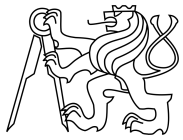
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Systém pro hodnocení pedagogického výkonu - verze 3.2
Student:	Milan Resch
Vedoucí:	Ing. Michal Valenta, Ph.D.
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Cílem práce je kompletní revize systému z pohledu udržitelnosti kódu, dále pak doplnění a přepracování některých funkcí stávající verze.

1. Seznamte se s aktuální verzí systému.
2. Analyticky, návrhově a implementačně doplňte/přepracujte tyto funkce:
- zadávání a zobrazování podílu vyučujících na klasifikaci předmětu,
- import dat přes KOSApi - zejména možnost filtrace importovaných objektů a vedení/oponování ZP,
- u zobrazení pro učitele doplnit náhled celého předmětu,
- zobrazování (pro náhled učitelů) případně zadávání (pro úvazkáře) vedení ZP,
- podpora přihlášení pomocí SSO shibboleth.
3. Pomocí metod SW inženýrství vytvořte dokumentaci a proveďte revizi zdrojového kódu, aby mohla být celá aplikace předána do správy oddělení ICT na FIT.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 15. listopadu 2016

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

Bakalářská práce

Systém pro hodnocení pedagogického výkonu - verze 3.2

Milan Resch

Vedoucí práce: Ing. Michal Valenta, Ph.D.

9. ledna 2017

Poděkování

Chtěl bych poděkovat vedoucímu mé bakalářské práce, Ing. Michal Valenta, Ph.D., za jeho vstřícný přístup a velkou trpělivost, kterou se mnou měl.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 9. ledna 2017 .....

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2017 Milan Resch. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí, je nezbytný souhlas autora.

Odkaz na tuto práci

Resch, Milan. *Systém pro hodnocení pedagogického výkonu - verze 3.2.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato bakalářská práce se věnuje informačnímu systému Podpora tvorby učitelských úvazků, který slouží pro měření pedagogického výkonu na Fakultě informačních technologií ČVUT v Praze. Při práci byly provedeno několik úprav tohoto systému. V práci je zdokumentována současná verze systému spolu s úpravami, které byly provedeny.

Klíčová slova učitelský úvazek, započitatelné hodiny, smarty, databáze, dokumentace, import dat

Abstract

This bachelor thesis is about information system Support for the creation of teacher's harness which is used for the rating of teacher duties at The Faculty of Information Technology of CTU in Prague. Modification of this system were made during this thesis. This thesis contains a description of present version of the system and the modifications made on the system.

Keywords teacher's harness, accounted hours, database, documentation, data import

Obsah

Úvod **1**
Seznámení s problematikou 1
Motivace 2
Výstup práce 2
Struktura práce 2

1 Analýza původní verze systému **3**
1.1 Historie vývoje systému 3
1.2 Funkcionality 3
1.3 Problémy verze 3.0 6
1.4 Změny provedené v rámci této práce 7

2 Implementace **17**
2.1 Model 18
2.2 Controller 23
2.3 View 32

3 Testování **35**
3.1 Automatizované testy 35
3.2 Manuální testování 36

4 Datové úložiště **37**
4.1 Schéma uvazky 37
4.2 Schéma funkce 38

5 Možnosti dalšího rozšíření **47**
5.1 UsermapApi 47
5.2 Shibboleth 47

Závěr **49**

Literatura **51**

A Seznam použitých zkratek **53**

B Obsah přiloženého CD **55**

Seznam obrázků

1.1 Diagram užití. 4
1.2 Odkaz na přehled. 8
1.3 Filtr pro nastavení rozsahu importu oponentur. 9
1.4 Nabídka dodatečného importu. 10
1.5 Detail předmětu v rozhraní učitele. 11
1.6 Pokrytí předmětů bez učitelů. 12
1.7 Chyba zobrazení zbývající garance. 13
1.8 Přehled parametrů semestru. 15
1.9 Sekce Oponentury v náhledu detailu učitele. 16
1.10 Rozkliknutí výpočtu ZH v detailu učitele. 16

2.1 MVC architektura. 17
2.2 Diagram tříd Modelu. 18
2.3 Diagram tříd Controlleru. 24
2.4 Přidat předmět. 27
2.5 Přidat předmět po zvolení předmětu. 27

3.1 Možnost validace v detailu učitele. 35
3.2 Výsledek validace v detailu učitele. 36

4.1 Mapa vzájemného volání funkcí ve schématu *funkce*. 39
4.2 Mapa volání souhrnných funkcí ve schématu *funkce*. 40

Úvod

Tato bakalářská práce se zabývá pokračováním vývoje systému pro Podporu tvorby učitelských úvazků, také známého jako *Úvazkostoj*, který vznikl v rámci bakalářských prací bc. Josefa Němečka[1] a Ing. Jakuba Štefana[2]. Funkcí tohoto systému je usnadnit monitorování pracovního vytížení učitelů, kteří se podílejí na výuce na Fakultě informačních technologií. V této práci bude tento systém upraven a spolu s provedenými úpravami popsán.

Seznámení s problematikou

Každý učitel na FIT vykonává v rámci svého úvazku v průběhu semestru mnoho různých činností: *přednáší, vede proseminář, učí cvičení, vede nebo oponuje závěrečné práce.* Na základě těchto jeho činností je učitel ohodnocen za každou z nich *započitatelnými hodinami (ZH)*, které určují míru jeho zatížení v každém semestru.

Toto měření pracovního zatížení je velice důležitou aktivitou. Učitelé pracující na FIT jsou zaměstnáni v rozdílných formách úvazků a každá z těchto forem má stanovenou nějakou minimální hodnotu ZH, kterou musí učitel v semestru absolvovat. Díky měření získají zaměstnavatelé přesný přehled o tom, jak je který zaměstnanec vytížen a do jaké míry plní svůj úvazek.

Na základě těchto informací je možno učitelům, kteří ještě nenaplnili míru ZH odpovídající jejich formě úvazku, přidělit další povinnosti. Také tato funkce umožňuje pracovníky motivovat k většímu nasazení příslibem odměn při splnění zvolených podmínek. Při opakovaném překročení požadovaného počtu ZH lze také učiteli zvýšit jeho úvazek (např.: částečný úvazek → plný úvazek), nebo při opakovaném nenaplnění formu úvazku snížit.

Motivace

Současná verze systému se liší od té, kterou v rámci svých prací vypracovali bc. Josef Němeček a Ing. Jakub Štefan, byla upravena databázová část systému. Dále při užívání tohoto systému byly zjištěny problémy v oblasti importu dat. Motivací této práce je opravit odhalené problémy, přidat mu některé dodatečné funkce a zdokumentovat tento systém před jeho předáním k plnému nasazení na FIT.

Výstup práce

Výstupem této bakalářské práce je plně funkční a zdokumentovaná verze systému Podpory tvorby učitelských úvazků, který nabízí svým uživatelům rozšířenou množinu funkcí, a který je možné předat k plnému nasazení na FIT.

Struktura práce

Hlavním cílem této práce je vytvořit dokumentaci současné verze systému. Tato práce je členěna do pěti hlavních kapitol. V těchto kapitolách popíšu základní vlastnosti systému a jejich realizaci.

První kapitolou je Analýza původní verze systému. V této kapitole je popsána historie vývoje systému, jaké možnosti nabízí svým uživatelům a jaké problémy byly odhaleny během jeho užívání. V závěru kapitoly je uvedena analýza problémů, které byly v rámci této práce odstraněny, spolu s jejich řešením.

V druhé kapitole s názvem Implementace jsou popsány metody implementace systému a popis jeho hlavních částí společně s technologiemi, které byly využity.

Třetí kapitola jménem Datové úložiště se věnuje databázi, ve které jsou uloženy data systému. Je zde popsána struktura této databáze a její části.

V kapitole Testování, která následuje, je popsáno, jaké typy automatických testů systém nabízí a jak byl před svým nasazením testován.

Poslední kapitolou jsou Možnosti rozšíření, kde jsou zmíněny možnosti dalšího vývoje systému, které již nebyly v rámci této práce splněny.

Kapitola 1

Analýza původní verze systému

1.1 Historie vývoje systému

Systém Podpory tvorby učitelských úvazků, který je v současné době využíván pro měření pedagogického výkonu na FIT, je již třetí verzí tohoto systému. Vznikl na základě úprav verze 3.0, která byla vytvořena v rámci bakalářských prací bc. Josefa Němečka a Ing. Jakuba Štefana v roce 2014 a byla následně spuštěna na FIT.

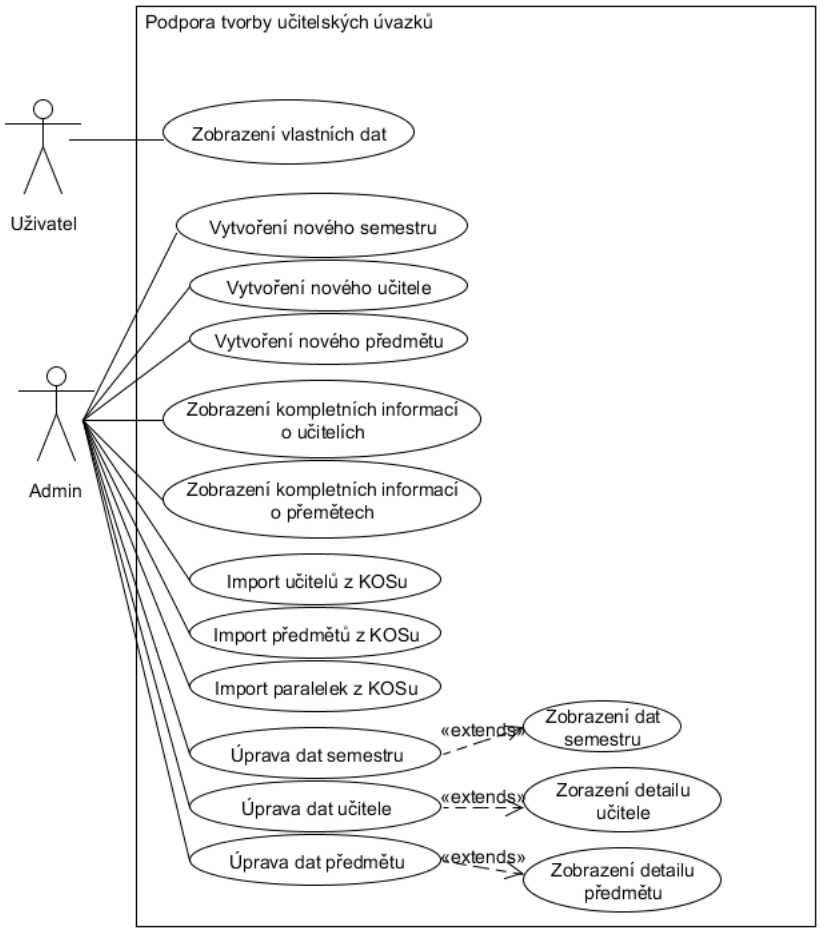
Jejich verze vznikla na základě webové aplikace, kterou roku 2011 v rámci své bakalářské práce před nimi vytvořil bc. Jan Toman. Během analýzy verze 2.0 se Josef Němeček s Jakubem Štefanem rozhodli, že tato verze již není vhodná k dalšímu vývoji, a proto byla vytvořena zcela nová verze systému 3.0.

Během užívání tohoto systému však byly odhaleny nedostatky, které bylo nutno odstranit. Pod vedením Ing. Michala Valenty Ph.D. tak vznikla roku 2015 za spolupráce s bc. Němečkem a Ing. Štefanem verze 3.1, na které je založena tato práce.

1.2 Funkcionality

Na FIT existuje na každé katedře speciální pozice *úvazkář*, jehož povinností je hlídat naplnění úvazku zaměstnanců katedry. Pro úvazkáře je existence tohoto systému velice užitečná, protože jim zprostředkovává informace o všech učitelích a jejich vytížení. Během analýzy problému a konzultace s úvazkáři, byly pro systém vytyčeny základní funkcionality, které by měl systém splňovat. Tyto funkcionality můžeme vidět na obr. 1.1.

Jak můžeme vidět, byly v systému vytvořeny dvě základní role. První rolí je role klasického uživatele, která je vytvořena pro zaměstnance FIT. Tato role umožňuje uživateli jedinou funkci v systému a tou je zobrazení informací o jeho úvazku v posledních semestrech.



Obrázek 1.1: Diagram užití.

Druhou rolí je role admina, která byla vytvořena pro úvazkáře. Pro účely této role byly vytvořeny v systému účty *admin_kam*, *admin_kcn*, *admin_kps*, *admin_ksi* a *admin_kti*, každá pro jednu katedru. Po přihlášení na kterýkoliv z těchto účtů se úvazkář dostane do *administrativního rozhraní*, které úvazkáři poskytuje další funkcionality systému.

Mezi tyto funkce patří:

- *Vytvoření nového semestru* - úvazkář vytvoří v databázi nové schéma na základě kódu semestru, které bude dostupné pro všechny ostatní uživatele systému, do tohoto schématu pak bude moci on i ostatní úvazkáři vkládat data o vytížení učitelů na FIT.
- *Vytvoření nového učitele* - vytvoření nového učitele přidá nový záznam do databáze. Tato funkce nabízí nastavit pro učitele jeho výši úvazku,

ke které katedře patří, jeho pracovní poměr a ID v KOS. Povinným parametrem pro vytvoření nového učitele je jeho `username`, podle kterého je rozpoznán v databázi. Tato funkce je vhodná především pro vložení nového učitele, který není evidován v KOSu.

- *Vytvoření nového předmětu* - vytvoření nového předmětu stejně jako u učitele vytvoří v databázi nový záznam o předmětu, možnosti nastavení jsou však mnohem rozsáhlejší, protože předmět obsahuje více parametrů, a je zde i možnost rovnou vytvořit pro předmět paralelky přednášek a proseminářů.

- *Zobrazení kompletních dat o učitelích* - tato funkce slouží pro rychlou orientaci o učitelích a jejich vytížení. Úvazkáři je zde předložen kompletní seznam učitelů s informacemi o jejich úvazku a počtu ZH ve vybraném semestru. Z této nabídky lze pak přejít na detail jednotlivých učitelů, kde jsou úvazkáři předány podrobné informace o složkách činnosti učitele.

- *Zobrazení informací o předmětech* - informace o předmětech jsou úvazkáři předány ve dvou formách. První z nich je přehled o *pokrytí* předmětů, v této části je úvazkář seznámen se stavem předmětu (kolik chybí obsadit přednášek, proseminářů, cvičení, atd.). Druhou variantou je informace o *náročnosti* předmětu, která obsahuje data o počtu ZH za daný předmět. Z obou těchto variant se lze dostat na detail předmětu, kde jsou bližší informace o předmětu společně s učiteli, kteří se na něm podílejí.

