresy pouze s tím rozdílem, že zpracování je prováděno na straně serveru a nežádoucí kód zabudován přímo do struktury generovaného dokumentu
- **persistent** nebo též **stored** či **sekond-order** – jde o neošetřený vstup z formulářů, který je uložen přímo do databáze a při zobrazování je pak do stránky vložen tento kód, který je považován za bezpečný

Zabezpečení

První dva typy nelze na systém aplikovat, jelikož v aplikaci nejsou vstupy z URL adresy zpracovávány přímo do stránky. Třetí typ je ošetřen úpravou hodnot vkládaných do databáze pomocí datových typů zpracovávající řetězce. Proces je zajištěn pomocí PHP funkce *strip_tags()*, která odstraní z vkládaných řetězců všechny HTML tagy. Výjimku tvoří datový typ *Type_Wysiwyg*, který slouží zpracovávání dat z WYSIWYG editorů, který je ve formě HTML kódu, nelze proto odstraňovat všechny HTML tagy. Tento datový typ odstraní pouze tagy `<script>` a `</script>` pomocí metody *stripScript()*:

```
public function stripScript($val)
{
    return ereg_replace('<(\s*\|\/)?\s*script\s*[\^>]*>', '', $val);
}
```

6.5 Session hijacking

Session je soubor informací, které jsou zpravidla unikátní v rámci jedné komunikace mezi klientem (uživatel) a serverem. Každá session pak obsahuje identifikátor, tzv. `session_id`, kterým se při každém dalším požadavku klient identifikuje serveru. Jde o náhodně vygenerovanou obtížně uhodnutelnou posloupnost alfanumerických znaků. Způsoby přenosu `session_id` mezi klientem a serverem jsou tři: pomocí cookies, URL adresou nebo pomocí skrytého formulářového pole.

Session hijacking označuje útok, který je proveden tak, že útočník ukradne `session_id` nějakého již přihlášeného uživatele a tím se za něj může bez jakýchkoliv problémů vydávat.

Zabezpečení

Proti tomuto útoku je potřeba se bránit jak na straně serveru, tak na straně klienta. U klienta je ochrana poměrně složitá a nelze ji ze strany severu a systému nijak ovlivnit. Jako zabezpečení systému jsem použil předávání `session_id` pomocí cookies a šifrování komunikace pomocí protokolu HTTPS.

6.6 Session fixation

Jde opět o útok pomocí `session_id`. Útočník podstrčí uživateli vlastní `session_id` a když se pak uživatel přihlásí, použije jej pro sebe.

Zabezpečení

Proti tomuto typu útoku se lze v PHP lehce bránit a to pomocí funkce `session_regenerate_id()`, která změní `session_id` se zachováním dat, která jsou v session doposud uložena. Funkci jsem v systému použil pro kritické místo po přihlášení uživatele.

ZÁVĚR

V této diplomové práci jsem řešil návrh a realizaci www portálu pro zadávání požadavků na rozvrh. Jakou součástí systému jsem pak implementoval framework a redakční systém, ve kterém byl celý systém vyvinut.

V rámci zadání jsem se seznámil s obecnou problematikou tvorby rozvrhů na školách a konkrétními problémy při sestavování rozvrhu na vysokých školách. Součástí zadání bylo také obeznámení se s prací v programu Rozvrh, který slouží pro automatické generování rozvrhů podle předem stanovených pravidel.

V teoretické části jsem uvedl charakteristiku programu Rozvrh a nastínil hlavní problémy při tvorbě rozvrhů na vysokých školách. V této části jsem také provedl hrubou analýzu a specifikoval požadavky, které by měl vyvíjený systém splňovat. Pro popis jednotlivých specifikací jsem použil metodiku vytváření případů užití a vytváření scénářů, které jsou součástí jazyka UML.

Praktická část byla zaměřena na hledání nástrojů a technologií pro vývoj systému a odůvodnění jejich použití. Poté jsem provedl podrobnou analýzu požadavků specifikovaných v teoretické části a analýzu vytvářeného frameworku a redakčního systému. Analýza proběhla za pomoci sekvenčních diagramů, které znázorňují případy užití v čase. V kapitole Návrh systému jsem uvedl bližší informace o modulech pro správu požadavků a řešení některých kritických částí systému. Nakonec jsem popsal obecná bezpečnostní rizika při provozu informačních systémů spolu s nejčastějšími druhy útoků a uvedl způsoby zabezpečení implementované v systému.

V současné době je navržený systém v provozu na serveru společnosti Web4ce.