- *Import učitelů z KOS* - import dat z databáze systému KOS značně usnadňuje úvazkáři práci. Kdyby musel totiž každého učitele zadávat manuálně, velice by se tím zvýšila náročnost této pozice. Systém automaticky načte z KOSu učitele FITu a nabídne je úvazkáři, který z nabídky vybere pouze ty učitele, které chce importovat do databáze systému.

- *Import předmětů z KOS* - pro import předmětů platí ve své podstatě to samé, systém získá informace o předmětech, které jsou pro daný typ semestru (*zimní/letní/obojí*), protože je zbytečné importovat předměty, které v daném semestru vypsané nejsou a zabíraly by jen místo v DB. Ty pak nabídne úvazkáři k uložení.

- *Import paralelek z KOS* - pro import paralelek platí stejné podmínky jako pro předmět a tak nemá smysl to více rozepisovat.

- *Úprava dat semestru* - úprava dat semestru probíhá ve dvou rovinách. První je úprava základních dat o semestru, jako je jméno, doba trvání, atd. Druhou je pak úprava parametrů pro daný semestr, na základě nichž jsou vypočítávány ZH.

- *Úprava dat učitele* - jedná se o možnost, která je nabídnuta úvazkáři v detailu učitele. Úvazkář zde může učiteli změnit jeho pracovní informace, nebo mu přidat, či odebrat povinnosti.

- *Úprava dat předmětu* - úprava předmětu probíhá stejně jako u učitele v jeho detailu. Úvazkář může v tomto detailu změnit veškeré informace, které se týkají jeho popisu, a může mu vytvořit i nové paralelky pro cvičení. Také může k předmětu přidat učitele, který se na jeho výuce podílí nebo odebrat učitele, který se na něm již nepodílí.

1.3 Problémy verze 3.0

1.3.1 Redundance dat v DB

Problémů, které byly během práce s verzí 3.0, je několik. Prvním, který zmíním, je redundance dat v datovém úložišti. Data v databázi systému jsou zpracovávána pomocí funkcí, které jsou pro každý semestr stejné. V původní verzi jsou však obsaženy v každém schématu. Další redundantní informací, která byla obsažena v každém schématu, byla tabulka s informacemi pro přihlášení administrátorů (úvazkářů) a tabulka s parametry pro výpočet ZH, která obsahovala jedinou řádku. Toto bylo způsobeno tím, že systém byl nastaven tak, aby se dotazoval vždy pouze nad aktuálně zvoleným semestrem.

Tento problém byl později vyřešen ve verzi 3.1, kdy byly tyto informace vyčleněny do samostatného schématu *funkce*. Toto schéma je blíže popsáno v kapitole Databázové úložiště.

1.3.2 Neúplnost a nesprávnost importovaných dat

Dalším problémem byla neúplnost a nesprávnost dat importovaných z KOSu. Jak už je uvedeno výše, systém importuje pouze učitele, kteří jsou zaměstnanci FIT, a předměty, které jsou určeny pro daný typ semestru. Na FITu však působí i zaměstnanci jiných fakult nebo externisté, kteří nespadají pod žádnou fakultu. Některé předměty také bývají vypsány i mimo sezónu, pro kterou jsou určeny, nebo nemají ve svém popisu tuto informaci uvedenu. Co se nesprávnosti dat týče, ta se týká importovaných dat oponentur. U těchto dat byla pozorována odchylka od skutečných dat.

1.3.3 Náhled vlastních dat úvazkářů

Jak je výše vidět v seznamu funkcionalit (obr. 1.1), nemá úvazkář katedry automaticky možnost prohlédnout si vlastní podrobný rozpis svého vytížení, jaký je nabídnut běžnému uživateli. Jedinou možností, jak může tyto informace získat během svého přihlášení v roli administrátora, je zobrazit si vlastní detail učitele z přehledu všech učitelů. Další možností, jak k těmto informacím získat

přístup, je odhlásit se z účtu úvazkáře a přihlásit se pomocí běžného účtu. Tím ale přijde o možnosti správy systému, které jsou úvazkáři nabídnuty.

1.3.4 Nepřehlednost dat o oponenturách a vedení ZP

V systému jsou ukládána data o oponenturách a vedení závěrečných prací (ZP) ve dvou formách. První je tabulka `oponentura`, která ve svých sloupcích uchovává informace o počtu oponentur. Druhou formou je pak uložení dat o vedení ZP prostřednictvím tabulky `uci`. V té je pro každého učitele vytvořen řádek pro předmět BI-BAP nebo MI-DIP na základě toho, jaké ZP vede. Informace o počtu vedených prací jsou pak uvedeny ve sloupci s počtem vedených projektů. Informace o podílu na ZP jsou pak v přehledu uvedeny na třech místech - u předmětů BI-BAP a MI-DIP a v části s oponenturami.

1.3.5 Nezobrazování pokrytí předmětů bez učitelů

V přehledu pokrytí jednotlivých předmětů, nejsou uvedeny předměty, které nikdo nevyučuje. Nastává pak situace, kdy úvazkář chce přiřadit učitele k předmětu, ale předmět nenalezne. Snaží se proto předmět vytvořit manuálně, ale je mu vrácena chyba, protože předmět již v systému je, jen není zobrazen. V takovém případě musí k detailu předmětu přistupovat buď zadáním adresy detailu, nebo jej vyhledá v seznamu náročnosti předmětů.

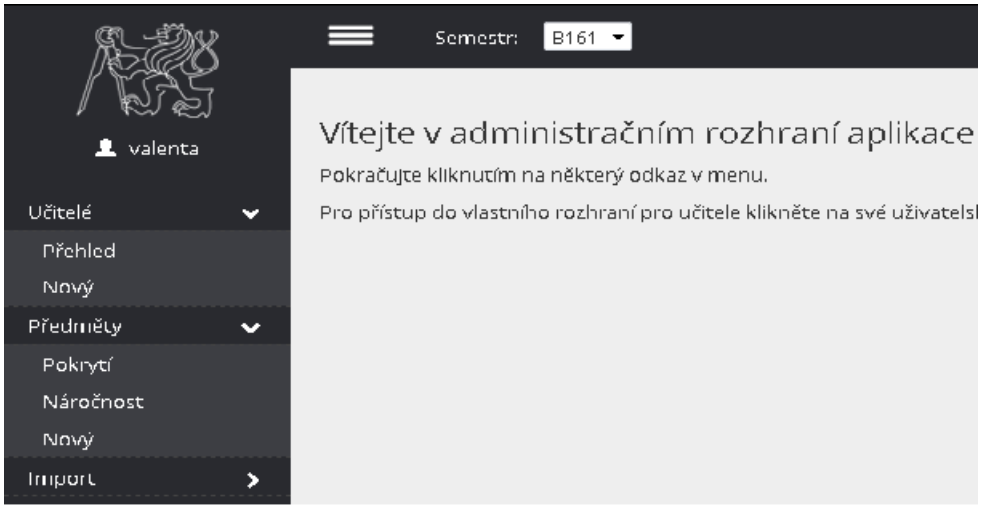
1.4 Změny provedené v rámci této práce

V rámci této práce byly vytvořeny a implementovány řešení některých z výše uvedených problémů. V této sekci budou uvedeny tyto problémy i s jejich analýzou.

1.4.1 Nabídka zobrazení vlastního detailu pro úvazkáře

Tato vlastnost systému, jak už bylo zmíněno, úvazkáři značně usnadní získání přehledu o vlastním úvazku v posledních semestrech. Problémem současné verze je, že tento náhled je uživateli zprostředkován na základě uživatelského jména, pod kterým se přihlásil do systému. Systém použije toto uživatelské jméno a získá k němu informace, které jsou uložené v databázi. Pro uživatelská jména uvazkářů, která mají formu *admin_katedra*, však nejsou v databázi uložená žádná data.

Tento problém byl vyřešen pomocí úpravy dat, která jsou uložena v tabulce `admins`. Do ní byl přidán sloupec `uname`, který obsahuje uživatelská jména úvazkářů. Při přihlášení je pak do `$_SESSION['username']` uložena hodnota tohoto sloupce místo uživatelského jména, které úvazkář použil. Úvazkář pak může přejít na stránku svého přehledu pomocí odkazu s jeho uživatelským jménem v levém horním rohu, který je vidět na obrázku 1.2.



Obrázek 1.2: Odkaz na přehled.

V systému byla připravena i další možnost, kdy se úvazkář přihlásí do systému v roli administrátora pomocí svého fakultního jména a hesla. Za tímto účelem byl do tabulky *admins* přidán ještě jeden sloupec s názvem `overeni`. Pokud je uživatelské jméno v tabulce a hodnota `overeni` nastavena na `local`, je heslo porovnáno s tím, které je uloženo v databázi. Pokud je hodnota jiná, proběhne ověření pomocí LDAP a v případě úspěchu, je uživatel přihlášen v roli administrátora.

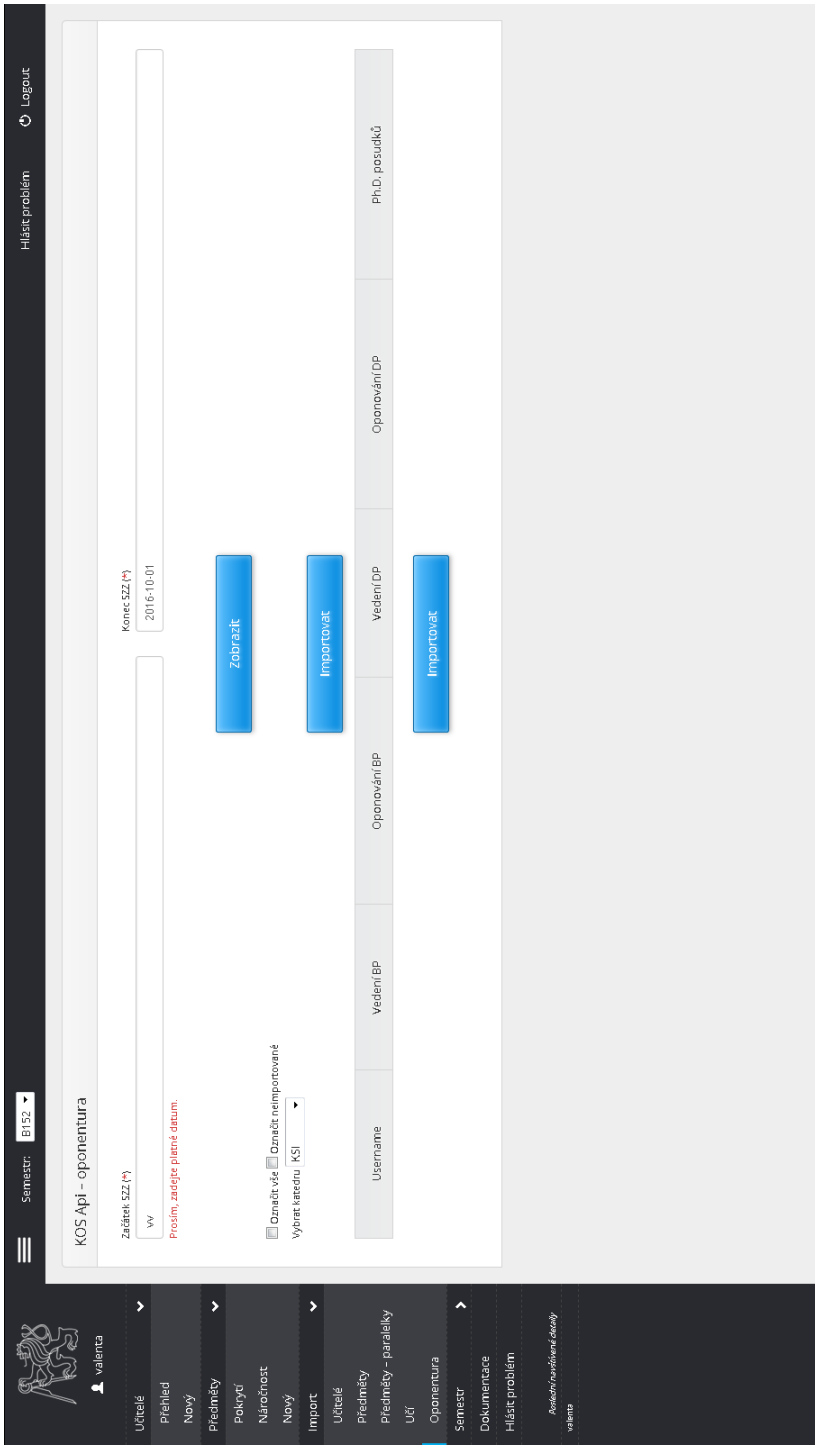
1.4.2 Nesprávnost importovaných dat o ZP

Během importu dat z KOSu o počtu oponovaných a vedených ZP, sledoval pan Ing. Valenta, že data, která jsou zobrazena v některých případech neodpovídají skutečnosti. Po prozkoumání kódu současné verze bylo zjištěno, že problém vzniká při dotazu do KOSu. Původní verze se dotazovala na všechny obhájené práce a z nich pak odstranila ty, jejichž datum obhájení byl vyšší než konec semestru, nebo nižší než začátek semestru. Některé starší záznamy však tuto informaci neobsahovali a byly proto zpracovány také.

V rámci řešení tohoto problému byl proto změněn dotaz pro KOS. Ten nyní získává data o ZP, které byly obhájeny po datu, které úvazkář zadá. Pro tento účel byl na stránku importu oponentur přidán formulář, do kterého úvazkář zadá údaje o časovém rozmezí, ze kterého chce informace importovat. Tento formulář je zobrazen na obrázku 1.3.

1.4.3 Neúplnost importovaných dat

Další problém, který je řešen v této práci, je neúplnost dat, která jsou importována. Důvody, proč nejsou některá data importována, jsou popsány v před-



Obrázek 1.3: Filtr pro nastavení rozsahu importu oponentur.

Obrázek 1.4: Nabídka dodatečného importu.

chozí části a nebudu je zde proto opakovat.

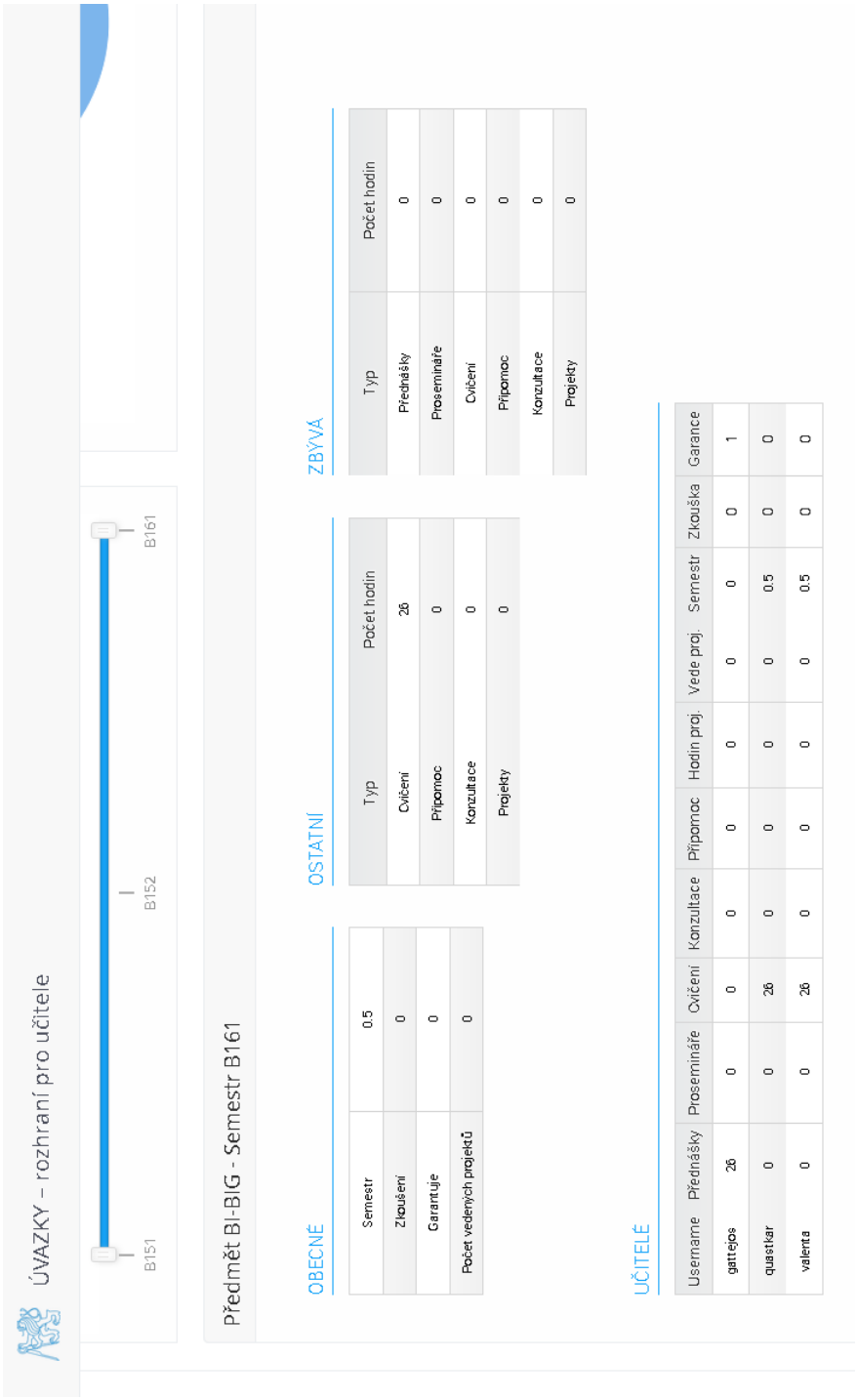
Tato chybějící data jsou patrná při importu *oponentur* a *učí*. Pokud se v těchto sekcích pokusíme o import informací o předmětu nebo učiteli, který není v databázi, je nám vrácena chyba. Při řešení, které bylo vytvořeno, jsou základem právě tyto chyby. Při každém neúspěšném pokusu o uložení dat je vyhodnoceno, co bylo chybou, a pokud se jednalo o chybějící předmět, nebo chybějícího učitele, je informace o jeho `kodu` nebo `username` uložena do pole, které je vráceno úvazkáři společně s informacemi o výsledku importu. Na základě těchto informací je pak úvazkáři nabídnuta možnost dodatečného importu chybějících dat (obr. 1.4).

1.4.4 Doplnění informací do rozhraní učitele

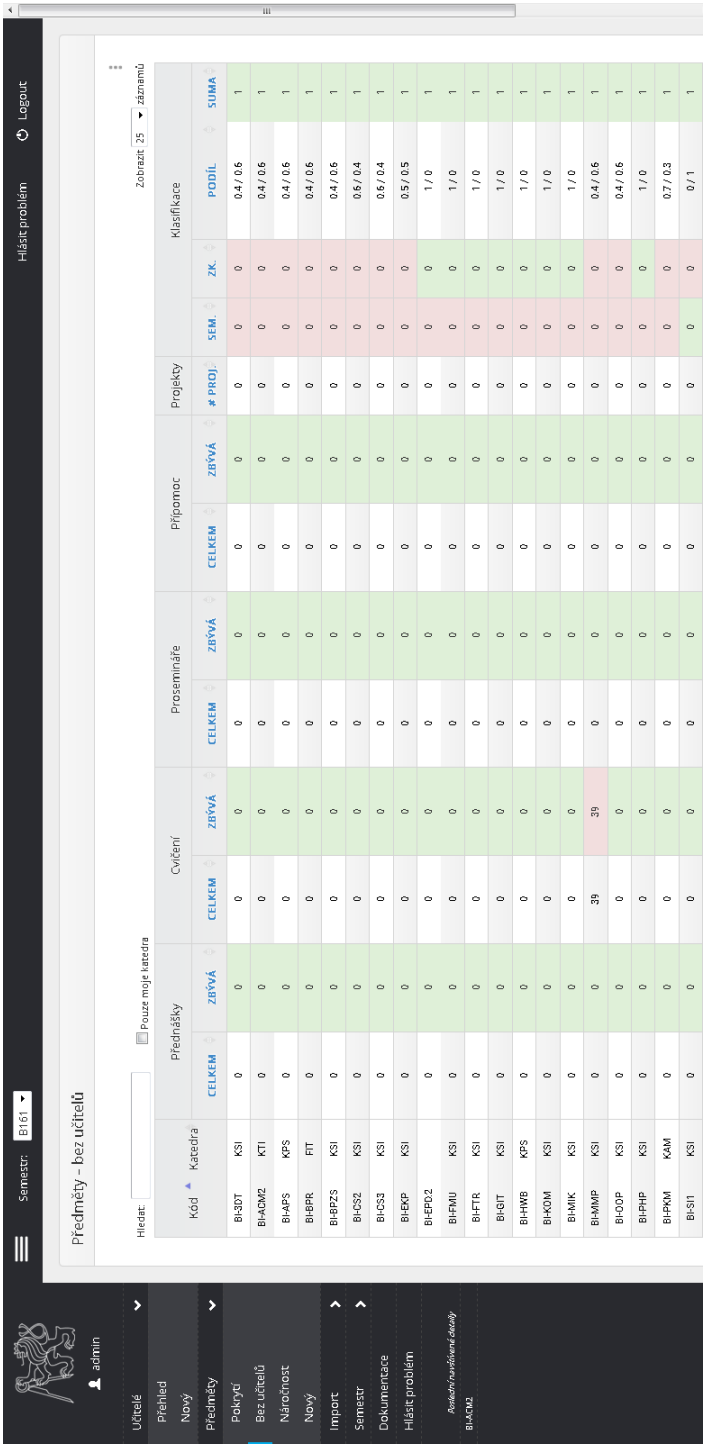
Do rozhraní učitele bylo doplněno několik informací. Byly přidány názvy k jednotlivým semestrům a bylo tak usnadněno jejich rozlišení. Dále byly do detailu předmětu přidány informace o hodinách, které zbývá pokrýt, a byla přidána tabulky s informacemi o ostatních učitelích, kteří předmět vyučují (viz obr. 1.5).

1.4.5 Zobrazení předmětů bez učitelů

Jak už bylo výše zmíněno, verze 3.0 nezobrazuje pokrytí předmětů, které k sobě nemají přiřazeného žádného učitele. To je způsobeno funkcí v databázi, která zprostředkovává data pro tuto stránku. Ta vrací informace pouze o předmětech, které jsou v tabulkách *uci, uci_prednasky* a *uci_proseminare*. Byla proto přidána nová stránka o pokrytí předmětů bez učitelů a pro ni do databáze uložena funkce, která vrací informace o předmětech nenacházejících se v těchto tabulkách.



Obrázek 1.5: Detail předmětu v rozhraní učitele.



Obrázek 1.6: Pokrytí předmětů bez učitelů.

j.	Semestr	Zkouška	Garance	
	0	0	0	
	1	0	0	
	0	0	1	
	0	1	1	
	1	0	1	

Obrázek 1.7: Chyba zobrazení zbývající garance.

1.4.6 Oprava zobrazení zbývající garance

Další chybou, která byla odstraněna, je problém se zobrazením zbývající garance v tabulce s učiteli, která je zobrazována v detailu předmětu. Jak můžeme vidět na obrázku 1.7, je v předposledním řádku stále uvedeno, že chybí pokrýt garance, i když už byla přidělena v plném rozsahu jednomu z učitelů. To je způsobeno tím, že funkce v databázi, která zprostředkovává informace, nevrací data o výši přidělené garance. Byla proto upravena, aby v jejím výstupu byly obsaženy i tyto informace.

1.4.7 Spojení dat o ZP

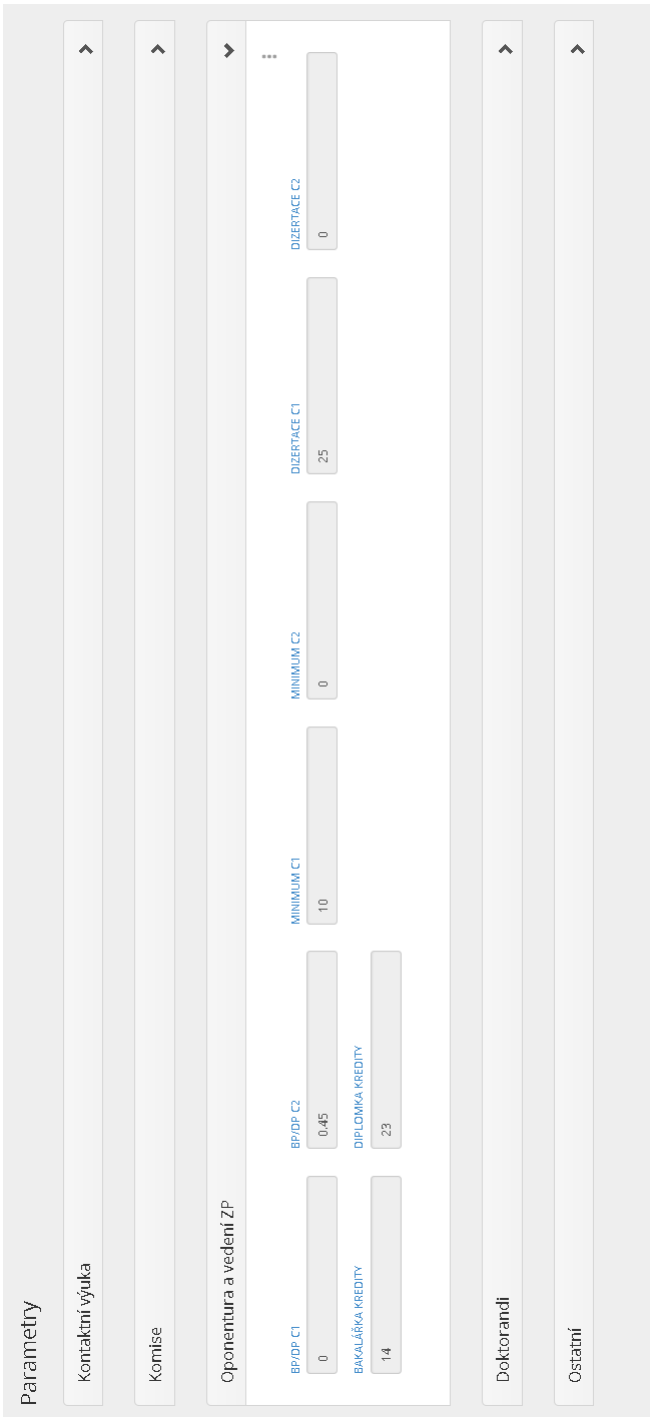
Poslední funkcí vypracovanou v rámci této bakalářské práce, je spojení informací o oponování a vedení ZP. Toto spojení je vhodné hned ze dvou důvodů. Tím prvním je, zlepšení přehlednosti o činnostech učitele týkajících se ZP. Druhým důvodem je úbytek přebytečných dat z databáze. Většina učitelů, kteří vedou nějakou ZP, zároveň alespoň jednu oponují. Vzniká tak nadbytečný záznam v tabulce `uci`, který obsahuje jen jméno učitele, zda se jedná o bakalářskou, nebo diplomovou práci a údaj o počtu prací. Takové záznamy mohou být v tabulce i dva, pokud vede bakalářské i diplomové práce. Navíc se tyto předměty neimportují automaticky a musí se dodatečně do systému vložit.

Jako optimální řešení bylo proto zvoleno sloučit všechny tyto informace do jedné tabulky `oponentura`. Pro vedení bakalářských a diplomových prací byly do této tabulky přidány sloupce `pocet_vedeni_bc` a `pocet_vedeni_dip`, jejichž výchozí hodnota je 0.

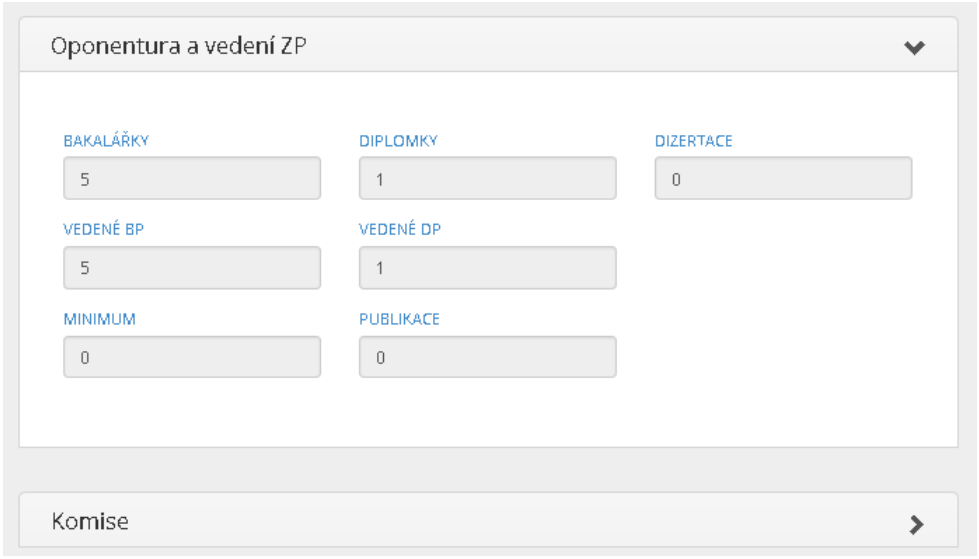
Spolu s touto úpravou, ale byla ztracena informace o hodnotě kreditů u předmětů BI-BAP a MI-DIP, které jsou nutné k vypočítání ZH za daný předmět. Proto byly tyto informace přidány do tabulky `funkce.parametry`

a je možno je upravovat prostřednictvím stránky `semestr/parametry` (viz obr. 1.8).