ZÁVĚR V ANGLIČTINĚ

In this diploma thesis, I solved the design and realization of web portal for entering requirements for the schedule. As part of the system I have implemented the framework and content management system, in which the whole system is developed.

Under award I was acquainted with the general problem of the creation of schedules in schools and specific problems in composition the schedule at universities. Part of the award was also familiar with the work in the program Rozvrh, which is used for automatic generation of schedules according to predetermined rules.

In the theoretical part, I introduced the characteristics of the program Rozvrh and outlined the main problems in creating the schedules at universities. In this section, I also realized an raw analysis and specified requirements, which should satisfy the developed system. For the description of the specifications I used the methodology of creation use cases and scenarios, which are part of UML language.

Practical part was focused on search tools and technologies for the development of the system and rationalization for their use. Then I made a detailed analysis of the requirements specified in the theoretical part and analysis developed framework and content management system. The analysis was made with the help of sequential diagrams illustrating the use cases in time. In chapter System design I introduced more information about the modules for the administration of requirements and solution certain critical parts of the system. Finally, I described the general security risks in the running of information systems, together with the most common types of attacks and said security method implemented in the system.

Currently, the designed system is running at the site of Web4ce Company.

SEZNAM POUŽITÉ LITERATURY

- [1] ČERVENÝ, Ľ. *Rozvrh* [počítačový program]. Ver. 4.1. Žilina, 2003, 15.9.2008 [cit. 2009-05-20]. Dostupný z WWW: <<http://www.cervený.sk/>>.
- [2] *Www.root.cz* [online]. 1998-2009 [cit. 2009-05-25]. Dostupný z WWW: <<http://www.root.cz/>>.
- [3] *PHP* [online]. 2001-2009 [cit. 2009-05-25]. Dostupný z WWW: <<http://php.net/>>.
- [4] *PostgreSQL* [online]. 1996-2009 [cit. 2009-05-25]. Dostupný z WWW: <<http://www.postgresql.org/>>.
- [5] *Sourcefouge.net* [online]. [cit. 2009-05-25]. Dostupný z WWW: <<http://schmaspy.sourcefouge.net/>>.
- [6] *Doxygen* [online]. 1997-2007 [cit. 2009-05-25]. Dostupný z WWW: <<http://www.stack.nl/~dimitri/doxygen/>>.
- [7] *Graphviz - Graph Visualization Software* [online]. [cit. 2009-05-25]. Dostupný z WWW: <<http://www.graphviz.org/>>.
- [8] *Interval.cz* [online]. [cit. 2009-05-14]. Dostupný z WWW: <<http://interval.cz/>>.
- [9] *PHP triky* [online]. 2005-2008 [cit. 2009-05-14]. Dostupný z WWW: <<http://php.vrana.cz/>>.
- [10] *Jak psát web* [online]. [cit. 2009-05-14]. Dostupný z WWW: <<http://www.jakpsatweb.cz/>>.
- [11] *OO, UML, analýza, metodologie* [online]. 2005 [cit. 2009-05-14]. Dostupný z WWW: <<http://mpavus.wz.cz/>>.
- [12] *MujMAC* [online]. 1998-2006 [cit. 2009-05-14]. Dostupný z WWW: <<http://mujmac.cz/>>.
- [13] *Wikipedia* [online]. 2001 [cit. 2009-05-25]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Main_Page>.
- [14] *Wikipedie* [online]. 2001 [cit. 2009-05-25]. Dostupný z WWW: <<http://cs.wikipedia.org/wiki/Wikipedie>>.

- [15] *SQuery Gaming Server Module* [online]. 2005 [cit. 2009-05-25]. Dostupný z WWW: <<http://squery.com/>>.
- [16] LAVIN, L.: *PHP – objektově orientované, koncepty, techniky a kód*. Praha: Grada Publishing, 2009. ISBN 978-80-247-2137-8.
- [17] KOSEK, J.: *PHP - tvorba interaktivních internetových aplikací*. Praha: Grada Publishing, 1999. ISBN 80-7169-373-1.
- [18] WYKE-SMITH, C.: *CSS - Využijte kaskádové styly naplno!* Praha: Computer Press, 2006. ISBN 80-251-1297-7.
- [19] MOMJIAN, B.: *PostgreSQL - Praktický průvodce*. Praha: Computer Press, 2003. ISBN 80-722-6954-2.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AJAX	Asynchronous JavaScript and XML
BLOB	Binary Large Object
C/C++	Programovací jazyky
CLOB	Character Large Object
cookies	Malé množství dat, která jsou serverem posílána prohlížeči a následně ukládána do klientského počítače
Cron	Plánovač úloh UNIXových a Linuxových systémech
CSV	Comma separated values
DHTML	Dynamic HyperText Markup Language
FAI	Fakulta aplikované informatiky
FT	Fakulta technologická
GET	Způsob předávání formulářových dat
GIS	Geographic Information Systém
HTML	HyperText Markup Language
ISO/OSI	International Organization for Standardization/Open Systems Interconnection Reference Model
Java	Programovací jazyk
LDAP	Lightweight Directory Access Protocol
Linux	Druh operačního systému
Oracle	Systém pro řízení báze dat
OS	Operační systém.
Perl	Programovací jazyk
PHP	PHP: Hypertext Preprocessor
PL/pgSQL	Procedural Language/PostgreSQL Structured Query Language