Další úpravy byly provedeny v rámci zobrazení o podrobnostech oponovaní a vedení ZP v detailu učitele, tyto úpravy jsou vidět na obrázcích 1.9 a 1.10 a bylo upraveno jejich zobrazení i v přehledu vlastního úvazku.



Obrázek 1.8: Přehled parametrů semestru.



Obrázek 1.9: Sekce Oponentury v náhledu detailu učitele.

Obrázek 1.10: Rozkliknutí výpočtu ZH v detailu učitele.

Kapitola 2

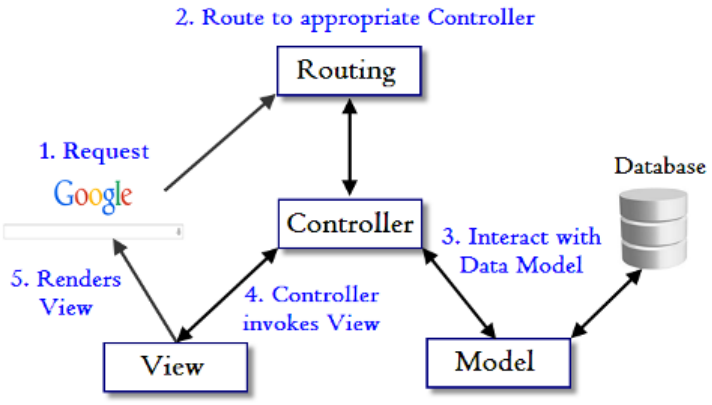
Implementace

Systém je v současné době implementován na základě architektury MVC (obr. 2.1), která ho rozděluje do tří základních částí. Tyto části jsou pak:

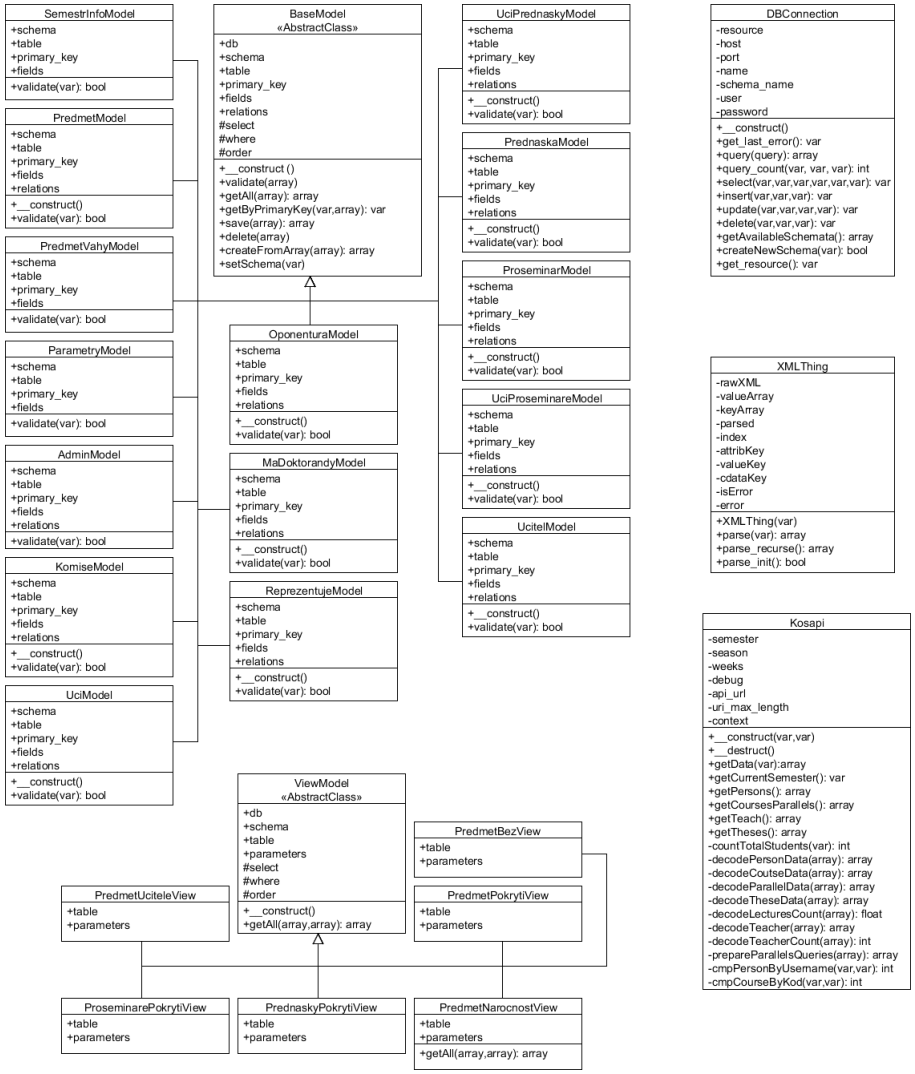
Model (datová vstva) Tato část systému zprostředkovává data z databáze a ostatních systémů (KOSApi) pro doménovou vrstvu.

Controller (doménová vrstva) Hlavní funkcí této vrstvy je zpracovávání dat a reakce na změny vyvolané uživatelem. Tato vrstva přebírá data od datové vrstvy a zpracovává je pro prezentační vrstvu.

View (prezentační vrstva) Tato vrstva se stará o prezentaci dat uživateli aplikace a oznamuje doménové vrstvě události vyvolané uživatelem.



Obrázek 2.1: MVC architektura.



Obrázek 2.2: Diagram tříd Modelu.

2.1 Model

Model neboli datová vrstva je částí systému, která, jak už její název napovídá, se zabývá zprostředkováváním dat pro doménovou vrstvu. V *úvazkostroji* je tato komponenta realizována pomocí několika PHP tříd, které jsou uloženy v adresáři `model`. Jak je možné vidět v diagramu těchto tříd, dají se tyto třídy rozdělit do několika skupin (viz obr. 2.2).

2.1.1 DBConnection

Funkcí této třídy je navázání spojení s databází. Třída obsahuje privátní proměnné, které jsou nutné k navázání spojení s databází, tyto informace získá ze souboru `config.php` v kořenovém adresáři, a spojení samotné. Dále třída obsahuje několik funkcí:

- `get_last_error()` - která vrátí poslední error, který v rámci spojení proběhl.
- `query($query)` - tato funkce provede zadaný dotaz a vrátí jeho výsledek jako pole, všechny následující funkce, kromě funkce `get_resource()`, tuto fukci volají pro vykonání jejich dotazů nad databází.
- `query_count($schema, $table, $where)` - jejím výstupem je počet řádek v tabulce `$table`, které splňují podmínku `$where`.
- `select($schema, $table, $select, $where, $order, $limit)` - tato funkce na základě zadaných dat vytvoří select dotaz pro databázi a zavolá na něj funkci `query`.
- `insert($schema, $table, $instance)` - která z proměnné `$instance` vytvoří SQL příkaz `INSERT INTO` a vykoná ho.
- `update($schema, $table, $pkey, $instance)` - která z proměnné `$instance` vytvoří SQL příkaz `UPDATE` pro klíč `$pkey` a vykoná ho.
- `delete($schema, $table, $pkey)` - tato funkce provede SQL příkaz `DELETE` na základě předaných proměnných.
- `getAvailableSchemata()` - jejím výstupem je pole všech schémat semestrů v databázi.
- `createNewSchema($kod)` - která vytvoří nové schéma v databázi pro semestr s kódem `$kod`.
- `get_resource()` - tato funkce vrátí spojení s databází.

2.1.2 BaseModel

Tato část reprezentuje data uložená v databázi systému. Jejím základem je abstraktní třída `BaseModel`, která je uložena v souboru `model.php`. Třídy `SemestrModel`, `PredmetModel`, `PredmetyVahyModel`, `ParametryModel`, `AdminModel`, `PrednaskaModel`, `ProseminarModel`, `UcitelModel`, `KomiseModel`, `MaDoktorandyModel`, `OponenturaModel`, `ReprezentujeModel`, `UciPrednaskyModel`, `UciProseminarModel`, `UciModel` jsou potomky této třídy v samostatných souborech a každá z nich reprezentuje jednu tabulku v databázi.

`BaseModel` obsahuje informace o připojení k databázi, které je uložené ve veřejné proměnné `$db`, a také informace o tom s jakým schématem a tabulkou je model spojen. Tyto informace jsou v proměnných `$schema` a `$table`. Dalšími proměnnými jsou `$primary_key`, který obsahuje primární klíč tabulky, pole `$fields`, které reprezentuje sloupce tabulky, a pole `$relations`, které obsahuje vazby na jiné modely. Tímto veřejné proměnné této třídy končí a zbývají poslední tři protected, které jsou využívány pro dotazy do databáze. Jsou jimi `$select`, která určuje, jaké sloupce jsou požadovány, `$where`, obsahující na základě jaké podmínky jsou data vybrána, a `$order` určující, jak mají být data seřazena.

Další částí této třídy jsou funkce:

- `__construct()` - což je konstruktor třídy,
- `validate($model)` - která slouží pro ověření správnosti dat uložených v modelu,
- `getAll($options)` - sloužící k získání všech dat z databáze na základě podmínek ve vstupním poli,
- `getByPrimaryKey($primary_key, $options)` - tato funkce přidá do podmínek `$options` podmínku pro rovnost primárního klíče a zavolá funkci `getAll`,
- `save($instances)` - která uloží nebo upraví data v databázi na základě vstupních informací,
- `delete($instances)` - tato funkce slouží k odstranění dat z databáze,
- `createFromArray($array)` - která na základě vstupního pole vytvoří instanci pro předcházející funkce,
- `setSchema($schema)` - jedná se o klasický setter pro proměnnou `$schema`.

2.1.3 ViewModel

Tento model má několik společných vlastností s předcházejícím `BaseModelem`. I u něj je základem abstraktní třída `ViewModel`, která je uložena v souboru `view.php`. Dokonce i proměnné `$db`, `$schema`, `$select`, `$where` a `$order` mají stejný význam. Rozdíl je však v tom, na co instance tohoto modelu odkazují. Zatímco ve třídě `BaseModel` odkazovala proměnná `$table` na název tabulky ve schématu, zde slouží k označení názvu funkce, která je ve schématu definována. Pole `$parameters` pak obsahuje parametry, které se této funkci předávají.

Tento model má pouze dvě funkce, z čehož první je její konstruktor `__construct()` a druhou je funkce `getAll($parameters,$options)`, která volá

funkci z databáze s parametry uloženými v atributu `$parameters` a dodatečnými podmínkami předanými prostřednictvím atributu `$options`.

Od tohoto modelu dědí pět tříd: `PredmetPokrytiView`, `PredmetBezView`, `PrednaskyPokrytiView`, `ProseminarePokrytiView`, `PredmetUciteleView` a `PredmetNarocnostView`. Všechny tyto třídy jsou uloženy v jednom souboru `pokryti.php`

2.1.4 Kosapi a XMLThing

Model `Kosapi`, uložený v `kosapi.php`, je velice důležitou součástí systému, protože pro něj zprostředkovává data z KOSu prostřednictvím *KOSapi*[3] za pomoci RSQL dotazů. Výsledek těchto dotazů je následně zpracován pomocí třídy `XMLThing` a je vrácen pro další zpracování modelem `Kosapi`. Všechny proměnné uložené v modelu `Kosapi` jsou nastaveny na `private`. Těmito proměnnými pak jsou:

- `$semester` - v níž je uložen kód semestru,
- `$season` - v ní je obsažena informace, zda se jedná o zimní, nebo letní semestr,
- `$weeks` - uchovávající informace o počtu týdnů v daném semestru, tato informace je využívána pro vyhodnocení počtu hodin v semestru,
- `$debug` - tato proměnná slouží k signalizaci, zda se mají provedené dotazy pro KOSapi evidovat do logu,
- `$api_url` - zde je uchováván prefix každého dotazu,
- `$uri_max_length` - hodnotou této proměnné je limit pro maximální délku dotazu pro KOSapi,
- `$context` - zde je uchován kontext pro php funkci `file_get_contents`, která je využívána pro získání dat z KOSapi.

Pro získání dat a jejich úpravu pak slouží v tomto modelu funkce:

- `__construct($semester, $weeks)` - konstruktor třídy, který na základě vstupních atributů nastaví vnitřní proměnné a připraví kontext pro komunikaci s KOSapi.
- `getData($param)` - tato funkce slouží pro získání dat z KOSapi, jako vstupní parametr `$param` jí je předán dotaz v RSQL, vrácená hodnota je pak rozparsována pomocí `XMLThing` a výsledek vrácen funkci, která volala `getData()`.
- `getCurrentSemester()` - získá z KOSapi aktuální semestr.

- `getPersons()` - slouží k získání učitelů. Tato funkce má dvě odvětví, ze kterých se vždy provede jen jedno. Prvním je nově implementovaná možnost, která načte informace o chybějících učitelích uložených v poli `$_SESSION['missing_teachers']`, které bylo vytvořeno na základě chyb při importu dat. Druhá větev slouží k importu dat o učitelích, kteří jsou zaměstnanci FIT. Funkce využívá privátní funkci `decodePersonData ($person)` pro zpracování dat o učiteli, která byla vrácena z `getData ($param)`.

- `getCoursesParallels()` - tato funkce stejně jako ta předešlá má na začátku dvě větve, které získávají rozdílná data a ty jsou pak dále zpracovány. První vytvoří dotaz z pole `$_SESSION['missing_courses']`, ve kterém jsou chybějící předměty, objevené během pokusu o import. Druhá získává data o předmětech, které se v dané sezóně (zimní/letní semestr) vyučují. Za pomoci funkce `decodeCourseData($course)` následně zpracuje data do pole `$courses`. Z toho pak vytvoří pomocí funkce `prepareParallelsQueries($courses)` dotazy do KOSapi, z jejichž výstupu dostane informace o paralelkách jednotlivých předmětů (přednášky, proseminář, cvičení).

- `getTeach()` - tato funkce slouží pro import dat o tom, co který učitel vyučuje. Získá seznam všech předmětů a k nim pomocí `prepareParallelsQueries ($courses)` vytvoří dotazy na paralelky v daném semestru. Ty následně převede pomocí `decodeParallelData($parallel)`. Z výsledku pak vytvoří pole přednášek, proseminářů a cvičení, které vrátí.

- `getTheses()` - zavolá funkci `getData($param)` pro všechny obhajoby ZP od data uloženého v proměnné `$_SESSION['oponentura_start']`. Z výsledných dat získá ke každé obhajobě pomocí funkce `decodeThese ($these)` jen informace důležité pro systém a ty vrátí v poli.

- `countTotalStudents($course)` - pro zadaný předmět vrátí celkový počet jeho studentů.

- `decodePersonData($person)` - tato funkce převede složité vstupní pole na jednodušší (př.: `$person['atom:content']['username']` vloží do `username`).

- `decodeCourseData($course)` - převede složité vstupní pole na jednodušší.

- `decodeParallelData($parallel)` - zjednoduší vstupní pole a dodá k němu informace o počtu hodin, uživatelská jména učitelů a počet vyučujících pomocí funkcí `decodeLecturesCount($timetablesSlot)`, `decodeTeacher ($teacher)` a `decodeTeacherCount($teacher)`.

- `decodeTheseData($these)` - funkce ze vstupního pole údajů o ZP získá informace o vedoucím, oponentovi, typu a datu obhajoby a tyto informace vrátí v poli.
- `decodeLecturesCount($timetablesSlot)` - na základě vstupních informací a proměnné `$weeks` vypočítá počet hodin paralelky za semestr.
- `decodeTeacher($teacher)` - tato funkce získá ze vstupního řetězce, nebo pole řetězců, uživatelské jméno učitele/jména učitelů.
- `decodeTeacherCount($teacher)` - vrátí počet záznamů ve vstupním poli.
- `prepareParallelsQueries($courses)` - tato funkce vytvoří seznam dotazů na paralelky předmětů uložených ve vstupním poli `$courses`. Pokud by dotaz, po přidání dalšího předmětu, překračoval maximální délku `$max_uri_length`, uloží se do výstupního pole `$queries` a začne se tvořit další dotaz.
- `cmpPersonByUsername($a, $b)` - porovnává dva vstupní učitele podle hodnoty `username`.
- `cmpCourseByKod($a, $b)` - porovnává dva vstupní předměty podle `kódu`.

2.2 Controller

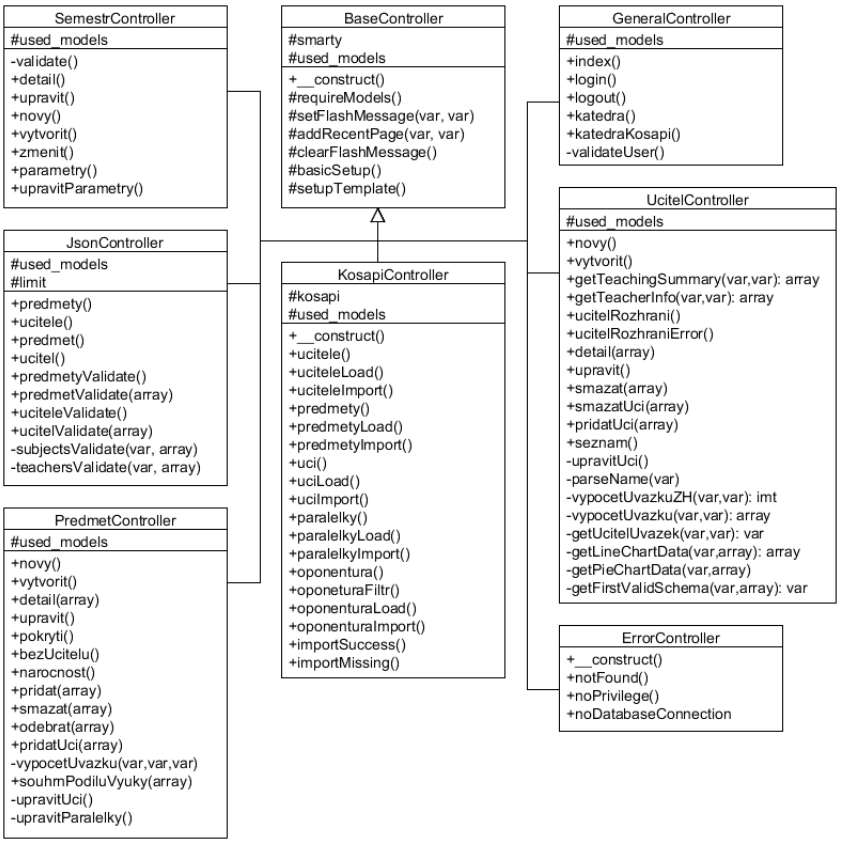
Controller, jinak také zvaný doménová vrstva, je stejně jako model tvořen množinou PHP tříd 2.3, z nichž každá je uložena v adresáři `controller` v samostatném souboru. Tato vrstva slouží k předávání dat prezentační vrstvě a k reakci na podněty uživatele.

2.2.1 BaseController

Základem je třída `BaseController` v souboru `controller.php`, která je předkem pro všechny ostatní třídy doménové vrstvy. Tato třída obsahuje 2 proměnné. První je `$smarty`, což je objekt vytvořený pro šablonovací systém *Smarty*, který je blíže popsán v sekci *View*. Duhou proměnnou je pole `$used_models`, které obsahuje seznam modelů potřebných pro tuto třídu.

Dále tato třída obsahuje funkce:

- `__construct()` - konstruktor této třídy, který získá potřebné modely třídy pomocí metody `requireModels()` a pomocí `setupTemplate()` nastaví důležité proměnné do `$_SESSION` a `$smarty` pro renderování šablon.
- `requireModels()` - získá potřebné modely třídy, uložené v proměnné `$used_models`.



Obrázek 2.3: Diagram tříd Controlleru.

- `setFlashMessage($status, $text)` - naství do `$_SESSION` proměnných informace o flashové zprávě předané jako parametry této funkce.

- `addRecentPage($url, $text)` - slouží k ukládání stránky s adresou `$url` a popisem `$text` do seznamu naposledy navštívených detailů předmětů a učitelů.

- `clearFlashMessage()` - odstraní ze `$_SESSION` informace o flashové zprávě.

- `basicSetup()` - slouží k nastavení základních proměnných pro načítání šablon stránek.

- `setupTemplate()` - nastaví základní `$smarty` proměnné pro renderování šablon a základní proměnné `$_SESSION`.

2.2.2 ErrorController

Tato controller třída slouží k zobrazování chybových stránek. Má pouze konstruktor, který nastaví základní šablonovací proměnné, a tři funkce pro základní problémy. První je funkce `notFound()`, která je zavolána, pokud nebyla požadovaná stránka nalezena. Druhou je `noPrivilege()`, ta je zavolána v případě, že se uživatel snaží zobrazit stránku, pro níž nemá oprávnění. Poslední je `noDatabaseConnection()`, která je volána v případě, že se nepodařilo navázat spojení s databází. Všechny tyto funkce nastaví do `$smarty` kód a popis chyby, která nastala, a zobrazí stránku `error.tpl`.

2.2.3 GeneralController

V této třídě jsou hlavně funkce sloužící pro přihlášení uživatele. Těmito funkcemi jsou:

- `index()` - zobrazí stránku `index.tpl`.
- `login()` - tato funkce ověří přihlašovací údaje pomocí funkce `validateUser()` a na základě výsledku uživateli zobrazí příslušnou stránku.
- `logout()` - ukončí přihlášení uživatele.
- `katedra()` - nastaví filtr katedry pro zobrazovaná data na stránkách.
- `katedraKosapi()` - nastaví filtr katedry pro import dat z KOS Api.
- `validateUser()` - tato private funkce slouží k ověření správnosti údajů zadaných při přihlášení a nastavení práv. Pokud je uživatelské jméno v tabulce *admins* a souhlasí heslo, je uživatel přihlášen jako administrátor. Pokud zadané jméno není mezi administrátory, jsou údaje ověřeny na LDAP serveru a při úspěšném ověření je uživatel přihlášen jako učitel.

2.2.4 JsonController

Tato třída controlleru obsahuje chráněnou proměnnou `$limit`, funkce umožňující přidání učitele k předmětu a předmětu k učiteli a funkce sloužící k validaci uložených dat v databázi. Seznam funkcí:

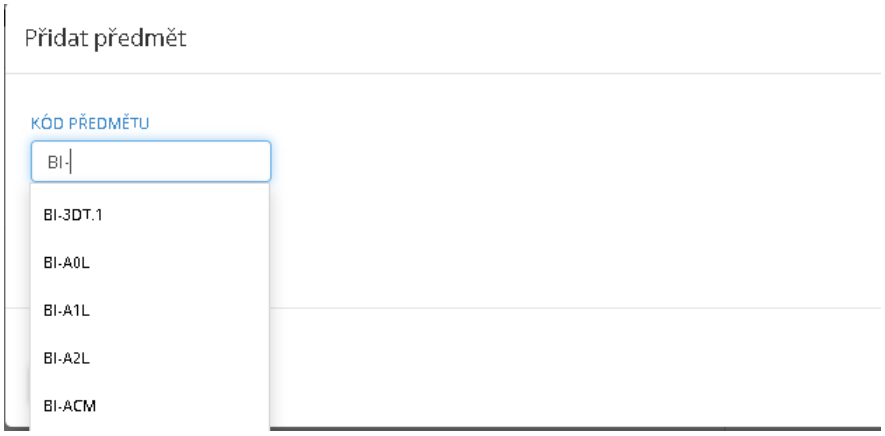
- `predmety()` - získá z databáze předměty jejichž kód obsahuje řetězec zadávaný uživatelem (viz obr. 2.4). Pro dotaz nastaví limit výsledků podle proměnné `$limit`.
- `ucitele()` - získává z databáze učitele, jejichž username obsahuje zadaný řetězec.
- `predmet()` - funkce zkontroluje, zda je předmět v databázi a získá informace o jeho pokrytí (obr. 2.5).

- `ucitel()` - zkontroluje, zda existuje učitel se zadaným uživatelským jménem.
- `predmetyValidate()` - podle nastavení filtru katedry získá všechny předměty, nebo předměty jedné katedry a zavolá pro ně funkci `subjectsValidate ($recursion, $subjects)` s rekurzí nastavenou na 1.
- `predmetValidate($params)` - získá ze vstupního pole kód předmětu, načte o něm data z databáze a zavolá pro něj funkci `subjectsValidate` s rekurzí nastavenou na 1.
- `uciteleValidate()` - získá z databáze informace pro všechny učitele a zavolá pro ně funkci `teachersValidate ($recursion, $teachers)` s rekurzí nastavenou na 1. Pokud je nastaven filtr katedry, jsou získáni pouze učitelé dané katedry.
- `ucitelValidate($params)` - získá ze vstupního pole username učitele, načte jeho data z databáze a zavolá pro něj funkci `teachersValidate` s rekurzí nastavenou na 1.
- `subjectsValidate($recursion, $subjects)` - získá ze vstupního pole `$subjects` kódy předmětů a zjistí, zda je u nich v databázi nastaven typ. Pokud je nastaven parametr `$recursion`, je zavolána i funkce `teachersValidate` pro učitele, kteří učí tyto předměty.
- `teachersValidate($recusion, $teachers)` - zjistí, zda je u učitelů ve vstupním poli `$teachers` nastaven jejich pracovní poměr a pokud je poměr roven smlouvě, zkontroluje, zda je nastaven úvazek. Pokud je nastavena rekurze je zavolána i funkce `subjectsValidate` pro předměty, které tito učitelé vyučují.

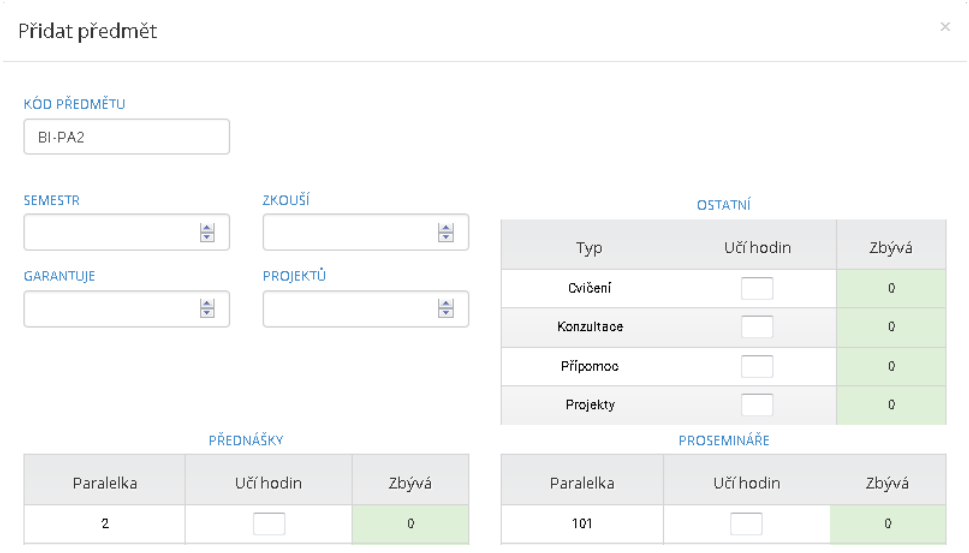
2.2.5 KosapiController

Tento controller slouží pro zobrazování a ukládání dat z KOS Api. Tato třída obsahuje protected proměnnou `$kosapi`, ve které je uložen model Kosapi. Za tímto účelem obsahuje funkce:

- `__contruct()` - zavolá konstruktor rodičovské třídy a vytvoří nový model Kosapi pro semestr, se kterým aktuálně pracujeme.
- `ucitele()` - zobrazí stránku `kosapi_ucitele.tpl`.
- `uciteleLoad()` - načte data o učitelích z KOS Api, zkontroluje, kteří jsou již uloženi v databázi, a zobrazí jejich seznam na stránku pomocí ajaxu.



Obrázek 2.4: Přidat předmět.



Obrázek 2.5: Přidat předmět po zvolení předmětu.

- `uciteleImport()` - funkce se snaží uložit všechny vybrané učitele do databáze a eviduje si informace o výsledcích těchto operací. Po provedení pokusu o uložení všech učitelů, funkce uloží informace o výsledku importu do `$_SESSION` proměnných a zobrazí stránku `import_success.tpl`.

- `predmety()` - zobrazí stránku `kosapi_predmety.tpl`.

- `predmetyLoad()` - načte předměty z KOS Api, zkontroluje, které už jsou v databázi, a zobrazí seznam předmětů pomocí ajaxu.