PL/SQL	Procedural Language/Structured Query Language
POST	Způsob předávání formulářových dat
Python	Programovací jazyk
SSL	Secure Sockets Layer
Tcl	Tool Command Language
UAI	Ústav aplikované informatiky
UEM	Ústav elektrotechniky a měření
UNIX	Druh operačního systému
URL	Uniform Resource Locator
WYSIWYG	What You See Is What You Get
WWW	World Wide Web
W3C	World Wide Web Consortium
XHTML	eXtensible Hypertext Markup Language
XML	Extensible Markup Language

SEZNAM OBRÁZKŮ

<i>Obr. 1. Diagram případů užití</i>	18
<i>Obr. 2. Základní architektura systému</i>	30
<i>Obr. 3. Obsluha URL požadavku</i>	37
<i>Obr. 4. Vztah redakčního systému a frameworku</i>	39
<i>Obr. 5. Architektura jádra redakčního systému a modulů</i>	39
<i>Obr. 6. Přihlašování běžného uživatele</i>	40
<i>Obr. 7. Přihlašování vyučujícího</i>	41
<i>Obr. 8. Editace osobních údajů</i>	42
<i>Obr. 9. Zadávání požadavků vyučujících</i>	43
<i>Obr. 10. Zadávání požadavků předmětů</i>	43
<i>Obr. 11. Editace požadavků vyučujících</i>	44
<i>Obr. 12. Editace požadavků předmětů</i>	44
<i>Obr. 13. Editace vyučujících</i>	45
<i>Obr. 14. Export záznamů modulu do CSV</i>	45
<i>Obr. 15. Zadávání deadlinů</i>	46
<i>Obr. 16. Zasilání e-mailových notifikací</i>	47
<i>Obr. 17. Diagram tříd přístupových oprávnění</i>	50
<i>Obr. 18. Příklad nastavení přístupových oprávnění</i>	51
<i>Obr. 19. Diagram tříd pro požadavky vyučujících</i>	54
<i>Obr. 20. Schéma databázových tabulek modulu Mod_Requirements_Teachers</i>	55

SEZNAM PŘÍLOH

P I Postup instalace webového portálu

PŘÍLOHA P I: POSTUP INSTALACE WEBOVÉHO PORTÁLU

Pro úspěšnou instalaci systému je třeba si uvědomit jeho strukturu. Samotný redakční systém a framework jsou odděleny v samostatném adresáři. Webový portál jako takový je pak soubor modulů a tříd, které redakční systém a framework pouze využívají a konfigurují dle svých potřeb.

Popis instalace je pak následující:

1. Zkopírování adresáře obsahu adresáře *dp*, který se nachází na přiloženém CD v adresáři *install*, do adresáře virtualhostu, ve kterém bude aplikace spouštěna.
2. Zkopírování adresáře *rszr*, který se také nachází na přiloženém CD v adresáři *install*, o dvě úrovně adresářů výše, než je virtualhost aplikace. Pokud nelze provést kopírování do této úrovně, zvolte libovolnou jinou cestu, avšak je pak třeba změnit nastavení cesty v souboru *index.php* ve virtualhostu aplikace k souboru *boot.php*, který se nachází právě v adresáři *rszr*.
3. Vytvořil novou databázi v PostgreSQL, která bude obsahovat jazyk PL/pgSQL a funkce, datové typy a operátory modulu Ltree.
4. Upravit konfiguraci připojení k databázi v souboru *pages/db/config.php*, který se nachází ve virtualhostu aplikace, pro novou databázi.
5. Upravit konfiguraci `$config[RsInstall::INSTALATION_ALLOWED]` na hodnotu *true* v souboru *admin/rsinstall/config.php*, který se nachází ve virtualhostu aplikace.
6. V prohlížeči zadat adresu virtualhostu aplikace s cestou */zradm/install* (např. <http://www.example.com/zradm/install>)