- `predmetyImport()` - stejně jako u učitelů, funkce se snaží postupně importovat jednotlivé předměty a ukládá si informace o výsledku těchto importů. Na závěr uloží souhrnné informace o výsledku importu a zobrazí stránku `import_success.tpl`.

- `uci()` - zobrazí stránku `kosapi_uci.tpl`.

- `uciLoad()` - načte data z KOS Api o všech vyučovaných přednáškách, proseminářích a cvičení a zobrazí je pomocí ajaxu.

- `uciImport()` - funkce se snaží uložit data získaná v předchozí funkci. Pokud není v databázi nalezen předmět nebo paralelka a nejedná se o předmět AKCE, je kód předmětu uložen do pole `$missing_courses`. Pokud není v databázi nalezen učitel vyučující danou paralelku, je jeho username uložen do pole `$missing_teachers`. Na závěr jsou uloženy do `$_SESSION` informace o výsledku importu a chybějících datech a je zobrazena stránka `import_success.tpl`.

- `paralelky()` - zobrazí stránku `kosapi_paralelky.tpl`.

- `paralelkyLoad()` - načte z KOS Api informace o paralelkách přednášek a proseminářů předmětů, zjistí, které jsou uloženy v databázi, a zobrazí získaná data pomocí ajaxu.

- `paralelkyImport()` - tato funkce ukládá jednotlivé paralelky do databáze a eviduje si výsledky těchto uložení. Na závěr uloží souhrnná data importu do `$_SESSION` proměnných a zobrazí stránku `import_success.tpl`.

- `oponentura()` - zobrazí stránku `kosapi_oponentura` pro import oponentur.

- `oponenturaFiltr()` - nastaví do `$_SESSION` proměnných data od uživatele pro import oponentur a vedení ZP.

- `oponenturaLoad()` - načte data z KOS Api o oponovaní a vedení ZP a zobrazí je pomocí ajaxu.

- `oponenturaImport()` - ukládá záznamy o jednotlivých učitelích a eviduje výsledek tohoto importu. Pokud není importovaný učitel v databázi, je jeho username uložen do pole `$missing_teachers`. Na závěr funkce uloží informace o výsledku importu a chybějících učitelích a zobrazí stránku `import_success.tpl`.

- `importSuccess()` - funkce zpracovává data o výsledcích importu pro stránku `import_success.tpl`.

- `importMissing()` - nastaví `$_SESSION` proměnnou, aby byla načítána data o chybějících záznamech.

2.2.6 PredmetController

Tento controller zpracovává data o předmětech.

- `novy()` - zobrazí stránku `novy_predmet.tpl`.

- `vytvorit()` - zkontroluje, zda není v databázi předmět se stejným kódem a pokud jej nenalezne, tak předmět uloží. Následně uloží i přednášky a proseminář, které s předmětem úvazkář vytvořil.

- `detail($params)` - tato funkce slouží pro načtení detailu předmětu. Funkce přidá odkaz na načítaný detail do seznamu posledních navštívených detailů a načte data o jeho pokrytí z databáze. Následně načte učitele, kteří se na předmětu podílejí, a zobrazí detail předmětu.

- `upravit()` - funkce uloží do databáze změny, které v detailu předmětu nastavil úvazkář.

- `pokryti()` - pokud je nastaven filtr katedry, načte pokrytí předmětů jedné katedry z modelu `PredmetPokrytiView`. Pokud filtr není nastaven, načte pokrytí všech předmětů. Na závěr zobrazí stránku `predmety_pokryti.tpl`.

- `bezUcitelu()` - stejně jako předchozí funkce načte informace v závislosti na nastavení filtru katedry. Načítá informace o pokrytí předmětů, u nichž není žádný učitel, z modelu `PredmetBezView` a zobrazí `predmety_bez.tpl`.

- `narocnost()` - funkce načte předměty na základě nastavení filtru katedry. K nim načte informace o pokrytí a zobrazí stránku `predmety_narocnost.tpl`.

- `pridat($params)` - funkce přidává do databáze k předmětu paralelky proseminářů a přednášek.

- `smazat($params)` - funkce smaže předmět podle kódu, který získá ze vstupního pole `$params`.
- `odebrat($params)` - funkce odebere od předmětu přednášku, proseminář nebo učitele. Pokud odebírá učitele, odebere i všechny záznamy o tom, co vyučuje.
- `pridatUci($params)` - funkce přidá k předmětu učitele. Nejdříve zkontroluje, zda učitel existuje a není už u předmětu uveden. Následně uloží do databáze učitele i informace o tom, co učí.
- `vypocetUvazku($username, $kod, $semester)` - pro zadaná data získá počet ZH učitele za předmět.
- `souhrnPodiluVyuky($params)` - vrátí souhrn podílu výuky pro zadaný předmět přes všechny učitele.
- `upravitUci()` - uloží do databáze změny, které úvazkář provedl v sekci *Učí* detailu předmětu.
- `upravitParalelky()` - uloží změny paralelek přednášek a proseminářů provedené úvazkářem v detailu předmětu.

2.2.7 SemestrController

`SemestrController` slouží pro práci s daty semestru.

- `validate()` - slouží ke kontrole zadaných dat pro vytvoření nového semestru.
- `detail()` - načte z databáze informace o semestru a zobrazí stránku `semestr_detail.tpl`.
- `upravit()` - uloží do databáze upravená data o semestru pokud jsou správně zadaná.
- `novy()` - zobrazí stránku `novy_semestr.tpl`.
- `vytvorit()` - zkontroluje zadané informace nového semestru pomocí funkce `validate()`. Pokud jsou správné, vytvoří v databázi nové schéma pro daný semestr. Dále pokud úvazkář vybral možnost nakopírování učitelů, jsou do schématu přidáni učitelé ze schématu semestru, který je nastaven jako předchozí. Nově vytvořený semestr je nastaven jako aktuální.
- `zmenit()` - funkce změní semestr, s jehož daty úvazkář pracuje.
- `parametry()` - načte z databáze parametry semestru pro výpočet ZH a zobrazí stránku `parametry.tpl`.

- `upravitParametry()` - funkce uloží upravené parametry pro výpočet ZH.

2.2.8 UcitelController

Tento controller slouží pro získávání dat o učitelích a jejich úpravu.

- `novy()` - zobrazí stránku `novy_ucitel.tpl`.
- `vytvorit()` - pro nově vytvářeného učitele zkontroluje, zda již není v databázi učitel se stejným uživatelským jménem a zda jsou úvazkářem správně vyplněna data. Poté, pokud je vše v pořádku, uloží nového učitele do databáze.
- `getTeachingSummary($ucitel, $schemaName)` - pro učitele zadaného v `$ucitel` vrací informace o všech předmětech, které učí. Vrátí pole s informacemi o celkovém počtu přednášek a proseminářů předmětu, kolik jich učitel vyučuje a kolik vede cvičení.
- `getTeacherInfo($username, $schemaName)` - ze všech tabulek schématu `$schemaName` v databázi, které se týkají výuky, získá informace o učiteli s uživatelským jménem `$username`. K těmto informacím ještě přidá celé jméno učitele.
- `ucitelRozhrani()` - tato funkce získává informace pro rozhraní učitele. Zavolá pro všechny semestry funkci `getTeacherInfo`. Pokud není učitel v aktuálním semestru, je pro rozhraní zvolen jako aktuální první semestr, ve kterém je učitel nalezen. Dále funkce načte data pro grafy zobrazené v detailu. Pokud není učitel nalezen, je zobrazena stránka `rozhrani_error.tpl`.
- `ucitelRozhraniError()` - zobrazí stránku `rozhrani_error.tpl`.
- `detail($params)` - funkce získá ze vstupního pole `$params` username učitele a načte všechna data o jeho úvazku. Také načte pokrytí předmětů, na kterých se podílí. Tyto data jsou důležité pro detail učitele, který na závěr zobrazí.
- `upravit()` - funkce uloží úpravy provedené na stránce detailu učitele.
- `smazat($params)` - získá username učitele ze vstupního pole `$params` a odstraní jeho data z databáze.
- `smazatUci($params)` - odebere od učitele předmět s kódem, který je předán ve vstupním poli.
- `pridatUci($params)` - přidá k učiteli předmět, jehož kód byl předán prostřednictvím vstupního pole `$params`, a zkontroluje správnost zadaných dat o výuce.

- `seznam()` - načte informace pro stránku se seznamem učitelů a zobrazí stránku `ucitele.tpl`.
- `upravitUci()` - tato funkce slouží k úpravě dat v sekci *Učí* v detailu učitele. Funkce zkontroluje zadané úpravy, uloží je do databáze a aktualizuje informace o pokrytí.
- `parseName(&$teacher)` - rozdělí jméno učitele na jednotlivé složky.
- `vypocetUvazkuZH($username, $semester)` - funkce vrátí pro zadané jméno učitele a semestr hodnotu ZH, kterou získá po zavolání funkce `ucitel_uvazek_zh` z databáze.
- `vypocetUvazku($username, $semester)` - funkce získá z databáze počet ZH a XML rozpis úvazku. Rozpis následně převede na pole pomocí třídy `XMLThing`.
- `getUcitelUvazek($username, $semester)` - funkce vrátí, jaký měl učitel v zadaném semestru úvazek.
- `getLineChartData($username, $semesters)` - funkce vrátí pro každý semestr ze vstupního pole `$semesters` data potřebná pro vytvoření grafu o naplnění úvazku. Za tímto účelem volá funkce `vypocetUvazkuZH` a `getUcitelUvazek`.
- `getPieChartData($username, $semesters)` - tato funkce volá pro všechny semestry ve vstupním poli `$semesters` funkci `vypocetUvazku`. Vrácené hodnoty slouží jako základ pro graf složek úvazku v rozhraní učitele.
- `getFirstValidSemestrCode($username, $semesters)` - funkce vrátí kód prvního semestru z pole `$semesters`, ve kterém je nalezen záznam o učiteli.

2.3 View

Prezentační vrstva je v systému pro podporu tvorby učitelských úvazků realizována pomocí HTML, CSS a JavaScriptu. Jejím prostřednictvím jsou uživateli zobrazena data, se kterými systém pracuje. Pro její implementaci byl využit šablonovací systém *Smarty*[4]. Ten nám umožňuje vytvořit šablonu, která je pro většinu stránek totožná a do ní připravit *bloky*, které budou obsahovat data specifická pro každou stránku. Tyto bloky jsou pak měněny pomocí `Controlleru`, který vybírá akci na základě vstupu od uživatele. Soubory s bloky stránek jsou uložené v adresáři `view`.

Základem pro všechny stránky *administrativního prostředí* je soubor `layout.tpl`. Většina zbylých stránek je pak realizována jediným souborem, který obsahuje

bloky dané stránky. Rozdílem jsou pouze stránky pro import, do kterých se pomocí AJAXu doplňují informace.

Základem prostředí pro běžného uživatele je soubor `rozhrani_layout.tpl` a jeho obsah je doplněn buď ze souboru `rozhrani.tpl`, kde je implementováno zobrazení dat všech semestrů, nebo `rozhrani_error.tpl` podle toho, zda byl učitel nalezen v databázi.

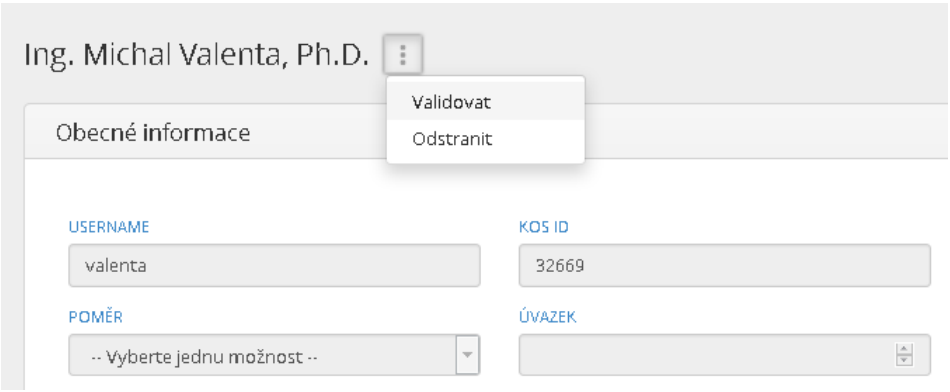
Kapitola 3

Testování

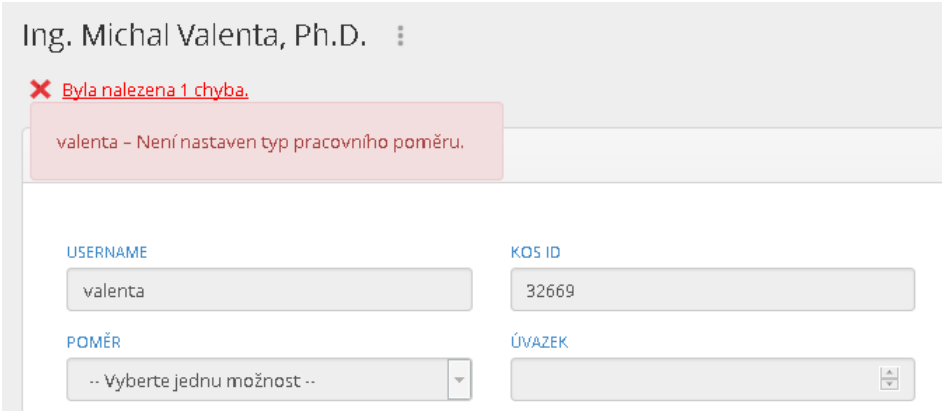
Tato část práce se zabývá testováním dat systému ve dvou základních rovinách. První jsou automatizované testy, které využívají implementované validační funkce, které byly uvedeny v předchozí implementační části. Druhou rovinou je manuální testování nově implementovaných funkcí v rámci této práce.

3.1 Automatizované testy

Pro automatizované testy slouží implementované funkce v modelu a controlleru. Tyto funkce kontrolují správnost zadaných dat uživatelem (viz vytvoření nového semestru) a v reakci na požadavek uživatele zkontroluje data učitelů a předmětů. K tomu slouží funkce v `JsonControlleru`. Tento požadavek může být zadán na stránkách seznamů předmětů a učitelů a v jejich detailech (viz obr. 3.1 a 3.2).



Obrázek 3.1: Možnost validace v detailu učitele.



Obrázek 3.2: Výsledek validace v detailu učitele.

3.2 Manuální testování

Aktuální verze systému byla testována na servru `http://si-projekty-vyvoj.fit.cvut.cz/uvazky-dev/` a na mém lokálním počítači. Při testech byly kontrolovány počty oponentur a vedení ZP při různých časových úsecích. Dále bylo kontrolováno, zda se importují všechny chybějící data a zda nějaký předmět nebo učitel nechybí. Také bylo kontrolováno, zda se při změně údajů o vedených ZP a počtu kreditů za ně správně ukládají upravená data do databáze. Mezi poslední prováděné testy patří, zda se pro přihlášeného úvazkáře načítá správné uživatelské jméno.

Kapitola 4

Datové úložiště

Původní datové úložiště bylo vytvořeno v rámci bakalářské práce bc. Josefa Němečka[1]. Toto úložiště od té doby však prošlo několika změnami. V této části práce bude popsán současný stav datového úložiště.

Úložiště je v systému realizováno pomocí PostgreSQL databáze. Tato databáze je rozčleněna do několika schémat, z nichž jedno má název *funkce* a ostatní mají jméno ve formě *uvazky_kodSemestru*. Schémata *uvazky_kodSemestru* v sobě uchovávají informace o vytížení učitelů v semestru, kterému odpovídá kód v názvu. Schéma *funkce* pak v sobě uchovává obecné informace o semestrech a informace, které jsou pro všechny semestry společné.

4.1 Schéma uvazky

Toto schéma se skládá z 11 tabulek, které v sobě uchovávají data za daný semestr. Tyto data slouží jako základ pro výpočet ZH.

Tabulka *ucitel* obsahuje informace o učiteli, o jeho úvazku, jménu a dalších osobních údajích. V *predmet* jsou uložené základní informace o předmětech vyučovaných v daném semestru. Tabulky *prednasky* a *proseminare* v sobě uchovávají seznam vypsaných paralelek přednášek a poseminářů jednotlivých předmětů. K těmto paralelkám je pak přiřazen učitel v tabulkách *uci_prednasky* a *uci_proseminare*. V tabulce *uci* jsou uvedené zbylé informace o podílu učitele na předmětu (počet cvičení, projektů, atd.).

Zbylé tabulky *komise, ma_doktorandy, oponentura* a *reprezentuje* se již netýkají výuky předmětů. *Komise* v sobě uchovává informace, v jakých komisích učitel byl a na jaké pozici v nich učitel vystupoval. Tabulka *ma_doktorandy* je přehledem o tom, kolik doktorandů a v jakém ročníku učitel má a jak se účastnil oborové rady. Tabulka *oponentura*, která má v sobě data o účasti učitele na ZP, je jedinou, která není totožná se svou předlohou v původním návrhu, protože do ní byly přidány sloupce pro vedení ZP. Poslední tabulka *reprezentuje* obsahuje informace o ostatních aktivitách učitele, jako jsou účast na akcích v Betlémské kapli nebo Den otevřených dveří.

4.2 Schéma funkce

Ve schématu *funkce* jsou uložené informace, které jsou pro všechny semestry společné. Informace zde se dají rozdělit do dvou základních kategorií, jimiž jsou tabulky, které obsahují obecné informace o semestrech a účtech úvazkářů, a funkce, které slouží k výpočtu ZH a předání těchto informací datové vrstvě systému.

4.2.1 Uložené tabulky

Ve schématu *funkce* jsou uloženy 4 základní tabulky. Těmi jsou *admins, parametry, predmet_vahy* a *semestr_info*. Tabulky *admins* a *parametry* byly vyčleněny do tohoto schématu oproti původnímu návrhu, kde byly obsaženy v každém schématu. Tabulka *admins* obsahuje informace o účtech úvazkářů (název, heslo, katedru a nově i uživatelské jméno) a tabulka *parametry* parametry pro každý semestr, které jsou nutné k výpočtu ZH.

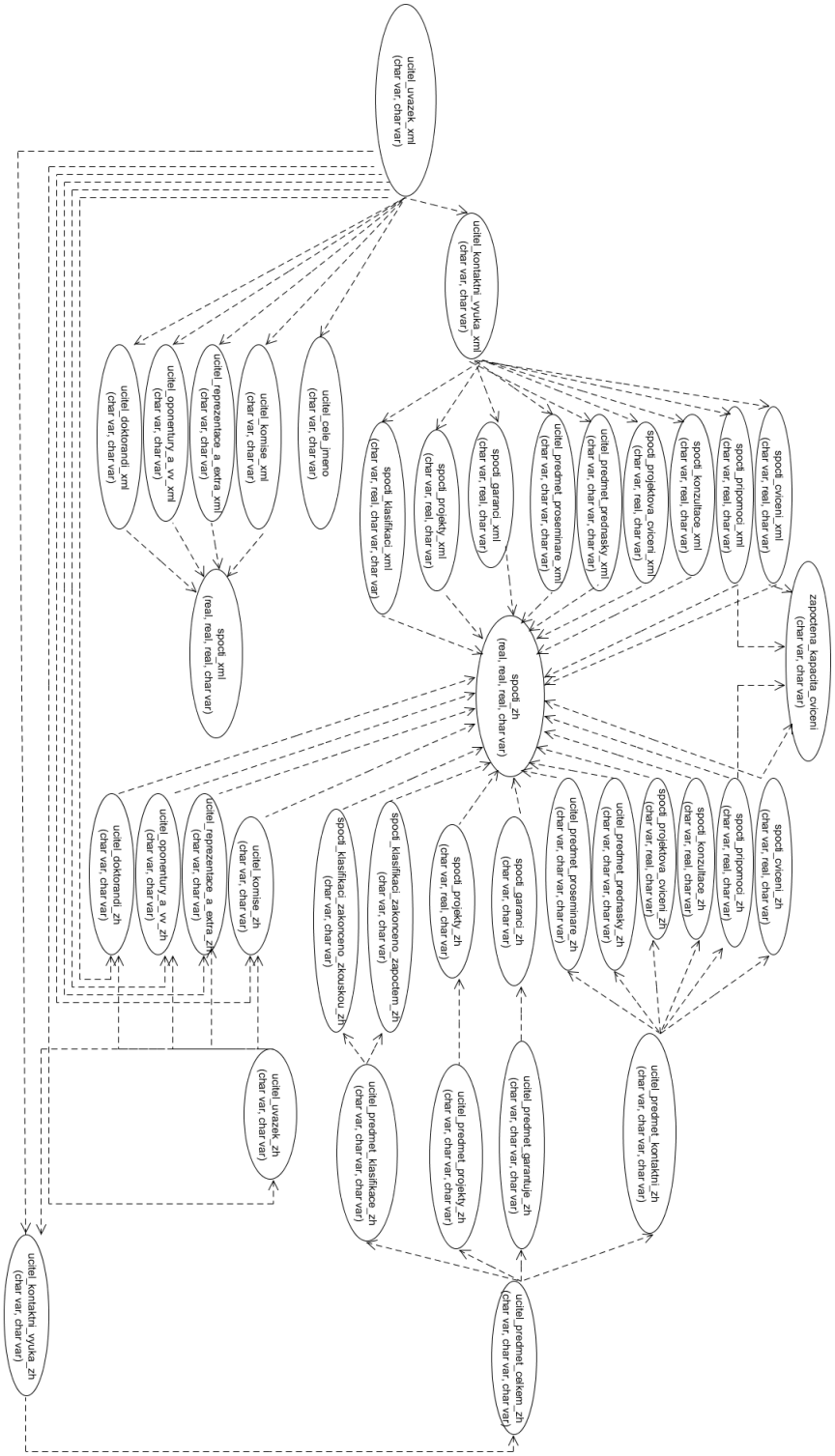
Zcela nové tabulky oproti původnímu návrhu jsou *predmet_vahy* a *semestr_info*. V *predmet_vahy* jsou uvedeny pro všechny předměty hodnoty, které udávají, jakou měrou ovlivňují výsledky v semestru a výsledky ve zkoušce celkovou známku. Tabulka *semestr_info* má v sobě pak informace o jednotlivých semestrech, které jsou v databázi uložené.

4.2.2 Uložené funkce

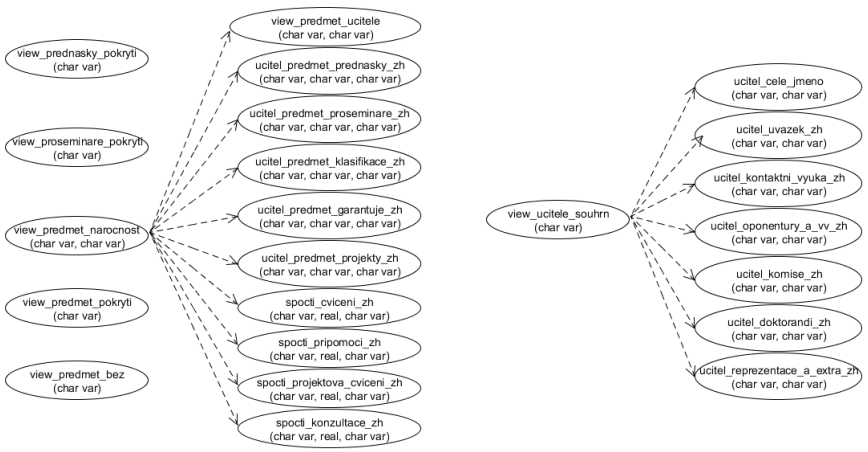
Druhou podstatnou částí tohoto schématu jsou PL/pgSQL funkce, které slouží pro výpočet ZH. Zde je ke každé uveden její význam a můžeme si prohlédnout i mapu jejich vzájemného volání. Tyto funkce, lze rozdělit do 2 skupin, na základě množství informací, které zprostředkovávají, na funkce pro jednoho učitele (viz obr. 4.1) a souhrnné funkce (viz obr. 4.2). Funkce pro jednoho učitele pak lze ještě dále rozdělit podle formy jejich výstupu na *funkce_xml*, které vrací informace ve formě XML, a *funkce_zh*, které vrací pouze číselnou hodnotu odpovídající počtu ZH.

4.2.2.1 Funkce pro úvazek jednotlivce

- `spocti_zh(k1, k2, v, m)` - funkce na základě koeficientů `k1` a `k2` vrátí odpovídající hodnotu ZH.
- `spocti_xml(k1, k2, v, mes)` - funkce spočítá z koeficientů ZH a vrátí XML text obsahující popisek `mes`, vzorec výpočtu a hodnotu ZH.
- `zapoctena_kapacita_cviceni(pkod, psemestr)` - vrátí počet studentů cvičení předmětu `pkod`.
- `spocti_cviceni_xml(pkod, phodin, psemestr)` - funkce získá typ předmětu a počet studentů jednoho cvičení. Pokud není nastaven typ, vrátí



Obrázek 4.1: Mapa vzájemného volání funkcí ve schématu *funkce*.



Obrázek 4.2: Mapa volání souhrnných funkcí ve schématu *funkce*.

se chyba. V opačném případě se načtou parametry pro daný typ cvičení z tabulky parametry a je spočítána hodnota ZH za cvičení. Funkce vrátí XML rozkrytí výpočtu obsahující popisek, vzorec výpočtu a hodnotu ZH za cvičení.

- `spocti_cviceni_zh(pkod, phodin, psemestr)` - funkce načte data o typu předmětu a počtu studentů cvičení. Pokud není nastaven typ, vrátí 0. Pro typ následně načte parametry pro výpočet ZH, spočte ZH a tuto hodnotu vrátí.

- `spocti_garanci_xml(pkod, ppodil, psemestr)` - funkce načte počet kreditů předmětu `pkod` a počet jeho studentů, pak načte parametry pro výpočet garance a vypočte celkovou hodnotu ZH za garanci předmětu. Funkce vrátí XML rozkrytí počtu ZH za garanci předmětu odpovídající podílu na garanci `ppodil`.

- `spocti_garanci_zh(pkod, psemestr)` - funkce vrátí počet ZH za garanci předmětu `pkod`.

- `spocti_klasifikaci_xml(pkod, ppodil, ptyp, psemestr)` - získá informace o počtu studentů, počtu kreditů, váhách a formě zakončení předmětu `pkod`. Podle formy zakončení načte parametry pro výpočet ZH a vypočítá počet ZH za klasifikaci. Funkce vrátí XML rozpis výpočtu ZH za klasifikaci.

- `spocti_klasifikaci_zakonceno_zapoctem_zh(pkod, psemestr)` - funkce načte parametry pro výpočet ZH za klasifikaci předmětu ukončeného klasifikovaným zápočtem, počet studentů předmětu a počet jeho kreditů. Z těchto dat vypočte celkový počet ZH za klasifikaci předmětu a tu vrátí.

- `spocti_klasifikaci_zakonceno_zkouskou_zh(pkod, psemestr)` - funkce načte parametry pro výpočet ZH, počet studentů a kreditů předmětu a vrátí celkový počet ZH za klasifikaci předmětu.

- `spocti_konzultace_xml(pkod, phodin, psemestr)` - načte počet studentů a typ předmětu `pkod`, pokud není nastaven typ, vrátí chybu. Dále načte parametry pro výpočet ZH za konzultace a vypočte počet ZH odpovídající počtu `phodin` konzultací. Vrátí XML rozpis tohoto výpočtu.

- `spocti_konzultace_zh(pkod, phodin, psemestr)` - načte počet studentů a typ předmětu, parametry pro výpočet ZH za konzultace a vrátí počet ZH odpovídající počtu hodin konzultací `phodin`.

- `spocti_pripomoci_xml(pkod, phodin, psemestr)` - načte typ předmětu a počet studentů jeho cvičení, načte parametry pro výpočet ZH za přípomoci, vypočte počet ZH odpovídající počtu hodin přípomocí `phodin` a vrátí XML rozpis tohoto výpočtu.

- `spocti_pripomoci_zh(pkod, phodin, psemestr)` - načte typ předmětu a počet studentů cvičení, načte parametry pro výpočet ZH za přípomoci daného typu předmětu a vrátí počet ZH odpovídající počtu přípomocí `phodin`.

- `spocti_projektova_cviceni_xml(pkod, phodin, psemestr)` - načte počet studentů projektu a typ předmětu. Pokud je nastaven typ předmětu, načte parametry pro výpočet ZH, vypočítá počet ZH odpovídající počtu hodin cvičení `phodin` a vrátí XML rozpis tohoto výpočtu.

- `spocti_projektova_cviceni_zh(pkod, phodin, psemestr)` - načte počet studentů projektu a typ předmětu, načte parametry pro výpočet ZH a vrátí hodnotu ZH odpovídající počtu hodin projektových cvičení `phodin`.

- `spocti_projekty_xml(pkod, projektu, psemestr)` - načte počet kreditů za předmět `pkod`, načte parametry pro výpočet ZH za projekt a vypočítá počet ZH odpovídající počtu vedených projektů `pprojektu`. Vrátí XML rozpis tohoto výpočtu.

- `spocti_projekty_zh(pkod, pprojektu, psemestr)` - načte počet kreditů za předmět, parametry pro výpočet ZH za projekt a vrátí hodnotu ZH odpovídající počtu projektů `pprojektu`.

- `ucitel_cele_jmeno(puname, psemestr)` - načte informace o učiteli s uživatelským jménem `puname` a pokud ho najde v databázi, spojí jeho příjmení, jméno a tituly do jednoho řetězce.

- `ucitel_doktorandi_xml(puname, psemestr)` - načte parametry pro výpočet ZH za práci s doktorandy, načte data o učiteli `puname` z tabulky *ma_doktorandy* v semestru `psemestr` a pro jednotlivé sloupce volá funkci `spocti_xml` s odpovídajícími parametry a návratové hodnoty si ukládá. Výstupem funkce je XML rozpis práce s doktorandy.

- `ucitel_doktorandi_zh(puname, psemestr)` - načte parametry pro výpočet ZH za práci s doktorandy, načte data o učiteli `puname` z tabulky *ma_doktorandy* semestru `psemestr` a pro jednotlivé sloupce volá funkci `spocti_zh`. Výsledné hodnoty ZH sčítá a celkový součet vrátí.

- `ucitel_komise_xml(puname, psemestr)` - načte parametry pro výpočet ZH za účast v komisích a načte data o učiteli `puname` z tabulky *komise* v semestru `psemestr`. Pro jednotlivé sloupce pak volá funkci `spocti_xml` a výsledky ukládá. Funkce vrátí celkový XML rozpis o účasti učitele v komisích.

- `ucitel_komise_zh(puname, psemestr)` - funkce načte parametry pro výpočet ZH za účast v komisích, načte informace o učiteli z tabulky *komise* a sečte hodnoty ZH za jednotlivá působení v komisích, které získá pomocí funkce `spocti_zh`. Funkce vrátí tento celkový součet ZH.

- `ucitel_kontaktni_vyuka_xml(puname, psemestr)` - načte kód všech předmětů, které učitel `puname` učí z tabulek *uci, uci_prednasky* a *uci_proseminare* semestru `psemestr`. Pro každý předmět pak zjistí, jestli vyučuje přednášky a proseináře. Pokud ano, zavolá funkci `ucitel_predmet_prednasky_xml` nebo `ucitel_predmet_proseminare_xml` a jejich výstup přidá k výstupním informacím. Následně zkontroluje informace o učiteli a předmětu v tabulce *uci* a na základě nalezených hodnot volá funkce `spocti_garanci_xml, spocti_cviceni_xml, spocti_pripomoci_xml, spocti_projektova_cviceni_xml, spocti_projekty_xml, spocti_konzultace_xml` a `spocti_klasifikaci_xml`. Poslední jmenovanou volá dvakrát pro výpočet ZH za klasifikaci v semestru a při zkouškách. Funkce vrátí kompletní XML rozpis za kontaktní výuku.

- `ucitel_kontaktni_vyuka_zh(puname, psemestr)` - funkce načte seznam kódů, které učitel vyučuje, a pro každý zavolá funkci `ucitel_predmet_celkem_zh` a vrácené hodnoty sečte. Výsledkem funkce je počet ZH za všechny předměty učitele.

- `ucitel_oponentury_a_vv_xml(puname, psemestr)` - načte parametry pro výpočet ZH za oponování a vedení ZP, načte o učiteli `uname` data z tabulky *oponentura* v semestru `psemestr` a pro jednotlivé sloupce volá funkci `spocti_xml` s příslušnými parametry. Vrátí kompletní XML rozpis o oponování a vedení ZP.

- `ucitel_oponentury_a_vv_zh(puname, psemestr)` - funkce načte parametry pro výpočet ZH, data o učiteli v tabulce *oponentura* a pro každý sloupec zavolá funkci `spocti_zh`. Sečte ZH za jednotlivé položky a součet vrátí.
- `ucitel_predmet_celkem_zh(puname, pkod, psemestr)` - volá pro předmět `pkod` funkce `ucitel_predmet_kontaktni_zh, ucitel_predmet_klasifikace_zh, ucitel_predmet_garantuje_zh` a `ucitel_predmet_projekty_zh`. Jejich výstupy sečte a výsledek vrátí.
- `ucitel_predmet_garantuje_zh(puname, pkod, psemestr)` - načte z tabulky *uci* semestru `psemestr` informace o učiteli `puname` a předmětu `pkod`. Pokud se podílí na organizaci předmětu, zavolá funkci `spocti_garanci_zh`. Výsledkem je počet ZH odpovídající podílu na organizaci předmětu.
- `ucitel_predmet_klasifikace_zh(puname, pkod, psemestr)` - načte data o učiteli a předmětu z tabulky *uci*, pokud data nalezne, načte informace o váhách a typu zakončení předmětu. Podle typu zakončení pak volá funkci `spocti_klasifikaci_zakonceno_zkouskou_zh` nebo `spocti_klasifikaci_zakonceno_zapoctem_zh`. Spočítá pro učitele ZH za klasifikaci v semestru a ve zkouškovém období a vrátí celkový počet ZH učitele za klasifikaci předmětu.
- `ucitel_predmet_kontaktni_zh(puname, pkod, psemestr)` - zavolá funkce `ucitel_predmet_prednasky_zh` a `ucitel_predmet_proseminare_zh` pro výpočet ZH za přednášky a proseminář předmětu, které učitel učí. Následně získá informace o předmětu a učiteli z tabulky *uci* a na základě uložených hodnot volá funkce `spocti_cviceni_zh, spocti_pripomoci_zh, spocti_projektova_cviceni_zh` a `spocti_konzultace_zh`. Součet všech ZH vrátí.
- `ucitel_predmet_prednasky_xml(puname, pkod, psemestr)` - načte typ předmětu a pokud je uveden, načte i parametry pro výpočet ZH za přednášky. Následně načte přednášky předmětu, které učitel učí, a pro každou zjistí její počet studentů a vypočte pro ní počet ZH. Funkce vrátí XML rozpis paralelek přednášek předmětu, které učitel vyučuje, s podrobnostmi výpočtu.
- `ucitel_predmet_prednasky_zh(puname, pkod, psemestr)` - načte typ předmětu a parametry pro výpočet ZH, načte učitelovy přednášky, pro každou vypočítá počet ZH a vrátí celkový počet ZH.
- `ucitel_predmet_projekty_zh(puname, pkod, psemestr)` - načte pro předmět `pkod` a učitele `puname` data z tabulky *uci* semestru `psemestr` a pokud má nějaké projekty, zavolá funkci `spocti_projekty_zh`. Vrátí počet ZH za vedené projekty.

- `ucitel_predmet_proseminare_xml(puname, pkod, psemestr)` - načte typ předmětu a pro něj parametry pro výpočet ZH. Pak načte všechny proseminář předmětu, které učitel učí, a pro ně volá funkci `spocti_zh`. Na závěr vrátí kompletní XML rozpis paralelek proseminářů s popisem výpočtu jejich ZH.

- `ucitel_predmet_proseminare_zh()` - načte typ předmětu, parametry pro výpočet ZH a paralelky proseminářů učitele. Pro jednotlivé paralelky vypočítá ZH pomocí funkce `spocti_zh` a vrátí celkový počet ZH.

- `ucitel_reprezentace_a_extra_xml(puname, psemestr)` - načte parametry pro výpočet ZH za reprezentaci, načte učitele z tabulky *reprezentuje* semestru `psemestr` a pro jednotlivé sloupce volá funkci `spocti_xml` s příslušnými parametry. Výstupem funkce je podrobný XML rozpis s výpočtem ZH reprezentace učitele.

- `ucitel_repretentace_a_extra_zh(puname, psemestr)` - funkce načte parametry pro výpočet ZH a data z tabulky *reprezentuje*. Následně pro každý sloupec zavolá funkci `spocti_zh` a vrátí součet výstupů těchto funkcí.

- `ucitel_uvazek_xml(puname, psemestr)` - načte kód a název semestru, zkontroluje, zda učitel existuje v databázi a zda jsou pro semestr nastaveny parametry pro výpočet ZH. Následně volá funkce pro jednotlivé položky úvazku (viz obr. 4.1) a z jejich výstupu vytvoří kompletní rozpis úvazku učitele.

- `ucitel_uvatek_zh(puname, psemestr)` - sečte výstup funkcí `ucitel_kontaktni_vyuka_zh`, `ucitel_komise_zh`, `ucitel_oponentury_a_vv_zh`, `ucitel_doktrorandi_zh` a `ucitel_reprezentace_a_extra_zh`. Tento součet je výstupem funkce.

4.2.2.2 Souhrnné funkce

Tyto funkce byly do schématu doplněny jako náhrada za pohledy nad databází. Každá z těchto funkcí má prefix *view_*. Tyto funkce jsou:

- `view_predmet_narocnost(pkod, psemestr)` - funkce vrací pro předmět `pkod` v semestru `psemestr` celkový počet ZH i s rozpisem na jednotlivé jeho části, které byly za předmět uděleny.

- `view_predmet_pokryti(psemestr)` - výstupem je tabulka s kompletními informacemi o pokrytí předmětů v semestru `psemestr`. Je v ní kód předmětu, katedra, podíl semestru a zkoušky na známce a ke každé složce

předmětu (přednášky, proseminář, atd.) údaje o vypsaném počtu, pokrytém počtu a počtu, který zbývá pokrýt. Ve výstupu jsou pouze informace o předmětech, které jsou v tabulkách *uci, uci_prednasky* a *uci_proseminare*.

- `view_predmet_bez(psemestr)` - jedná se o doplněk k předcházející funkci. Výstupem je tabulka s kompletními informacemi o pokrytí předmětů v semestru `psemestr`, která obsahuje data o předmětech, které nejsou v tabulkách *uci, uci_prednasky* a *uci_proseminare*.
- `view_predmet_ucitele(pkod, psemestr)` - návratovou hodnotou této funkce je tabulka se seznamem učitelů, kteří se podíleli na předmětu s kódem `pkod` v semestru `psemestr` a údaji o tom, jak se podíleli (vedli cvičení, přednášky, proseminář, atd.).
- `view_prednasky_pokryti(psemestr)` - tato funkce vrátí tabulku s informacemi o všech přednáškách v semestru `psemestr`. Ta obsahuje kód předmětu, paralelku přednášky, počet studentů přednášky, počet hodin za semestr, kolik hodin je pokryto a kolik hodin zbývá pokrýt.
- `view_proseminare_pokryti(psemestr)` - tato funkce vrací stejné informace o proseminářích v semestru `psemestr`, jako byly informace o přednáškách v předcházející funkci.
- `view_ucitele_souhrn(psemestr)` - výstupem této funkce je tabulka, která obsahuje všechny učitele, pro které platí, že v semestru `psemestr` není výše jejich ZH rovna 0. Tabulka obsahuje jejich základní informace a jednotlivé složky jejich úvazku.

Kapitola 5

Možnosti dalšího rozšíření

V průběhu práce nebyly splněny všechny požadavky a byly nalezeny i další možnosti, které by se daly využít pro další rozvoj systému a jeho vlastností.

5.1 UsermapApi

Při analýze importu dat a návrhu možných úprav v rámci této práce byla zvažována možnost využití zdroje *Usermap API*[5]. Tento zdroj byl zvažován, protože poskytuje mnohem rozsáhlejší informace o osobách než KOSapi. Mezi tyto role patří zejména tzv. *celoškolské* a *fakultně specifické business role* z fakultního IDM. Tyto role by se dali při importu využít pro získání informací o pracovním poměru učitele (plný úvazek, částečný úvazek, atd.), které nám KOSapi neposkytuje a musí je proto úvazkáři doplňovat do systému manuálně.

Bohužel se nepodařilo zprovoznit autorizaci, a proto byla tato možnost přesunuta do možného dalšího vývoje systému.

5.2 Shibboleth

V posledních letech začal být na FIT využíván přihlašovací systém *Shibboleth*. Ten umožňuje svým uživatelům využívat přístup téměř ke všem fakultním systémům prostřednictvím jednoho přihlášení a uživatelé tak nemusí pro každý systém znovu zadávat své uživatelské jméno a heslo.

Jedním z úkolů této práce, který nebyl splněn, bylo i zprovoznění této podpory přihlašování. Důvodem je, že *Shibboleth* se musí specificky nastavit na výsledné platformě, na které bude systém pro podporu tvorby učitelských úvazků spuštěn. Nastavovat ho nyní by tedy nemělo žádný smysl.

Pro budoucí realizaci bych jen doporučil, aby byla tato možnost realizována pomocí dodatečného tlačítka na přihlašovací stránce, které bude tuto možnost nabízet. Touto formou mohou zůstat zachovány i současné úvazkářské účty bez jakýchkoliv změn.

Závěr

Zadáním této práce bylo seznámit se se současnou verzí systému, návrhově a implementačně zpracovat požadované funkcionality a zdokumentovat celý systém spolu s revizí jeho kódu.

Tato bakalářská práce, byla zaměřena především na první část třetího bodu zadání, jímž je dokumentace systému. Tato část byla obsáhle rozebrána v předchozích kapitolách a myslím, že k této části není třeba mnoho dodávat.

Bohužel však z nedostatku času nebyly splněny všechny požadavky druhého bodu zadání a revize kódu. Místo některých požadavků byly doplněny funkcionality, které byly považovány v současné době za důležitější. Možnost přihlášení prostřednictvím *Shibboleth* byla odložena z dříve uvedených důvodů a bylo pro ni pouze navrženo, jakým způsobem, by měla být realizována.

Tato práce pro mě byla velice zajímavou. Hlavním důvodem je, že s mnohými technologiemi, které jsou ve zpracovávaném systému použity, jsem se setkal poprvé.

Literatura

[1] Němeček, J.: *Systém pro hodnocení pedagogického výkonu verze 3.0: metodika, datové struktury a datové zdroje*. 2014.

[2] Štefan, J.: *Systém pro hodnocení pedagogického výkonu verze 3.0: rozhraní pro úvazkáře a učitele*. 2014.

[3] Jirůtka, J.: KOSapi: Hlavní stránka. 2016. Dostupné z: `https://kosapi.fit.cvut.cz/projects/kosapi/wiki`

[4] Group, S. D.: Smarty template engine: Hlavní stránka. 2016. Dostupné z: `http://www.smarty.net/`

[5] Jirůtka, J.: Usermap API: Hlavní stránka. 2016. Dostupné z: `https://rozvoj.fit.cvut.cz/Main/usermap-apii`

Příloha **A**

Seznam použitých zkratek

ČVUT České vysoké učení technické

FIT Fakulta informačních technologií

HTML HyperText Markup Language

MVC Model-View-Controller

PHP Hypertext Preprocessor

PL/pgSQL PostgreSQL Procedural Language

SQL Structured Query Language

XML Extensible markup language

ZH Započitatelné hodiny

ZH Závěrečné práce

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF