Sem vložte zadání Vaší práce.

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

Bakalářská práce

Optimalizace hodnocení a automatické kontroly semestrální práce v předmětu Databázové systémy

Jan Sýkora

Vedoucí práce: Ing. Jiří Hunka

6. května 2015

Poděkování

Mé poděkování patří panu Ing. Jiřímu Hunkovi za odborné vedení, cenné rady a vřelý přístup při zpracování této práce. Dále bych chtěl poděkovat všem vyučujícím předmětu Databázové systémy na FIT ČVUT, kteří se zúčastnili uživatelského testování, jmenovitě panu Mgr. Ondřeji Dvořákovi, Ing. Ivanu Halaškovi, Ing. Pavlu Krejčímu, Ing. Josefu Pavlíčkovi, Ph.D., Ing. Michalu Valentovi, Ph.D. a ještě jednou panu Ing. Jiřímu Hunkovi.

Také bych chtěl poděkovat za spolupráci jak členům mého původního týmu, se kterými jsem začal realizovat celou myšlenku podpory předmětu Databázové systémy, tak členům nového týmu, který jsem vedl, a který pokračoval v práci předchozího týmu. V původním týmu to byli Miroslav Brabenec, kterému bych chtěl obzvlášt poděkovat za jeho součinnost a nápomoc, Tomáš Brůna, Tomáš Kasalický, Petr Jirásko a Michal Pustějovský. Za nový tým bych chtěl poděkovat Oldřichovu Malcovi, Lukáši Vyčítalovi a Nguyenu Van Nhanovi. V neposlední řadě bych ještě rád směroval díky Bc. Zdeňku Pekáčkovi za jeho věcné rady při vývoji systému.

Hlavně bych chtěl ale ze srdce poděkovat své rodině za veškerou podporu při vytváření této práce, děkuji za to, že jste tu byli vždy pro mne.

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. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 6. května 2015 .....

Odkaz na tuto práci

Sýkora, Jan. *Optimalizace hodnocení a automatické kontroly semestrální práce v předmětu Databázové systémy.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

Abstrakt

Práce se zaměřuje na automatickou kontrolu a optimalizaci opravy semestrální práce v předmětu Databázové systémy, vyučovaného na Fakultě informačních technologií ČVUT v Praze. Cílem práce bylo analyzovat možnosti oprav semestrální práce v předmětu Databázové systémy a rozšířit připravovaný nástroj na automatickou kontrolu semestrální práce tak, aby maximálně urychlil a ulehčil celkový proces opravy. Výsledný nástroj byl uživatelsky otestován a výsledky byly zohledněny ve finální aplikaci.

Klíčová slova Databázové systémy, automatická oprava, uživatelské rozhraní, relační algebra, SQL, XML, PHP, použitelnost

Abstract

The thesis is focused on automatic check and optimalization of evaluation of semestral project in the course Database systems which is taught in Faculty of Information Technology at CTU in Prague. The purpose of this thesis was to analyze possibilities of checks of semestral project in course Database systems and extend the upcoming tool for automatic check of semestral project to ease the whole process of check. The created tool went through the usability testing and the results were taken into account in the final version of this tool.

Keywords Database systems, automatic check, user interface, relational algebra, SQL, XML, PHP, usability

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Semestrální práce v předmětu Databázové systémy (DBS) slouží k prověření znalostí z návrhu databáze, relační algebry a dotazovacího jazyka SQL. Úkolem této semestrální práce je, aby student vymyslel situaci z praxe, kde by měla být nasazena databáze, navrhl ji, vytvořil, naplnil daty a prováděl nad ní dotazy v relační algebře a v jazyku SQL. Opravování a hodnocení je prováděno cvičícím ve třech kontrolních bodech (iteracích) během semestru na příslušných cvičeních.

V současné době student průběžně vypracovává semestrální práci do předpřipraveného XML souboru, který nahrává společně s dalšími soubory (obrázky schémat, SQL skripty, tabulky s výsledky SQL dotazů) do webového adresáře. Vyučující během semestru musí provést při každém kontrolním bodu ruční opravu požadované části semestrální práce u všech svých studentů. Tento proces je velice neefektivní, protože vyučující musí zkontrolovat i ty části, které by se daly kontrolovat automaticky, nebo které by se daly alespoň částečně předpřipravit k opravě. A vzhledem k vysokému počtu studentů připadajícímu na jednoho vyučujícího zabere oprava všech odevzdaných prací spoustu času.

Problémem s opravou semestrálních prací se již v minulosti na Českém Vysokém Učení Technickém zabývaly některé práce, ale žádné z nich nedosáhly takového výsledku, aby byly aktivně používány všemi vyučujícími při opravě. I proto se v rámci školního týmového projektu (Softwarový týmový projekt 1 a Softwarový týmový projekt 2), vedeným Ing. Jiřím Hunkou, začal vyvíjet nový informační systém na podporu předmětu Databázové systémy, který měl jako jeden z cílů usnadnit opravování semestrálních prací.

Cílem bakalářské práce je analyzovat současný stav opravování semestrálních prací a nástrojů využívaných k jejich opravě a na základě získaných požadavků, od vyučujících předmětu Databázové systémy, navrhnout a implementovat nástroj na automatickou kontrolu semestrálních prací, který zefektivní jejich opravu.

Kapitola 1

Analýza semestrální práce a aktuální možnosti její automatické opravy

K tomu, aby student úspěšně absolvoval předmět Databázové systémy, musí samostatně vypracovat semestrální práci (dále SP), jejímž cílem je navrhnout a vytvořit datové úložiště pro nějakou smysluplnou aplikaci v relační SQL databázi.[2]

Semestrální práce se vypracovává do XML souboru a obsahuje takové části, které umožňují její automatizovanou kontrolu. Cílem mé bakalářské práce je vytvoření nástroje, který bude schopný provádět tuto automatickou kontrolu. Ale před tím, než jsem se pustil do vývoje tohoto nástroje, tak jsem se musel seznámit se samotnou semestrální prací a s aktuálním stavem její opravy.

1.1 Struktura semestrální práce

Student musí v průběhu semestru vytvořit tyto části semestrální práce:[1]

- Slovní zadání (popis modelované domény)
- Návrh struktury datového úložiště
  - Konceptuální model
  - Logický (fyzický) model
- Implementace datového úložiště
  - Skript na vytvoření databáze

[1] Příklad kompletní ukázkové semestrální práce naleznete v přiloženém CD nebo na adrese `http://users.fit.cvut.cz/~valenta/BI-DBS/semestralka/ukazka/`

  - Skript na naplnění databáze testovacími daty
  - Dotazy nad vytvořenou databází formulované v relační algebře a v SQL

- Závěr, seznam použité literatury

m

1.1.1 Slovní zadání

Student si nejprve vymyslí slovní zadání, popis modelované domény. Slovní zadání by mělo být dlouhé přibližně 300 slov a musí být natolik rozsáhlé, aby se na něm dala demonstrovat celá sada SQL příkazů. Příkladem může být informační systém školy, knihovny nebo nemocnice.

1.1.2 Návrh struktury datového úložiště

V dalším kroku student provede návrh struktury datového úložiště ve zvolené grafické notaci. Grafickou notací se rozumí **konceptuální model** nebo **logický (fyzický) model**. Konceptuální model zachycuje realitu pomocí objektů a vazeb mezi nimi. Příkladem takového modelu je ER[2] model nebo UML[3] Class Diagram. Konceptuální model se vyznačuje svojí univerzálností a jednoduchým zachycením reality. Není závislý na použitých technologiích a je lehce pochopitelný. Naproti tomu logický model se vždy vztahuje ke konkrétní databázi.

Vytvoření konceptuálního modelu a diskuse smyček v něm je povinné. Student musí také dodat zdrojový soubor konceptuálního modelu (ne pouze obrázek), který lze vygenerovat pomocí nástroje na vytváření modelu. Vytvoření fyzického modelu povinné není, ale pro úspěšné vytvoření databáze je téměř nezbytný.

Student má od školy k dispozici nástroj na modelování databázových modelů (Oracle SQL Developer), který se používá při výuce, nicméně si může vybrat i jiný nástroj.

1.1.3 Implementace datového úložiště, Dotazy

V posledním kroku má student za úkol implementovat datové úložiště na konkrétním databázovém stroji. Student si může vybrat, jaký RDBMS[4] zvolí k implementaci, automaticky má podporu pro databázi Oracle, ta ale není povinná.

[2] Entity-relationship

[3] Unified Modeling Language

[4] Relational Database Management System - Systém řízení báze dat, který má data uložena v relačních tabulkách

Student tedy podle návrhu, který zhotovil v předešlé části semestrální práce, vypracuje SQL skripty na vytvoření tabulek a vztahů mezi nimi a naplní vytvořenou databázi testovacími daty. Nad touto databází musí dále provést SQL dotazy a k některým z SQL dotazů formulovat ekvivalentní dotazy v relační algebře. U dotazů navíc musí pokrýt všechny kategorie SQL příkazů podle příslušné tabulky pokrytí.

1.1.4 Závěr semestrální práce

Nakonec student musí dodat seznam literatury, ze které vycházel, a sepíše závěr, ve kterém shrne jím vytvořenou semestrální práci. Výsledkem studentova snažení by mělo být pochopení celého cyklu návrhu a vývoje relačního databázového systému a seznámení se s některými databázovými technologiemi.

1.2 Hodnocení semestrální práce

Průběžné výsledky SP se kontrolují třikrát během semetru na cvičeních - tedy ve třech iteracích.

U semestrální práce se kromě správnosti návrhu databáze a její implementace kontrolují také formální náležitosti dokumentace, jako jsou například údaje autora, prohlášení o samostatném vypracování, závěr, či použitá literatura a zdroje. Avšak největší důraz při opravování je kladen právě na tři dílčí části - slovní zadání, datové modely a dotazy.

Způsob bodování probíhá tak, že student dostane na začátku přidělený určitý počet bodů a od tohoto počtu bodů se mu v průběhu semestru při každé iteraci strhávají body za chyby v jeho práci či za pozdní odevzdání. Student má ale také možnost získat bonusové body za časné odevzdání, či za originální řešení.

Pro úspěšné splnění semestrální práce musí mít student po posledním odevzdání (tj. po odevzdání do 3. iterace) předem stanovený minimální počet bodů. Přehled jednotlivých kontrolovaných částí a bodování naleznete v tabulce 1.1.

1.2.1 1. iterace

V první iteraci se hodnotí slovní zadání modelované domény a student také musí vytvořit alespoň tři dotazy formulované v přirozeném jazyce (tedy pouze slovní popis dotazu, například: *Vyber všechny zelené lodě, které byly na plavbě s průvodcem*).

1.2.2 2. iterace

Ve druhé iteraci se kontroluje 10 dotazů v přirozeném jazyce, přičemž tyto dotazy musí pokrýt kategorie dotazů C a D (tabulka kategorií viz příloha

	Co se kontroluje	**Maximální počet stržených bodů**
1. iterace	Popis 3 dotazy v přirozeném jazyce	5
2. iterace	Konceptuální model Diskuze smyček a zdrojový soubor Logický model (nepovinný) 10 dotazů v přirozeném jazyce Pokrytí kategorií C a D	5
3. iterace	Create skript (vytvoření databáze) Insert skript (naplnění databáze daty) 25 dotazů (25 SQL, 10 relační algebra) Pokrytí všech kategorií Závěr Použitá literatura	10
Student začíná s **20 body**. Minimální počet bodů nutný ke splnění je **10 bodů**. Je možnost získat i bonusové body - určuje si každý vyučující podle sebe.		

Tabulka 1.1: Souhrn hodnocení semestrální práce

B.1) a dále se kontroluje konceptuální model a nepovinně logický (fyzický) model.

Učitel kontroluje, jestli modely odpovídají slovnímu zadání, obsahují alespoň 10 entit a slovně vyjádřená integritní omezení, a jestli obsahují diskuzi smyček, tzn. v případě, že model obsahuje smyčku, zdůvodnit, proč ji obsahuje, a jestli je opravdu nutná.

U modelu se také kontroluje přítomnost všech typů kardinality vztahu[5] a parciality vztahu[6] mezi entitami.

1.2.3 3. iterace

Ve třetí iteraci se kontroluje správné vytvoření databáze pomocí SQL create scriptu, naplnění databáze daty pomocí SQL insert scriptu a největší pozornost se věnuje dotazům.

Student musí vytvořit alespoň 25 dotazů, z nichž všechny musí být formulovány v jazyku SQL a alespoň 10 z nich musí mít ekvivalentní formulaci v relační algebře. Také se kontroluje, zdali dotazy pokrývají všechny kategorie dotazů.

[5]Kardinalita vztahu mezi entitami určuje, kolik objektů vstupuje do daného vztahu. Například kardinalita vztahu 1:1 označuje vztah, ve kterém na obou stranách vystupuje pouze jeden objekt dané entity (např. vztah manželé mezi entitou muž a entitou žena)[3]

[6]Parcialita vztahu vyjadřuje povinnost a volitelnost existence vztahu (např. vztah manželé - musí mít každý muž manželku?)[3]

1.3 Současný způsob opravování SP

Celou práci student povinně dokumentuje do XML[7] souboru předepsané struktury. Tento XML soubor spolu s dalšími soubory nahrává do školního webového adresáře webdev, odkud je přístupný k prohlížení vyučujícímu.

Opravování probíhá na cvičeních, nebo po předem zvoleném termínu posledního možného odevzdání, nebo se studenti domluví se svým vyučujícím (například pomocí emailu), když chtějí nebo potřebují práci odevzdat dříve (s případnými bonusovými body) nebo naopak později (s případnou penalizací).

Proces opravování SP jsem rozdělil do několika základních kroků

1. Nalezení SP či více SP připravených k opravě
2. Samotné opravení SP
3. Uložení hodnocení a prohlížení výsledku

které rozeberu podrobněji dále a zaměřím se u nich na problémy, se kterými se vyučující setkává.

1.3.1 Nalezení SP či více SP připravených k opravě

Prvním z faktorů, který vyučujícímu ztěžuje proces opravy, je nalezení SP studentů, které chce opravit. SP se nacházejí ve školním webovém adresáři, v jednotlivých soukromých složkách studentů. Tyto složky studentů jsou rozdělené podle školních uživatelských jmen.

Vyučující má sice k dispozici seznamy svých studentů podle uživatelských jmen, takže samotné nalezení SP nepředstavuje výraznější problém, nicméně tento způsob vyhledávání prací není ideální. Vyučující si nemůže například jednoduše vyfiltrovat studenty podle paralelek či podle toho, kteří mu již SP odevzdali, a kteří ne.

1.3.2 Samotné opravení SP

Největším problémem současného řešení opravování SP je, že vyučující musí vše procházet a opravovat ručně, i když se dá spousta částí opravit, či předpřipravit k opravování, automaticky[8]. To má za důsledek, že oprava je časově velice náročná a při vysokém počtu opravovaných semestrálních prací na cvičeních nemusí být kontrola adekvátní a opravující učitel může snadno něco přehlédnout, nebo z časových důvodů opravu některých částí přeskočí.

[7] Extensible Markup Language

[8] Sice existuje nástroj na poloautomatickou kontrolu SP, ale kvůli špatné použitelnosti nebyl nikdy používán ve větší míře, viz analýza bakalářské práce Lucie Klimperové v části 1.4.3

Dalším důležitým faktorem při opravování SP je nemožnost jednoduše označit v SP místo, kde je chyba či v tomto místě alespoň vytvořit komentář. Vyučující tak ztrácí povědomí, kde byla při posledním odevzdání chyba a při následující opravě neví, na co by se měl zaměřit. Student by se na druhou stranu mohl vždy podívat, co měl špatně a do příštího kontrolního bodu to mohl opravit.

1.3.3 Uložení hodnocení a prohlížení výsledku

Posledním krokem při opravě SP je uložení hodnocení a prohlížení výsledků. Vyučující ukládá pouze celkové bodové hodnocení s případným krátkým komentářem na školní stránky předmětu[9], kde je výsledek hodnocení k dispozici i pro studenta.

Jak pro studenta, tak ale ani pro vyučujícího to není optimální způsob ukládání výsledků. Student například v případě nespokojenosti s hodnocením kontaktuje svého vyučujícího, aby mu vysvětlil, za co přesně mu strhl body a vyučující musí znovu zjišťovat, co bylo špatně, což je další zbytečná práce navíc.

1.4 Aktuální možnosti automatické kontroly SP

Na fakultě Informačních technologií ČVUT v Praze se používají aktivně dva systémy na automatické opravy - Moodle a ProgTest. Přímo na opravování semestrálních prací v předmětu DBS byl vytvořen nástroj AutDBS v rámci bakalářské práce, jejímž autorem je Lucie Klimperová. Nástroj AutDBS byl ještě rozšířen a odladěn v rámci projektu FRVŠ[10] studentem FIT ČVUT Janem Matysem.

1.4.1 Moodle

Moodle je softwarový balíček, který nabízí rozsáhlou sadu nástrojů na tvorbu výukových systémů. Moodle umožňuje vytvářet kurzy (předměty) a každý kurz je sestavován z tzv. modulů činností a studijních materiálů. Moduly činností jsou například Anketa, Přednáška, Test či Úkol.[4]

Modul Úkol umožňuje učitelům sbírat studentské práce, hodnotit je a poskytovat studentům zpětnou vazbu včetně známkování. Tento modul podporuje čtyři typy úkolů:[5]

- Offline činnost - nabízí pouze vystavení zadání studentům
- Odevzdání souboru - umožňuje studentovi nahrát právě jeden soubor k odevzdání

[9] Školní stránky předmětu Databázové systémy jsou spravovány v rámci serverového clusteru edux.fit.cvut.cz

[10] Fond rozvoje vysokých škol

- Pokročilé nahrávání souborů - umožňuje nahrát více souborů k odevzdání

- Online text - studenti mohou psát řešení přímo na Moodlu pomocí editačních nástrojů

Online text umožňuje editování textu přímo z prohlížeče, takže by student nemusel pokaždé stahovat, změnit a nahrát soubor, aby opravil drobnou chybu v textu, jako je tomu v současnosti. Navíc umožňuje učiteli vkládat komentáře ke studentovu řešení či přímo do jeho textu (například barevně zvýraznit část studentova textu).

Žádný další z těchto typů úkolů nenabízí žádný sotisfikovanější nástroj, který by se dal použít k opravě SQL dotazů a dotazů relační algebry, či na kontrolu validity obsahu nahraných souborů.

Nicméně Moodle by mohl umožňovat lepší řízení kontroly předmětu pomocí nastavování termínů odevzdání a díky systému ukládání souborů, by se tak dala kontrolovat jednotlivá data odevzdání.

1.4.2 ProgTest

Systém Progtest je testovací prostředí navržené původně pouze na testování programů v rámci výuky programování na fakultě Elektrotechnické a na fakultě Informačních technologií Českého vysokého učení technického.

Základní funkcionalitou systému Progtest je schopnost zkompilovat zdrojový kód[11] a příslušný program poté automaticky otestovat na sadě testovacích dat. Samotné testování sestává ze dvou částí. První částí je příprava zadání, referenčního řešení a testovacích dat[12] učitelem. V druhé části probíhá samotné vyhodnocení studentova řešení. Hodnotí se především výstup ze studentova programu, který je porovnáván s výstupem referenčního řešení, ale dalšími kritérii pro hodnocení může být také paměťová a časová složitost, kdy se hodnotí, kolik paměti resp. času program spotřebuje. Součástí celého testovacího procesu je také uložení výsledku a poloautomatická kontrola opisování.

Nicméně systém ProgTest, stejně jako systém Moodle, nenabízí žádný sotisfikovanější nástroj, který by se dal použít k automatické opravě dotazů v jazyce SQL či dotazů v relační algebře.

Ale stejně jako nástroj Moodle, tak i ProgTest by se dal využít pro efektivnější správu odevzdávání souborů, kde by se daly nastavit termíny odevzdávání a dalo by se kontrolovat, kdo kdy co odevzdal.

[11] Progtest podporuje řadu programovacích jazyků, avšak aktivně se využívá především jazyk C a C++

[12] Testovací data je možné pomocí generátoru testovacích dat generovat pro každá jednotlivá testování nová, originální.

1.4.3 AutDBS - bakalářská práce Lucie Klimperové

Nástroj na Automatickou kontrolu semestrálních prací z DBS AutDBS byl vytvořený Lucií Klimperovou v rámci bakalářské práce na Fakultě elektrotechnické ČVUT v Praze v roce 2010. Umožňuje kontrolu semestrální práce pomocí programu Ant[13] v příkazové řádce. Jedná se o sestavovací utilitu vytvořenou v programovacím jazyce Java, která se podobá utilitě *make*. Narozdíl od *make* je Ant platformově nezávislý, protože nevyužívá přímo příkazů operačního systému.[6] Program AutDBS byl ještě rozšířen v rámci projektu FŘVS v roce 2012 o sadu ant skriptů studentem FIT ČVUT Janem Matysem, ale nikdy nebyl plně nasazen, byl nasazen pouze v testovacím režimu.

AutDBS umožňuje zkontrolovat validitu dílčích souborů[14] semestrální práce, provést SQL příkazy z vybraných SQL skriptů nad vybranou databází a vrátit výsledky těchto příkazů ve formě textových souborů. AutDBS také umožňuje vygenerování reportu (hlášení z jednotlivých kontrol) pro jednu nebo více semestrálních prací najednou do XML souboru a tento report se dá následně převést do formátu HTML.[6] Nástroj je tak určen nejen pro samostatnou kontrolu studentů před odevzdáním, ale i pro hromadnou kontrolu více semestrálních prací najednou.

Nevýhodou nástroje AutDBS je jeho nízká uživatelská přívětivost. Instalace a konfigurace AutDBS je velice zdlouhavá, a vyučující by se museli seznámit s jeho nepříliš jednoduchým používáním, což by vyžadovalo spoustu času. A i když nabízí užitečné funkce na částečnou kontrolu SP, tak se v současné době nevyužívá právě kvůli jeho špatné použitelnosti a s nejvyšší pravděpodobností se ani v budoucnosti využívat nebude.

1.5 Shrnutí

Semestrální práce v předmětu Databázové systémy má pevnou strukturu a je ve formátu XML, což umožňuje automatizovanou kontrolu jejích částí.

Automatizovanou kontrolou SP se již zabývala Lucie Klimperová ve své bakalářské práci a vytvořila multiplatformní nástroj AutDBS pracující v příkazové řádce. Tento skriptovací nástroj byl později vyladěn a rozšířen o další funkce studentem Janem Matysem, avšak nikdy nebyl plně využíván.

Nástroj AutDBS také využíván nebude vzhledem k použitým technologiím a odlišnému návrhu řešení u nově se vyvíjejícího systému na celkovou podporu předmětu Databázové systémy.

K řešení automatické kontroly školních prací se používají na FIT ČVUT také systémy Moodle a ProgTest. Jedno z využití, které by tyto systémy mohly poskytnout, by bylo řízení a kontrola odevzdávání souborů semestrální práce,

[13] Another Neat Tool

[14] Semestrální práce se skládá ze souborů: *main.xml* - obsahuje dokumentaci celé práce, *create.sql* - skript na vytvoření databáze, *insert.sql* - skript na naplnění dat databáze a *queries.sql* - obsahuje seznam všech SQL dotazů

kde by se dalo využívat odevzdávacího systému a díky tomu nastavovat termíny odevzdání a kontrolovat, kdy studenti odevzdali svá řešení.

Avšak problematika automatického opravování semestrální práce v předmětu DBS je natolik specifická, že nelze využít ani jeden z nich k opravě jednotlivých částí semestrální práce (například je nelze využít ke kontrole validity dotazů v jazyce SQL či dotazů v relační algebře).

Z pohledu celkového procesu opravování a kontroly se dá na základě zkušeností vyučujících s těmito systémy při sběru požadavků a navrhování řešení přizpůsobit výsledný nástroj na automatickou kontrolu tak, aby se v něm vyučující nemuseli potýkat se stejnými či podobnými problémy, jako například u výše zmíněných systémů Moodle a ProgTest.

Kapitola 2

Systém na celkovou podporu předmětu Databázové systémy

V době psaní této bakalářské práce byl systém na celkovou podporu předmětu Databázové systémy ve vývoji. Hlavním důvodem vzniku tohoto systému bylo ulehčení práce vyučujícím v předmětu Databázové systémy, protože v současné době provádí téměř veškeré činnosti spojené s výukou ručně, i když by se spousta z nich dala zautomatizovat.

Navíc se každoročně zvyšuje počet studentů a nově se podle schváleného doporučeného průchodu studiem přesouvá předmět Databázové systémy z 2. ročníku do 1. ročníku, takže počet studentů bude ještě mnohem vyšší než doposud (je to dáno tím, že do 2. ročníku zpravidla postupuje výrazně méně studentů).

Od tohoto systému je očekáváno, že ulehčí práci jak studentům, tak vyučujícím a jedním z konkrétních nástrojů, který by se měl k tomu využít, by byl nástroj na automatickou kontrolu semestrálních prací.

2.1 Softwarový týmový projekt - SP1, SP2

Systém na celkovou podporu předmětu Databázové systémy začal být vyvíjen v rámci předmětu SP1 a SP2 v roce 2014 na Fakultě informačních technologií Českého vysokého učení technického v Praze pod vedením Ing. Jiřího Hunky, který je také vedoucím této bakalářské práce.

Softwarový týmový projekt je dvousemestrální školní předmět, v rámci kterého si studenti vyzkouší práci v týmu a projdou si celým cyklem vývoje softwaru od sběru požadavků, návrhu architektury a designu, přes implementaci, testování a v případě úspěchu projektu až po nasazení.

Členem tohoto projektu jsem byl i já (Jan Sýkora, autor této bakalářské práce) a velkou měrou jsem se podílel na návrhu a vývoji tohoto systému. Po skončení tohoto předmětu (tj. po skončení předmětu SP2 v zimním se-

mestru 2014) jsem pokračoval v rámci této bakalářské práce v rozvoji a rozšíření nástroje na automatickou kontrolu semestrálních prací. Mimo rámec své bakalářské práce jsem navíc zaučoval, vypomáhal a vedl nový tým studentů (absolvující předmět SP1 v letním semestru roku 2015), který navazoval na práci předchozích týmů v tomto projektu.

Tento projekt byl specifický tím, že se na jeho vývoji nepodílel pouze jeden studentský tým, nýbrž se na něm podílely dva týmy (každý po šesti členech). Každý tým měl za úkol navrhnout a implementovat jednu specifickou část systému. Tento projekt lze tedy zařadit svým rozsahem mezi ty největší na FIT ČVUT.

2.2 Cíloví uživatelé

Vzhledem k povaze projektu jsou cíloví uživatelé systému předem a jasně definováni a není tedy nutné vynakládat prostředky na zjištění cílových uživatelských skupin. Cílovými uživateli systému jsou na jedné straně studenti a na druhé vyučující.

U obou uživatelských skupin se předpokládá adekvátní schopnost používání počítače, naopak se budou řadit mezi zkušenější uživatele. Později při návrhu uživatelského rozhraní se tak bude moci naopak navrhnout takové řešení, které zkušeným uživatelům může posupně urychlovat práci se systémem.

Dále se u uživatelské skupiny vyučujících předpokládá, že budou rozumět odbornému obsahu, tj. že se vyznají v problematice databází. Nicméně u studentů, kteří se pravděpodobně teprve seznamují s databázemi, se naopak musí počítat s tím, že nebudou rozumět některým odborným termínům a bude tedy vhodné zaměřit se i na možnost nápovědy a vysvětlivek.

Avšak toto spadá již pod návrh uživatelského rozhraní a vzhledem k tomu, že primárním účelem v týmovém projektu byla implementace funkčního systému a problematice uživatelského rozhraní nebylo věnováno tolik času, nebyla potřeba se více zabývat specifickými vlastnostmi cílových uživatelských skupin.

2.3 Požadavky

Pro přesné stanovení chování systému a jeho kvality byly na začátku projektu definovány požadavky. Stanovení požadavků je první krok, který by se měl vždy provést při vývoji jakéhokoli systému. Vynechání nebo nedostatečná alokace zdrojů na definici požadavků je pak častou příčinou selhání softwarových projektů.

Ke kategorizaci požadavků lze použít různé metody, jednou z nich je metoda FURPS. Název této metody je akronym skládající se z pěti základních pohledů na kvalitu vyvýjeného softwaru: [7]

- F (functionality) - funkčnost

- U (usability) - použitelnost
- R (reliability) - spolehlivost
- P (performace) - výkon
- S (supportability) – podporovatelnost / rozšiřitelnost

2.3.1 Funkčnost

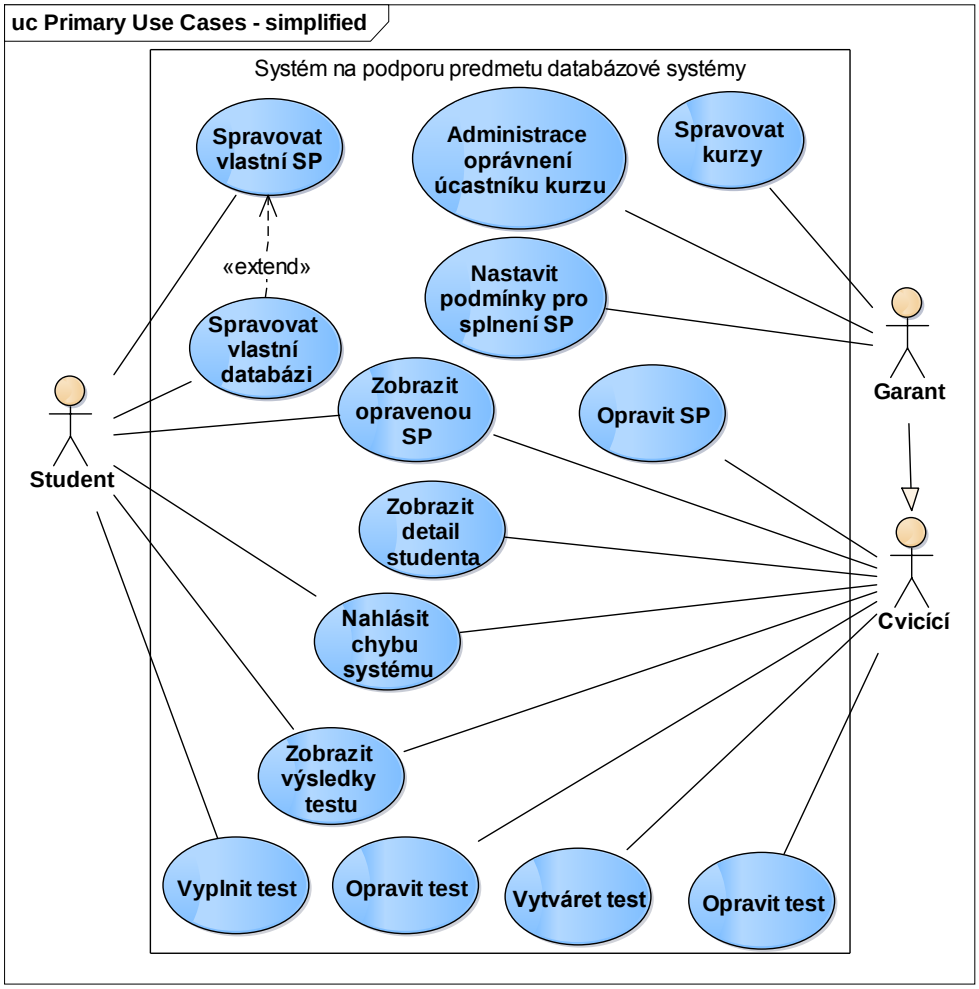
Funkčnost systému se zaměřuje na to, jestli software naplňuje všechny definované požadavky. Pro grafické znázornění funkčních požadavků a jejich specifikaci jsem zvolil diagram případů užití. Diagram případů užití (Use Case Diagram) se používá k popisu chování systému z hlediska uživatele a zachycuje, které typy uživatelů se systémem pracují a jaké činnosti v rámci systému vykonávají. Umožňuje znázornit funkční požadavky na systém tím, že popisuje interakci mezi ním a uživateli.[8]

Zjednodušený diagram případů užití pro tento systém (obrázek 2.1) názorně ukazuje, jaké funkce mohou nad systémem provádět jednotlivé kategorie uživatelů. Pro základní seznámení se se systémem je naprosto dostačující, nicméně v něm nelze zachytit všechny důležité funkce a vztahy mezi nimi. Proto je v příloze C.1 k dispozici podrobný diagram případů užití.

Funkční požadavky kladené na tento systém lze rozdělit do několika základních skupin:

- Přihlášení do systému a správa účtů
- Správa rolí a práv
- Správa předmětů a jednotlivých kurzů
- Hlášení chyb a jejich řešení (Bug report)
- Testy
- Semestrální práce

Jednotlivé logické celky funkčních požadavků rozeberu dále včetně aktuální situace jejich implementace. Výjimkou budou požadavky na semestrální práci - ty nebudu rozebírat v této části mezi ostatními požadavky na systém. Nástroji na podporu semestrální práce bude věnována samostatná kapitola (kapitola 3), ve které zároveň popíši návrh a vývoj 1. prototypu nástroje na podporu SP.



Obrázek 2.1: Zjednodušený diagram případů užití Systému na podporu předmětu Databázové systémy

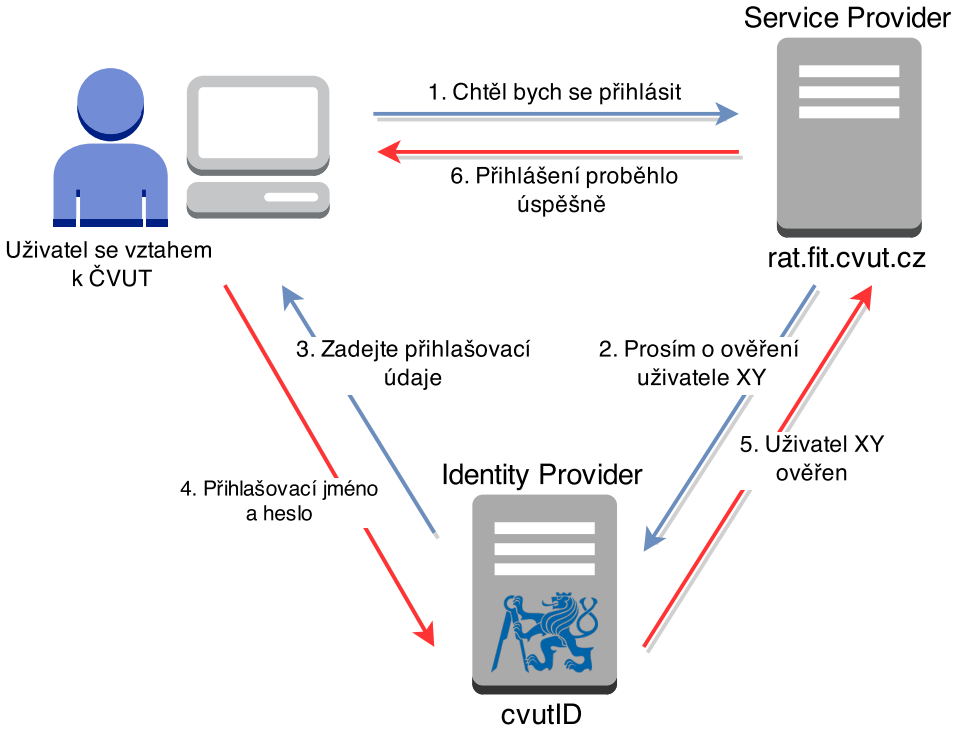
2.3.1.1 Přihlášení do systému a správa účtů

Přihlášení do systému je implentováno pomocí softwarového balíku Shibboleth, který umožňuje také SSO[15] - jednou se přihlásit a přistupovat k ostatním službám bez nutnosti zadávat znovu jméno a heslo. Zaručené odhlášení je potom ale možné pouze zavřením svého prohlížeče Internetu (nebo ručním smazáním cookies).[9]

Například studentovi tak stačí se pouze jednou přihlásit do nějaké školní aplikace přes Shibboleth pomocí svého cvutID a už se nemusí znovu autorizovat. Na obrázku 2.2 je uveden příklad, jak funguje přihlášení do další webové aplikace RAT, určené na podporu předmětu Databázové systémy, která ale

[15]Single Sign On

funguje naprosto odděleně od tohoto systému. Tento způsob autorizace je tedy naprosto totožný s tím, který využívá tento systém.



Obrázek 2.2: Schéma autorizace pomocí Shibbolethu[1]

Každý uživatel má vedený účet v systému. Pokud se příhlásí do systému, ale ještě nemá založený účet, tak se mu automaticky založí nový. Když se vytváří nový kurz, tak se každému účástníku kurzu (tj. vyučujícímu nebo studentovi) vytvoří účet, pokud jej ještě nemá založený.

V rámci svého účtu si uživatel může pouze zvolit některý z podporovaných jazyků a může si vybrat kurz, ve kterém je veden. Nic jiného nenabízí obecný účet v systému a veškerý systém oprávnění je již unikátní pro každý kurz.

2.3.1.2 Správa rolí a práv

Na začátek je důležité zmínit, že se nejedná o autorizaci a nastavení práv v rámci celého systému (autorizaci uživatele oproti systému zajišťuje softwarový balíček Shibboleth, viz sekce 2.3.1.1). jedná se o správu oprávnění v rámci předmětů a jejich jednotlivých kurzů.

Systém oprávnění daného uživatele je tedy unikátní pro každý kurz. Toto podporuje například situaci, kdy jeden uživatel může být v jednom kurzu stu-

dentem a v dalším kurzu vyučujícím (častý případ starších schopných studentů - pomocných cvičících).

Správu rolí a práv má na starost garant předmětu. Garant tedy bude moci upravovat oprávnění pouze v rámci svých předmětů (potažmo kurzů). Mezi základní požadavky na správu rolí a práv patří:

- Vytvořit roli
- Přiřadit roli uživateli (přidat/odebrat)
- Zrušit roli
- Přiřadit oprávnění roli (přidat/odebrat)
- Automatický import nastavení rolí a práv do následujících kurzů

Garant má tedy k dispozici rozhraní, ve kterém může uživatelům přiřazovat role a u těchto rolí definovat, co přesně mohou uživatelé s touto rolí dělat v systému. Uživatel může mít libovolný počet rolí.

Každá role má unikátní přiřazení oprávnění v rámci kurzu. Tím je například vyřešeno to, že student, který již absolvoval předmět a tento předmět je již uzavřen, tak má oprávnění pouze prohlížet své výsledky, ale už nemůže dále odevzdávat svoji SP. Není tedy nutné vytvářet například role *aktivní student* a *neaktivní student*.

V systému je stanoveno 5 základních rolí, které jsou definované pro každý kurz:

- Student
- Cvičící (instructor)
- Zkoušející (examiner)
- Přednášející (lecturer)
- Garant (guarantor)

Tyto role jako jediné nemůže garant zrušit, samozřejmě jim ale může upravovat příslušná práva.

Tento systém rolí byl převzat ze školní databáze KOS. Pro potřeby systému plně postačí pro vyučující pouze role cvičící a garant. U studenta není nutné žádné další rozdělení ani v našem systému. Nicméně garant má možnost si vše nastavit.

Instance Práva je přístupná pro celý předmět, tedy je přístupná ve všech kurzech tohoto předmětu. Například právo nahlédnout na všechna odevzdání své semestrální práce má přiřazeny role Student ve všech kurzech. Garant však

nemá možnost nové právo vytvořit nebo stávající zrušit, garant má pouze možnost využívat ta, která jsou k dispozici. Je to dáno způsobem implementace a odebírání či přidávání práv tak musí garant řešit na úrovni rolí.

Pokud chce nějakému uživateli či skupině uživatelů přidat či odebrat nějaké specifické oprávnění (například všem pomocným cvičícím chce odebrat právo na opravování SP), tak musí vytvořit novou roli (například role *pomocný cvičící*), v té nadefinovat příslušná oprávnění, přiřadit jim tuto novou roli a odebrat stávající roli *cvičící*. Garant tedy nemá k dispozici nějaký sotisfikovaný nástroj na kopírování či dědění rolí.

V současné chvíli ještě není funkční implementace automatického importu nastavení rolí a práv do následujících kurzů, která by například pro kurz DBS B141[16] automaticky nastavila poslední konfiguraci práv a rolí z předcházejícího kurzu DBS B131[17]. Ostatní požadavky jsou již plně implementovány.

2.3.1.3 Správa předmětů a jednotlivých kurzů

Dalšími důležitými požadavky ohledně správy předmětu je, že garant daného předmětu má mít možnost vytvořit kurz, a to včetně automatického naplnění daty ze školní databáze KOS, které zahrnuje:

- Vytvoření paralelek
- Vytvoření účtů pro studenty a vyučující (pokud je již nemají vytvořené)
- Automatické přiřazení rolí studentům a vyučujícím
- Automatické přiřazení studentů a vyučujících do paralelek

Nezbytným požadavkem je ještě možnost synchronizace kurzu se školní databází KOS. Typickým příkladem použití této funkce je, že po vytvoření kurzu a naplnění daty dochází na začátku semestru ke změnám v paralelkách (vznikají, zanikají, mění se jejich parametry jako výuková místnost, čas výuky, počet studentů či si studenti mění paralelky atd.), přihlašují a odhlašují se studenti aj. Tato funkce tedy zajistí, že aktualizuje databázi systému se školní databází podle nejnovější situace.

Garant, ani nikdo jiný, naopak nemá možnost ručně přeřazovat v systému studenty (ale ani vyučující) mezi paralelkami. Toto je také jeden z požadavků na systém, závazné je tedy to, co si daný účastník kurzu zapsal ve školním systému KOS. Typickým příkladem může být, když chce student docházet na jiné cvičení, než má zapsané. Vznikaly by tak nekonzistence. Je tedy na účastnících kurzu, aby dodržovali předem daná nastavení.

Všechny tyto požadavky jsou již plně implementovány a řádně otestovány. Nástroj na vytvoření a synchronizaci kurzů je tedy funkční.

[16] B141 je KOSí kód pro zimní semestr roku 2014[10]
[17] B131 je KOSí kód pro zimní semestr roku 2013[10]

2.3.1.4 Hlášení chyb a jejich řešení (Bug report)

Dalším požadavkem, který je nezbytný k fungování jakéhokoli většího systému, je implementace Bug reportu - systému na hlášení chyb. Bug report je již implementovaný a umožní přihlášenému uživateli odeslat informaci o nalezené chybě a tato chyba (bug) se automaticky zařadí do seznamu všech chyb (bug listu).

Tento seznam chyb je spravován ve webové aplikaci Redmine[18]. Řešení těchto úkolu lze přiřazovat konkrétním vývojářům, stanovovat jednotlivým chybám priority, sledovat historii a průběh odstraňování těchto chyb aj.

Uživatelé mají možnost si prohlédnout seznam všech chyb i s jejich stavem a v případě, že narazí na chybu, kterou už ohlásil někdo jiný, tak ji můžou "lajknout", aby se nevytvářel záznam znovu pro tu samou chybu. Poté, co je chyba vyřešena, se uživatelům, kteří tuto chybu ohlásili, odešle email, že tato chyba je již odstraněna.

2.3.1.5 Testy

Systém na podporu předmětu Databázové systémy obsahuje kromě nástroje na podporu semestrální práce ještě druhý důležitý nástroj na podporu testů. Tento nástroj byl vyvíjen druhým týmem v rámci týmového projektu. Nástroj na testování využívá společné rozhraní systému (autorizace uživatelů, Bug report, správa rolí a práv aj.). Celá kostra systému byla navržena a implementována týmem, který se zabýval semestrální prací.

Testy mohou být dvojího typu - testy v semestru a zkouškové testy. Obsahově si jsou tyto testy naprosto rovnocenné (mohou obsahovat stejné položky, funkčně se v ničem neliší, žádný z nich nenabízí oproti druhému nějakou zvláštní funkcionalitu), pouze se liší v tom, že zkouškové testy může vytvářet a aktivovat garant předmětu. Podíváme se tedy podrobněji, co přesně má podle stanovených požadavků umožňovat nástroj na testování jeho uživatelům.

Nástroj má podle požadavků umožňovat

- Cvičícím
  - Vytvořit šablonu testu
  - Upravit šablonu testu
  - Smazat šablonu testu
  - Vytvořit test v semestru
  - Aktivovat test v semestru
  - Opravit test v semestru
  - Opravit zkouškový test

[18] Redmine je open source software určený na řízení projektů a bug trackingu[11]

- navíc garant bude moci
  - Vytvořit zkouškový test
  - Aktivovat zkouškový test
- Student bude mít možnost
  - Vyplnit test (zkouškový i test v semestru)
  - Vyzkoušet si test nanečisto

Navíc všichni vyučující mají možnost si zobrazit jakýkoli test, ať už opravený nebo neopravený. Student bude mít možnost pouze zobrazit svoje testy.

Problematika testování v předmětu Databázové systémy je však natolik rozsáhlá, že by se o ní dala napsat samotná práce. Proto se v rámci analýzy tohoto systému plně spokojíme pouze se seznámením se s tímto nástrojem a dále jej už nebudu více rozebírat.

2.3.2 Použitelnost

Použitelnost (usability) je jedním z klíčových atributů, pomocí kterého se dá měřit kvalita vytvořeného softwaru. Použitelnost také odkazuje na metody určené ke zlepšení jednoduššího používání již v průběhu návrhu softwaru. Použitelnost je dle Jakoba Nielsena definována pěti základními komponentami:

- **Naučitelnost** (learnability) - jak jednoduché je pro uživatele zvládnutí prvních úkolů po prvý čas setkání s aplikací/programem?
- **Efektivnost** (efficiency) - jakmile je je uživatel seznámen s designem, jak rychle je schopen plnit zadané úkoly?
- **Zapamatovatelnost** (memorability) - Když se uživatel vrátí k designu po delší době jeho nepoužívání, tak jak jednoduše se obnoví jeho znalosti z minula, jak složité pro něj bude znovu efektivně pracovat s tímto designem?
- **Chybovost** (errors) - Kolik chyb uživatel dělá? Jak závažné jsou tyto chyby? Jak jednoduše se dá z těchto chyb zotavit, jak složité je jejich řešení?
- **Spokojenost** (satisfaction) - Jak příjemné je pro uživatele s programem pracovat?

Existuje ovšem spousta dalších atributů kvality. Nejdůležitější z nich je prospěšnost (utility) - obsahuje design opravdu to, co uživatel potřebuje? Použitelnost a prospěšnost jsou stejně důležité a společně určují, jestli je design skutečně užitečný.[12]

Bohužel použitelnosti systému nebylo při vývoji věnováno mnoho času, protože většina zdrojů byla alokována na dodržení funkčních požadavků. Primárním cílem tohoto projektu v jeho prvním roce byla řádná implementace funkčních požadavků, a proto nezbylo mnoho času na věnování se použitelnosti systému. Tento problém je však hlavní náplní navazujícího projektu a této bakalářské práce, ve které se budu věnovat především použitelnosti nástroje na automatickou opravu semestrálních prací.

2.3.3 Spolehlivost

Spolehlivost nám říká, jaká je četnost a závažnost chyb, přesnost zpracovaných vstupů a výstupů. V této oblasti se také sledují možnosti obnovení provozu a zotavení se z výpadku. Patří sem také zátěžové testy a testy zotavení aplikace po selhání některých komponent řešení. Vzhledem k tomu, že až na manuální testy dvou komponent (synchronizace systému se školní databází a správa práv a rolí) neprošel tento systém nikdy řádným testováním, tak právě spolehlivost tohoto systému bude s největší pravděpodobností jeho nejslabším místem. Sice v navazujícím projektu a v rámci této bakalářské práce se ještě provedou minimálně uživatelské testy, ale ty nezabrání vysoké pravděpodobnosti výskytu chyb.

2.3.4 Výkon

Výkon systému by měl zajistit, že bude schopný bez problému zvládnout přibližně 500 - 1000 uživatelů. Tento požadavek vychází z pravidelného počtu studentů a zvolené hardwarové požadavky na server, na kterém poběží systém, zaručí, že hardwarové vybavení bude pro tento počet uživatelů dostačující (více informací o technických parametrech naleznete v sekci 2.4).

2.3.5 Rozšiřitelnost

Rozšiřitelnost (Podporovatelnost) je další z faktorů, na který se kladl velký důraz. Jedním z požadavků je, aby se aplikace dala jednoduše rozšířit o další vlastnosti. Příkladem může být vytvoření nového nástroje či přidání podpory dalšího jazyka. Velký důraz byl kladen na zvolení takových technologií, u kterých bude garantováno, že budou podporovány i po skončení projektu, aby byla zajištěna udržovatelnost systému (více o zvolených technologiích viz sekce 2.4).

2.4 Zvolené technologie

Na začátek bych si dovolil uvést, jaké technologie jsme použili při samotném návrhu systému, protože vzhledem k tomu, že systém byl vyvíjen ve dvou

šestičlenných týmech, tak se musel vybrat nástroj nejen na návrhy diagramů, modelování požadavků aj., ale musel se zvolit i systém pro řízení projektu.

Na návrh modelů požadavků, diagramů aktivit, business procesů, datových modelů, modelů databáze aj. jsme zvolili nástroj **Enterprise Architect**. Enterprise architect je modelovací nástroj nabízející spoustu funkcí, od jednoduchého kreslení diagramů až po generování dokumentace, generování zdrojových kódu, aplikování reverzního inženýrství či využití verzovacích systémů[13]. Na řízení projektu jsme zvolili systém Redmine. Díky tomuto systému mohl týmový (ale také projektový) vedoucí kontrolovat činnost jednotlivých členů týmu a jednoduše vést celý projekt. V systému Redmine byla také vedena (nepříliš přehledná a užitečná) dokumentace projektu.

Při výběru technologií, které jsme použili k implementaci, jsme museli vzít v potaz výše zmíněné funkční a nefunkční požadavky (viz sekce 2.3), abychom byli schopni je dodržet. Vzhledem k tomu, že jsme vytvářeli webovou aplikaci, tak jsme se museli rozhodnout, v jakém programovacím jazyce ji napíšeme a jaký zvolíme webový framework.

Programovací jazyk jsme zvolili jazyk **PHP**[19], protože s ním měli alespoň někteří členové týmu zkušenosti a webový framework jsme zvolili český **PHP framework Nette** (stabilní verze 2.3.1). Pro tento framework jsme se rozhodli z důvodu jeho kvalitní dokumentace, jednoduchého používání a zároveň to pro nás byla příležitost naučit se s nejpoužívanějším frameworkem v České republice.

Nette Framework je šířen jako svobodný software a jeho licence BSD umožňuje, aby ho mohl kdokoli používat - zdarma, i pro komerční účely. Nette dále nabízí zabezpečení webových stránek před narušením metodami jako jsou například XSS[20], CSRF[21], session hijacking, session fixation aj.[14]

Nette framework dále nabízí sotisfikovaný knihovnu **Tracy** na debugování a zpracování chyb (nebo také pod zdomácnělým názvem Laděnka), která umožňuje rychle odhalit a opravit chyby, logovat chyby (ve vývojovém, testovacím i produkčním režimu), měřit čas aj.[15] V neposlední řadě Nette podporuje **AJAX** / AJAJ, Dependency Injection, SEO, DRY, KISS, MVC, Web 2.0, cool URL a jiné technologie a koncepty.

Pro ukládání dat jsme zvolili relační databázový systém **PostgreSQL** (verze 9.3.6). Pro přístup k datům do databáze využíváme v Nette vrstvu Nette\Database. Tato knihovna nám umožňuje snadno skládat SQL dotazy, snadno získávat data, pokládá efektivní dotazy a nepřenáší zbytečná data a nenutí nás opakovat SQL kód.[16]

[19] Pro lokální vývoj jsem používal verzi PHP Version 5.5.9-1ubuntu4.6
[20] Cross-Site Scripting
[21] Cross-Site Request Forgery

```
$selection = $context->table('author');
$selection->where(':book:book_tags.tag.name', 'PHP')
          ->group('author.id')
          ->having('COUNT(:book:book_tags.tag.id) > 0');
```

Zdrojový kód 2.1: Příklad zjednodušení psaní složitějších dotazů pomocí třídy Nette\Database\Table\Selection

Nesmíme také zapomenout na zvolení systému řízení verzí pro chod implementace - **Git**. Git je distribuovaný, free, open source systém správy verzí. Git umožňuje mít více lokálních větví (branches), které mohou být na sobě navzájem nezávislé.[17] Git tedy umožňuje jednoduchý a efektivní vývoj jednoho SW ve větších týmech.

Systém v současné chvíli běží na školním serveru s těmito parametry:

- čtyřjádrový procesor Intel(R) Xeon(R) CPU E5-2630 0 @ 2.30GHz
- 8 GB RAM
- 50 GB HDD
- 64 bit Debian 7.6 (stable version)
- PostgreSQL 9.1, apache 2.2.22
- PHP 5.4.36

K systému se dále váže využití dalších technologií, od zabezpečení, autorizaci, zvláštní přístup k jiným databázím v rámci vývoje SP, využití externích nástrojů (například nástroje RAT[22]). Tento přehled je však dostatečně podrobný na to, aby si mohl člověk udělat obrázek o technologickém zázemí tohoto systému.

2.5 Shrnutí

V systém na podporu předmětu DBS bylo za dva semestry práce dosaženo implementace základních funkcí podle návrhu. Čili v systému je možné vytvořit nový kurz a naplnit jej daty ze školní databáze, spravovat systém rolí a práv, hlásit chyby v systému a co se týče nástroje na podporu testů, tak byly implementovány alespoň základní funkce, nicméně by se nástroj na podporu testů ještě nedal nasadit ani na ostré testování.

[22]Relational Algebra Translator

Nástroji na podporu semestrální práce bude věnována následující kapitola, nicméně ale aspoň trošku předběhnu a prozradím, že tento nástroj dosáhl většího pokroku než nástroj na testování.

Po skončení prvního roku tohoto projektu byly na tento systém nabrány další dva týmy, které mají pokračovat v práci týmů předchozích a jejich cílem je, aby dokončily vývoj tohoto systému tak, aby se dal naostro nasadit na testování v následujícím běhu předmětu Databázové systémy.

Kapitola 3

Nástroj na podporu semestrální práce - 1. prototyp

Nástroj na podporu semestrální práce v předmětu Databázové systémy je stěžejním prvkem systému na celkovou podporu předmětu DBS. Cílem 1. prototypu je návrh procesů pro efektivní kontrolu semestrálních prací a jejich implementace.

Kromě implementace funkcí pro kontrolu semestrální práce se v 1. prototypu má implementovat i rozhraní pro studenta, aby mohl v vytvářet celou svoji semestrální práci.

3.1 Funkční požadavky

Funkční požadavky na nástroj na podporu semestrální práce byly definovány na začátku vývoje vedoucím projektu, Ing. Jiřím Hunkou. Během vývoje byly průběžně validovány, a to z důvodu změn některých požadavků a ujasnění si požadavků z obou stran - vývojového týmu a zadavatele (Ing. Jiří Hunka). Důležitou částí validace požadavků bylo také ověřování jejich správnosti garantem předmětu Databázové systémy - Ing. Michalem Valentou, Ph.D.

Základním funkčním požadavkem na podporu SP je, aby se kdykoli v průběhu kurzu dala vytvářet, odevzdávat a opravovat SP stávajícím způsobem. Hlavním důvodem tohoto požadavku je, aby se v případě nestability či nefukčnosti systému dalo vždy pokračovat současným způsobem. Tento požadavek je pokryt v jednotlivých níže uvedených požadavcích. Jedná se především o požadavek nahrání SP, stáhnutí SP a odevzdání SP.

Funkční požadavky na nástroj na podporu SP z pohledu jednotlivých uživatelů[23]:

- Student bude moci

[23] Graficky znázorněné funkční požadavky na SP naleznete v příloze C.1

  - Nahrát SP (uploadovat/importovat)
  - Vytvářet SP
  - Stáhnout svoji SP
  - Odevzdat SP cvičícímu k opravě
  - Zkušebně otestovat SP
  - Zobrazit SP
- Cvičící bude moci
  - Opravit odevzdanou SP
  - Zobrazit libovolnou SP
  - Stáhnout libovolnou SP
- Garant předmětu bude ještě moci oproti cvičícímu navíc
  - Nastavit bodové hodnocení SP
  - Nastavit termíny odevzdání SP

3.1.1 Nahrání a stažení SP

Student bude moci nahrát kdykoli v průběhu vytváření SP jednotlivé části SP - XML soubor, konceptuální model a jeho zdrojové soubory, logický model a skripty. Při nahrávání XML souboru se jednotlivé elementy rozeberou a uloží. Student tak bude moci jednoduše převést svoji rozpracovanou SP. Stejně tak bude fungovat i zpětná vazba - stažení SP - student si bude moci stáhnout rozpracovanou SP zpět do XML souboru spolu s dalšími soubory. Stáhnout si samozřejmě bude moci i odevzdanou SP (včetně hodnocení). Vyučující bude moci ukládat SP kteréhokoli studenta.

3.1.2 Vytváření SP

Student bude moci při vytváření SP využívat rozhraní tohoto systému. Student tak už nebude muset upravovat dokumentaci ve zdrojovém XML souboru. Psaní dokumentace pro něj bude přehlednější. Navíc bude mít k dispozici pomocné nástroje, a to především při vytváření dotazů. Student si totiž bude moci vytvořit databázové připojení a testovat jakékoli své dotazy (ať v relační algebře či v SQL) nad svojí databází.

Dalším pomocným nástrojem při vytváření databáze bude automatické přiřazování kategorií. Student tedy nebude muset přiřazovat ručně všechny kategorie dotazů. Manuálně bude muset přiřazovat pouze ty kategorie, které nelze jednoznačně detekovat z textového znění dotazu. Nicméně tento požadavek ulehčí především cvičícím při opravě, protože nebudou muset kontrolovat pokrytí kategorií dotazů, když se budou kategorie dotazů přiřazovat a kontrolovat automaticky. Toto samozřejmě neplatí pro ručně přiřaditelné kategorie, ale těch je pouze pár.

3.1.3 Zkušební otestování a odevzdání SP

Před odevzdáním SP si může student kdykoli nechat zkušebně otestovat vybrané části své SP. Požadavkem na samotné odevzdání SP je především to, aby se každé odevzdání zálohovalo na webdev a dalo se tak kdykoli v průběhu semestru pokračovat v opravě současným manuálním způsobem, kdy stačí cvičícímu pouze přístup k nahraným částem SP (více o současném způsobu opravování SP se dozvíte v části 1.3).

3.1.4 Opravení odevzdané SP

Cvičící bude moci v tomto systému opravovat odevzdané SP. Cvičící bude mít k dispozici nástroj automatické kontroly, který mu předpřipraví odevzdanou SP k manuální opravě. Každá opravená práce se uloží a jak student, tak každý vyučující si bude moci nechat zobrazit výsledek opravy SP. Student si bude moci zobrazit pouze svoje odevzdané a případně opravené SP, cvičící si bude moci nechat zobrazit libovolnou studentskou odevzdanou SP.

3.1.5 Nastavení bodového hodnocení a termínů odevzdání SP

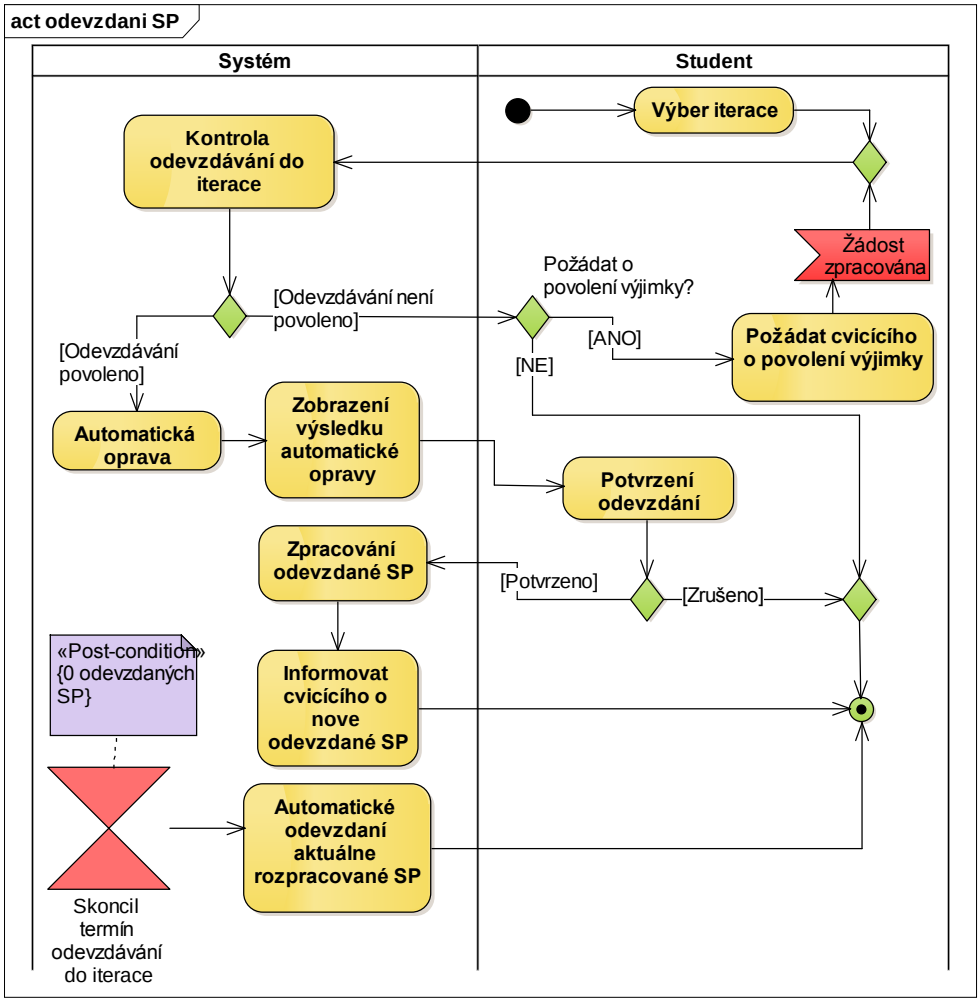
Garant bude mít k dispozici rozhraní pro nastavení bodového hodnocení jednotlivých částí SP v jednotlivých iteracích (kontrolních bodech). Garant bude také moci nastavit celkové hodnocení SP definující celkový počet bodů, které lze získat za SP, minimální počet bodů nutný k úspěšnému obhájení SP či maximální počet bonusových bodů za mimořádnou aktivitu nebo zajímavé řešení. K dispozici bude mít garant také rozhraní na nastavení termínů odevzdání SP. Termíny odevzdání se budou vázat k jednotlivým paralelkám, ve kterých jsou zapsáni studenti.

3.2 Návrh řešení

Dalším krokem při vývoji 1. prototypu byl návrh řešení, který vycházel ze stanovených požadavků a zvolených technologií. V návrhu řešení jsme se zaměřili na podrobnější specifikaci požadavků a vytvořili diagramy aktivit, stavové diagramy, modely tříd, doménový model a z něj vycházející relační model databáze. Těchto diagramů bylo vytvořeno skutečně velké množství a bylo by zbytečné je všechny uvádět. Proto zde rozeberu pouze ty nejdůležitější z pohledu automatické kontroly SP, a těmi jsou odevzdání SP a opravení SP.

3.2.1 Odevzdání SP studentem

Jak lze vidět na diagramu aktivity Odevzdání SP (obrázek 3.1), tak odevzdání může být iniciováno ze dvou stran. Buďto odevzdá SP sám student, nebo ji za něj automaticky odevzdá systém.



Obrázek 3.1: Diagram aktivit - Odevzdání SP

Student má možnost odevzdávat SP do vybrané iterace kolikrát chce, ale pouze do té doby, než mu bude odevzdávání uzamčeno. Uzamknutí odevzdávání může nastat pouze ze dvou důvodů: buďto již skončil termín odevzdávání do dané iterace[24], nebo mu odevzdávání uzamknul cvičící, který začal opravovat jeho poslední odevzdání v dané iteraci.

Důležité je tedy zmínit, že ve chvíli, kdy student odevzdá SP, tak jej může jeho cvičící kdykoli začít opravovat a musí tudíž studentovi uzamknout odevzdávání SP (aby mu student během opravy nemohl odevzdat další SP), i když ještě neskončil řádný termín odevzdávání.

Student má možnost požádat svého cvičícího o povolení znovu odevzdat SP v případě, kdy má uzamčené odevzdávání. Častou příčinou tohoto scénáře

[24]Termín odevzdávání na začátku semestru stanoví garant pro každou paralelku.

může být předčasné odevzdání SP studentem, kdy cvičící nalezl chyby ve studentově odevzdání, ale dává mu možnost si to ještě opravit. Jiným častým případem může být povolení znovuodevzdání SP z důvodu nemoci studenta.

U každého povolení znovuodevzdávání musí navíc cvičící uvést důvod, kvůli kterému jej povoluje. Nicméně znovuodevzdávání by rozhodně nemělo být nějakým častým jevem. Student má totiž k dispozici nástroje na zkušební otestování, pomocí kterých si může bez odevzdání vyučujícímu otestovat svoji práci a jakmile ji tedy odevzdá k opravě, tak již jeho práci může brát vyučující jako hotovou.

Pokud má tedy student povolené odevzdávání do dané iterace, tak je jeho práce nejprve automaticky předopravena a studentovi je zobrazen výsledek této automatické opravy. Z tohoto výsledku automatické opravy pak následně vychází i opravující cvičící (je mu zobrazeno přesně to samé, co studentovi), student má tak ještě naposled možnost zamítnout odevzdání, opravit případné chyby a odevzdat později.

V případě, že student potvrdí výsledek automatické opravy a chce pokračovat v odevzdání, tak se jeho SP uloží v systému, zazálohuje na webdev, zařadí do fronty odevzdaných prací připravených k opravě a příslušnému cvičícímu je odesláno upozornění, že mu byla odevzdána práce a je připravena k ruční opravě.

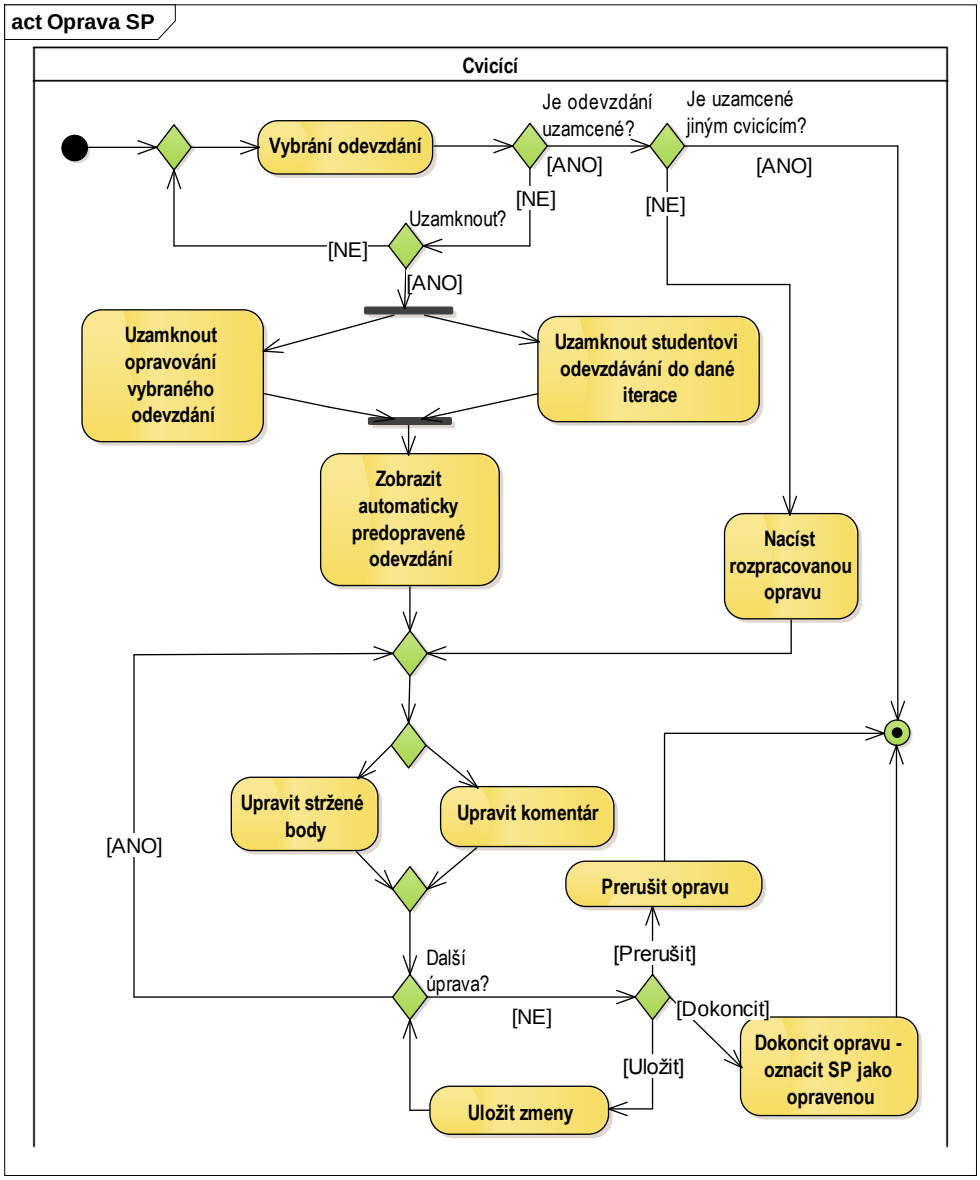
3.2.2 Automatické odevzdání SP

Druhý způsob způsob odevzdání, automatické odevzdání SP systémem, nastane ve chvíli, kdy vypršel čas řádného odevzdávání a student neodevzdal žádnou práci do dané iterace. Systém tedy automaticky za studenta odevzdá jeho aktuálně rozpracovanou SP (nezáleží na tom, kolik SP obsahuje chyb). Nutným předpokladem k automatickému odevzdání SP je, aby odevzdaná SP splňovala daný formát, což je díky prostředí, ve kterém je vytvářena, vždy zaručeno. Automatické odevzdání tak vždy proběhne bez problémů.

3.2.3 Oprava odevzdané SP

Druhou důležitou částí v automatické kontrole SP je oprava odevzdaných SP cvičícím. Cvičící si nejprve vybere odevzdání, které chce kontrolovat. Pokud student odevzdal do dané iterace vícekrát, tak učitel bude moci vždy opravit pouze poslední odevzdání. Všechna předchozí odevzdání jsou ale snadno dohledatelná a dají se zobrazit či stáhnout.

V dalším kroku se ověří, zdali je vybrané odevzdání uzamčené nebo odemčené. V případě, že odevzdání ještě nikdo nezačal opravovat, tak je odemčené, cvičící si jej může pro sebe zamknout, zároveň uzamknout odezvdávání studentovi do dané iterace a provést opravu. Zamezí se tak tomu, aby dva cvičící opravovali jednu semestrální práci zároveň. V případě, že je již



Obrázek 3.2: Diagram aktivit - Oprava SP

uzamčené jiným cvičícím, tak pouze garant má možnost odemknout toto odevzdání k opravě jinému cvičícímu.

V průběhu opravy bude cvičící moci upravovat stržené body a komentáře za jednotlivé části odevzdání. Dále bude mít možnost stav své aktuální opravy kdykoli uložit, přerušit opravu a nebo ji dokončit a označit SP jako opravenou. Pokud cvičící přeruší opravu, tak se k ní může kdykoli vrátit a opravu dokončit. Jakmile je oprava dokončená, tak se studentovi ještě odešle upozornění a jak cvičící tak student mají možnost si zobrazit hodnocení opraveného odevzdání.

Další procesy spojené s nástrojem na podporu SP zde již nebudu více rozebírat, protože je již z jejich názvu jasné, co se v nich přesně děje a zároveň byly výše popsány všechny důležité kroky během životního cyklu jednoho odevzdání.

3.2.4 Databáze

Při návrhu databáze jsme nejprve navrhli doménové modely, pomocí kterých lze popsat jedntolivé entity a vztahy mezi nimi v našem systému a následně jsme navrhli databázové modely tak, aby šly vytvořit v PostgreSQL. [25]

V databázovém modelu kostry systému mezi základní entity patří `Users` (uživatel), `Parallel` (paralelka), `Course` a `Subject` (kurz a předmět), `Role` a `Permission` (role a práva) a `Bug` (chyba). Rozbor těchto entit a vysvětlení vztahu mezi nimi naleznete v sekci Požadavky.

V databázovém modelu semestrální práce mezi základní entity patří `Semester_work` (semestrální práce, SP), `Turn_in` (odevzdání), `Iteration` (iterace, kontrolní bod), `Requirement` (požadavek), `Deadline` (termín odevzdání) a jednotlivé opravované části semestrální práce (`Description`, `Data_model`, `Relational_model`, atd.). Dovolil bych si upozornit na tabulku `Turn_in` (odevzdání). `Turn_in` je dekompozicí vztahu m:n mezi entitami `Semester_work` a `Iteration`. Vyjadřuje skutečnost, že jednu SP lze odevzdat ve více iteracích a že v jedné iteraci lze odevzdat více SP. Na tomto modelu by se dala popsat ještě spousta dalších vztahů, ale vzhledem k velikosti tohoto modelu (cca 40 entit) a k tomu, že se tento model velmi často mění kvůli změnám požadavků nebo nedostatečné podpoře pro implementaci, nebudu popisovat tento model více do hloubky. Nicméně pro základní seznámení se s datovou vrstvu tohoto systému je více než dostačující, protože základní vztahy mezi entitami zůstanou zachovány.

3.3 Implementace

Implementace nástroje na podporu SP byla psána pomocí jazyka PHP ve frameworku Nette. Vycházela tedy z technologiích zvolených pro implementaci celého systému (viz 2.4 - Zvolené technologie). Implementace probíhala ve ve druhém semestru tohoto týmového projektu (tedy v předmětu SP2, navazující na předmět SP1) a výsledkem bylo vytvoření logiky a základního uživatelského rozhraní nástroje na podporu SP.

V této části se zaměřím na důležité třídy, které mají na starost logiku nástroje na podporu SP a popíši využívaný návrhový vzor MVP.

[25] V přiloženém CD naleznete databázový model kostry systému a databázový model semestrální práce.

3.3.1 Návrhový vzor MVP

Nette využívá návrhový vzor MVP[26], který je derivací známějšího návrhového vzoru MVC[27].

Návrhový vzor MVC rozděluje aplikaci do tří vrstev: na datový model, uživatelské rozhraní a řídící logiku a modifikace některé z nich má minimální dopad na ostatní. Tato separace je velice užitečná pro udržení přehledného kódu, obzvláště v případech, kdy na jedné aplikaci pracuje více lidí. MVC také určuje vztah jednotlivých komponent[18].

- **Model** zajišťuje přístup k datům a manipulaci s nimi.
- **View** (pohled) převádí data reprezentovaná modelem do podoby vhodné k prezentaci uživateli.
- **Controller** (řadič) reaguje na události pocházející od uživatele a zajišťuje změny v modelu nebo v pohledu.

Tyto dva návrhové vzory jsou si velice podobné. Jejich nejdůležitějším společným rysem je stejné rozdělení aplikace do tří vrstev, které jsou na sobě navzájem co nejvíce nezávislé.

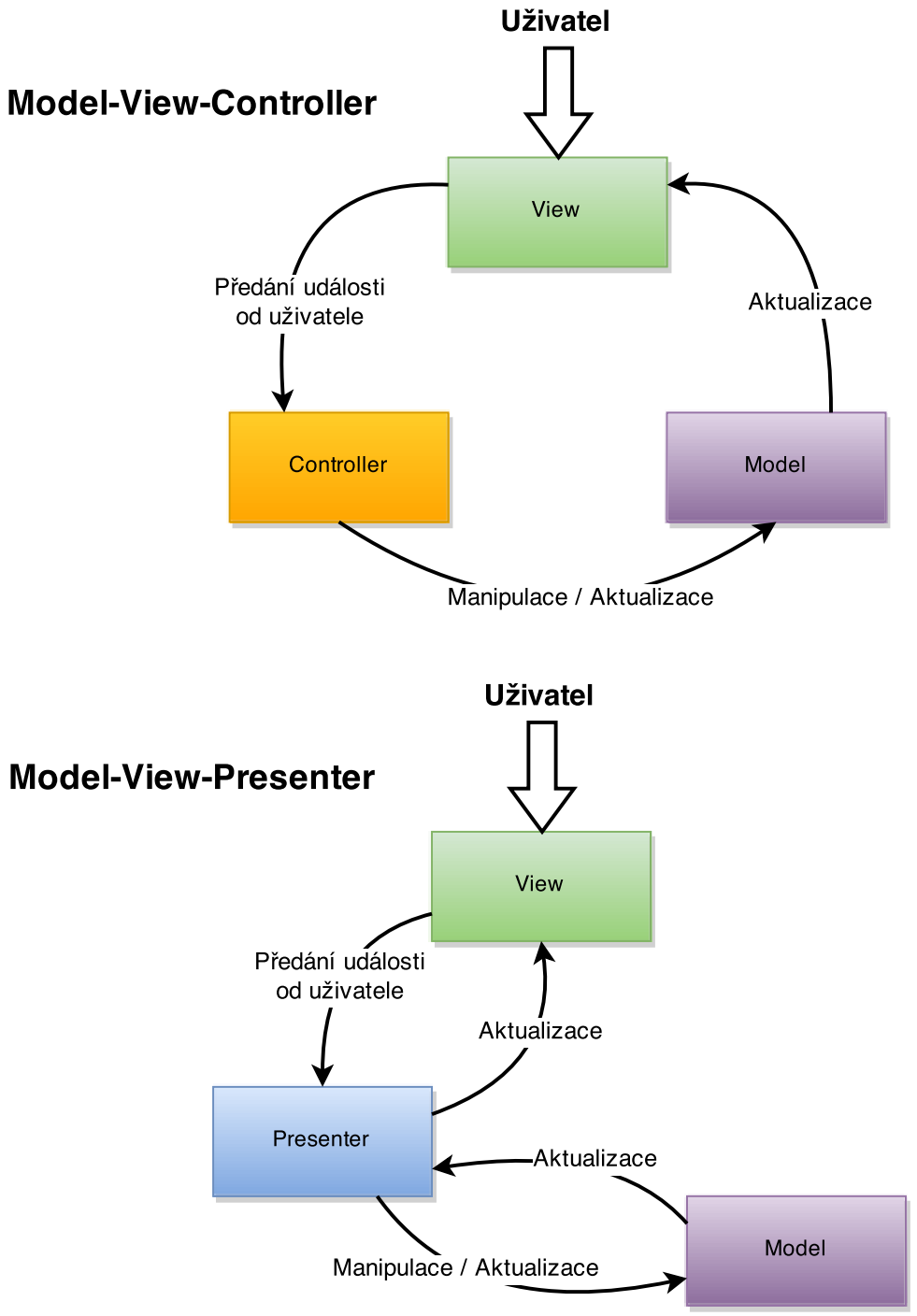
Na obrázku 3.3 jsou vidět závislosti a rozdíly mezi jednotlivými prvky těchto návrhových vzorů. V MVC interakce uživatele ve View vyvolá událost, která je delegována na Controller. Controller ji zpracuje a aktualizuje Model. Model pak spustí událost, která aktualizuje View. Hlavní rozdíl mezi MVP a MVC je v tom, že MVP se vyhýbá závislosti mezi Modelem a View tím, že všechny události řídí Presenter. Tedy každá změna v Modelu se neprojeví rovnou ve View, ale musí ji před tím zpracovat Presenter.[19]

Výhodou MVP oproti MVC je lepší testovatelnost díky oddělení view a modelu, ale nevýhodou u MVP je zase větší náročnost implementace, protože všechny aktualizace musí být zpracovány v presenteru.[20]

Pro MVP tedy zjednodušeně platí, že

- Model nemá vědět o tom, že nějaké view a presentery existují
- View o modelu vědět také nemusí (pasivní view), nebo naopak může data získávat přímo z něj, záleží na zvolené koncepci
- Presenter seznámí View s Modelem (ne naopak) a realizuje uživatelské akce. Ty patří do tří kategorií
  - změna view (nejčastější)
  - změna stavu (interakce v rámci aktuálního view)
  - příkaz pro model

[26] Model-View-Presenter
[27] Model-View-Controller



Obrázek 3.3: Rozdíl mezi návrhovými vzory MVC a MVP

Model ve většině případů bude tvořit více tříd. Jedna z nich se může starat právě o zapouzdření připojení k DB, které využijí jiné třídy modelu. Ale z pohledu celé aplikace je to chápáno jako jeden celek, jeden model.[18] Nejdůležitější třídy logické vrstvy (model) ale i důležité presentery si rozebereme níže.

3.3.2 SemesterWork

Třída `SemesterWork` má na starosti nahrání a vytváření SP (stažení SP není v tomto prototypu ještě implementováno). Pro nahrání SP (import hlavního XML souboru, diagramů a skriptů) využívá třídy `XmlParser` a `XmlCreator`. Třída `XmlParser` se používá při nahrávání XML souboru. Umožňuje z něj získat všechny dílčí části tak, aby je třída `SemesterWork` mohla uložit do systému. Třída `XmlCreator` se používá, opět jak napovídá sám název, k vytvoření semestrální práce ve formátu XML. Jak lze tedy vytušit, tak v ukládání SP ze systému ve formátu XML již téměř nic nebrání a vše je pro to připraveno.

Při vytváření SP tedy třída `SemesterWork` poskytuje základní přístup k databázi pro vytváření, aktualizaci a mazání jednotlivých částí SP.

3.3.3 Iterations

Třída `Iterations` nabízí rozhraní pro přístup k iteracím. Umožňuje tedy zjistit požadavky na danou iteraci - co se má přesně kontrolovat ve vybrané iteraci, kolik se dá za jednotlivé části v opravě v dané iteraci strhnout bodů. Také nabízí rozhraní pro aktualizaci požadavků v dané iteraci.

3.3.4 SemesterWorkBackup

Třída `SemesterWorkBackup` se využívá při odevzdávání SP. Při odevzdání aktuálně rozpracované semestrální práce se uloží všechny její části pomocí této třídy do databáze systému.

3.3.5 Requirement

Abstraktní třída `Requirement` nabízí pouze tři abstraktní funkce: `getName()`, `getMessage()` a `check()`. Metoda `check()` musí být implementována všemi třídami, které dědí od abstraktní třídy `Requirement` a v této třídě je implemenována kontrola požadavků. V průběhu metody `check()` se také logují nalezené chyby a ty jsou pak přístupné pomocí metody `getMessage()`.

Od abstraktní třídy `Requirement` dědí třídy, které již implementují jednotlivé konkrétní požadavky v SP, které se mají opravovat ve vybraných iteracích

- `Description` - popis SP
- `Queries` - dotazy

- `DataModel` - konceptuální model
- `RelationalModel` - logický model
- `CreateScript` - skript na vytvoření databáze
- `InsertScript` - skript na naplnění databáze testovacími daty
- `Conclusion` - závěr SP
- `References` - Odkazy / Literatura

V těchto třídách je tedy definováno, co přesně se má v jednotlivých požadavcích kontrolovat (například minimální počet slov v popisu) a jak se mají opravovat.

3.3.6 EvaluationService

`EvaluationService` je třída, která obstarává veškerou podpůrnou logiku pro opravování SP. Mezi to patří například základní předvyplnění bodů vypočítané na základě výsledků automatické opravy a nastavení bodového hodnocení garantem předmětu, zpracování manuální opravy (dočasné uložení opravy, dokončení a uložení výsledku opravy) či získání potřebných dat při opravování vícekrát odevzdané SP do jedné iterace (je potřeba získat výsledky z předchozí opravy a porovnat je s aktuálním odevzdáním).

3.4 Shrnutí

Výsledkem ročního snažení v předmětu SP1 / SP2 týmu, který měl také na starost nástroj na podporu SP, je funkční prototyp[28]. Tento prototyp však neprošel testováním a chybí mu i některé důležité funkce. Mezi ty nejdůležitější patří záloha odevzdaných SP na školní webový adresář webdev, uzamykání opravování odevzdané SP před ostatními opravujícími cvičícími, nedotažené automatické přiřazování a kontrola kategorií dotazů, systém notifikací a ukládání SP.

Kromě automatického zálohování odevzdaných SP na webdev, kde jediné, co chybělo k úspěšné realizaci, bylo vytvoření účtu s potřebnými právy na webdevu od administrátora tohoto webového adresáře, je zbytek chybějících funkcí způsoben pozdější změnou požadavků nebo (a s tím spjatým) nedostatkem času na dokončení, protože již na začátku tohoto projektu bylo zřejmé, že se nepodaří vše dodělat během jednoho roku.

Od toho byl ale nabrán nový tým, opět v rámci předmětů SP1 / SP2, který má za úkol dodělat chybějící funkcionality, zaměřit se na použitelnější uživatelské rozhraní tohoto nástroje a zajistit nasazení a údržbu tohoto systému v následujícím běhu předmětu Databázové systémy.

[28] V příloze C naleznete vybrané screenshoty 1. prototypu.

Mým dalším cílem bylo zaměřit se na zefektivnění tohoto nástroje tak, aby učitelům ještě více ulehčil opravování. V tomto bodě se tedy setkaly mé cesty a cesty nově nasazeného týmu, protože jsme měli společně na starost tento nástroj na podporu SP.

Rozhodl jsem se nad rámec své bakalářské práce pomáhat nově nabraným členům týmu v seznámení s celým systémem a v zaškolení využívaných technologií, zastával funkci vývojového poradce a prakticky zastával funkci vedoucího jejich týmu po celý běh předmětu SP1.

Kapitola 4

Uživatelské testování 1. prototypu

4.1 Uživatelské testování

Poté, co byl vytvořen 1. prototyp nástroje na podporu SP, tak bylo potřeba zaměřit se na zlepšení jeho použitelnosti. Existuje spousta metod na zlepšení použitelnosti, ale nejužitečnější z nich je uživatelské testování. Uživatelské testování se skládá ze čtyř základních kroků:

1. Sehnat **reprezentativní uživatele**.

2. Položit uživatelům **reprezentativní úkoly** na práci s designem.

3. **Pozorovat**, co uživatelé dělají, v čem jsou úspěšní, a co jim naopak činí potíže.

4. **Vyhodnotit výsledky pozorování** a zohlednit je v další fázi vývoje.

I když bylo zřejmé ještě před prvním testováním, že 1. prototyp obsahuje velké množství chyb v uživatelském rozhraní a zdálo by se tak, že by se dalo rovnou přejít k návrhu a vývoji dalšího prototypu, tak by to bylo špatně. Díky uživatelskému testování aktuálního prototypu se nejenom naleznou chyby v aktuálním uživatelském rozhraní, ale dají se identifikovat i dobré části, které tak mohou zůstat a ušetří se čas na návrhu a vývoji dalšího prototypu. Jinou výhodou je opětovné upřesnění požadavků, protože mojí testovanou skupinou reprezentativních uživatelů budou její budoucí opravdoví uživatelé, kteří si tak přímo mohou vyzkoušet práci v systému a budou mít prostor k podání připomínek. Správně by tak mělo každému návrhu a implementaci dalšího prototypu předcházet uživatelské testování.

4.2 Reprezentativní uživatelé

Reprezentativní uživatelé pro testování nástroje na podporu SP jsou v našem případě jasně daní - jsou jimi cvičící v předmětu Databázové systémy. Případnou alternativou pro výběr reprezentativních uživatelů by mohli být takoví uživatelé, kteří se orientují v problematice databázových systémů a mají zkušenosti s výukou, například tedy další cvičící z FIT ČVUT. V našem případě však není nutné více rozebírat problematiku výběru a rekrutování reprezentativních uživatelů díky tomu, že se jedná o školní systém a cílová uživatelská skupina je tedy předem určena.

4.3 Počet testovaných uživatelů

Vhodný počet uživatelů potřebných na jedno testování je 4 až 8. Toto rozmezí vychází z výzkumu Jakoba Nielsena a Toma Landauera, kteří ukázali, že poměr chyb N, nalezený během jednoho uživatelského testování z celkového počtu chyb v systému je roven výrazu

$$N = 1 - (1 - L)^n$$

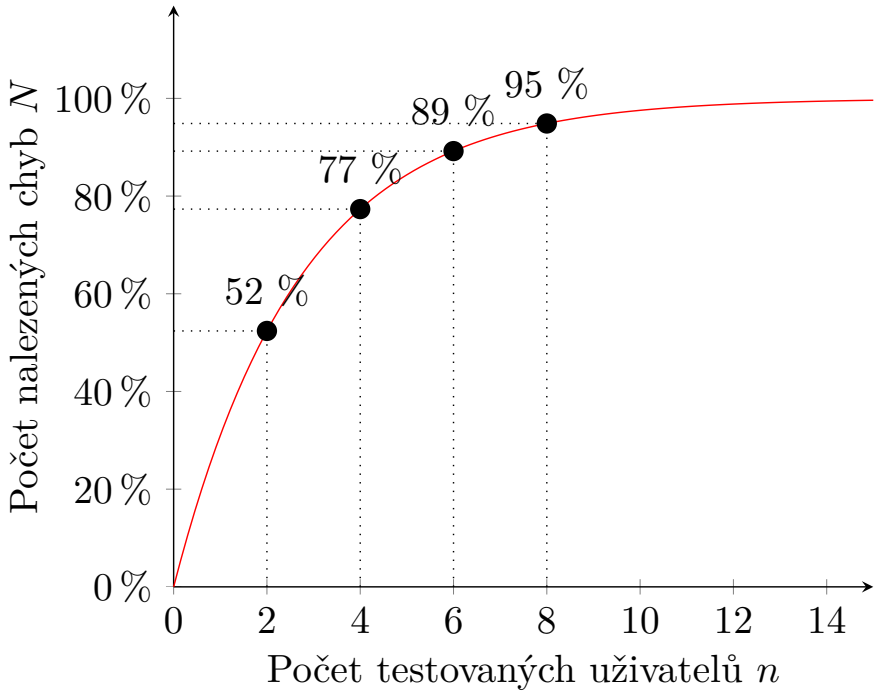
kde L je poměr chyb, které nalezne jeden testovaný uživatel z celkového počtu chyb v systému a n je počet testovaných uživatelů. Typická hodnota proměnné L je 31%[29], pro tuto hodnotu byl také vykreslen graf 4.1, na kterém je názorně vidět kolik uživatelů je potřeba k úspěšnému kvalitativnímu uživatelskému testování[21]

Jak lze vyčíst z grafu, tak ve chvíli kdy budeme provádět uživatelské testování s více než 8 uživateli, tak takové testování je neefektivní, protože nám nepřinese téměř žádnou přidanou hodnotu (už při 4 uživatelích jsme schopni odhalit přibližně 77 % chyb v UI) a zároveň budeme mít mnohem větší náklady na testování. Více uživatelů na testování znamená alokovat více času nejen na samotný průběh testování, ale hlavně se musí alokovat další finanční prostředky.

Podle Nielsen Norman Group ([22]), kde provedly výzkum s 201 společnostmi využívajícími uživatelské testování (polovina z nich byla z USA, další měly zastoupení např. ve Velké Británii, v Kanadě, Austrálii, kontinetální Evropě, ale i v Brazílii, Číně, Mexiku, Indii aj.) měly tyto firmy náklady na jednoho uživatele na testování v průměru 171 dolarů. Tato hodnota se samozřejmě velice lišila podle lokace a zaměření, nicméně je vidět, že to není částka zanedbatelná, a že opravdu nemá smysl provádět testování s velkým množstvím uživatelů.

Existují samozřejmě výjimky, kdy je vhodné testovat více uživatelů a mezi ně patří například:

[29]Hodnota $L = 31\%$ je průměrná hodnota získaná z velkého množství projektů ze studie Jakoba Nielsena a Toma Landauera[21]



Obrázek 4.1: Závislost počtu nalezených chyb na počtu testovaných uživatelů

- **Kvantitativní studie** (quantitaive studies) - cílíme na statistiky, zajímá nás například průměr a směrodatná odchylka toho, jak dlouho uživatelům trvá vykonání určitého procesu, často se řídí normálním rozdělením, testujeme alespoň s **20 uživateli**, abychom získaly statisticky významné hodnoty[23]

- **Card sorting** - necháme uživatele roztřídit předem dané pojmy do libovolného počtu kategorií, případně dodatečně požádáme uživatele o setřídění do větších skupin, vhodné pro návrh struktury menu rozsáhlého e-shopu, či pro restrukturalizaci interního firemního systému, testujeme třikrát s alespoň **15 uživateli**[24]

- **Eyetracking** - sledujeme, čeho si uživatelé všímají nejvíce, co čtou jako první, kde se pohybují nejčastěji myší, využití heatmap[30], gazeplots[31],testujeme s alespoň **39 uživateli**[25]

Nejdůležitější je však pro zlepšení UI kvalitativní testování a k tomu nám tedy plně postačí alespoň 4 reprezentativní uživatelé. Pro testování prvního prototypu jsem sehnal právě čtyři cvičící předmětu Databázové systémy. Díky

[30]Heatmapy reprezentují barevně oblasti, na kterých uživatel strávil nejvíce času a na kterých nejméně

[31]Gazeplot reprezentuje pohyb očí uživatele, zajímá nás, kudy se uživatel pohyboval očima

tomu jsem měl zajištěno, že bude odhalena většina chyb již během prvního testování.

4.4 Iterativní testování

Důležité je však také dodržet iterativní způsob testování. Podle obrázku 4.1 by se mohlo zdát, že stačí provést jedno testování s 15 uživateli k odhalení všech chyb v použitelnosti, ale mnohem lepší je rozdělit testování na jednotlivé iterace. Podle Jakoba Nielsena ([21]) je optimální provést 3 iterace testování pokaždé s 5 uživateli.

Důvodem, proč neprovedeme pouze jedno velké testování je, že sice po vytvoření nového designu se zbavíme všech chyb z minulého designu, avšak nový design může (a téměř jistě bude) obsahovat chyby nové, protože nikdo nedokáže vytvořit perfektní, bezchybný design. V druhém testování se nalezne většina ze zbylých chyb z předchozí staršího designu (v případě testování s 5 uživateli zbývalo cca 15 % chyb) a zároveň se objeví i chyby v novém designu. Zbylé chyby (přibližně 2 %) se tak odstraní ve třetí iteraci.

V případě testování nástroje na podporu SP hodlám tedy provést minimálně ještě jedno testování, kde budu opět testovat s minimálně 4 uživateli. Díky tomu budu mít zaručeno, že odhalím přibližně 95 % chyb v uživatelském rozhraní pro cvičící.

4.5 Úkoly

Samotnému průběhu testování předcházelo vytvoření úkolů, které mají uživatelé splnit. Tyto úkoly jsem navrhl tak, abych jimi pokryl všechny důležité aktivity cvičícího v chystaném nátroji. Úkoly byly následující:

1) Opravte semestrální práci z 1. iterace od studenta Matěj Nikl (niklmate)

2) Opravte semestrální práci z 2. iterace od studenta Matěj Nikl (niklmate)

3) Studentu Matěj Nikl (niklmate) povolte znovu odevzdat semestrální práce do 2. iterace

4) Opravte znovu odevzdanou semestrální práci z 2. iterace od studenta Matěj Nikl (niklmate)

5) Zahajte opravu semestrální práce studenta Matěj Nikl (niklmate) z 3. iterace

- **a**) Opravte důsledně všechny dotazy (ale tak, aby vám to nezebralo příliš mnoho času).
- **b**) Proveďte alespoň jednu změnu v komentáři u libovolného dotazu a alespoň jednou strhněte body u libovolného dotazu

c) Uložte provedené změny a opusťte opravování (výsledky neodesílejte)

d) Znovu zahajte opravování sem. práce, kterou máte zčásti opravenou z předchozích kroků a dokončete opravu (odešlete výsledky)

6) Najděte všechna odevzdání pro 1. iteraci (veškerá odevzdání, tj. i ta, která nebyla opravena) od studenta Matěj Nikl (niklmate). Podívejte se na výsledky té práce, kterou jste opravil/a v jednom z předchozích kroků.

7) Opravte semestrální práci z 1. iterace studenta Marek Reimer (reimemar) (tj. Studenta, kterého nemáte na svém cvičení)

4.6 Kde testovat

Dalším krokem byl výběr vhodného místa pro testování. Pokud má člověk štěstí, tak má na uživatelské testování k dispozici uživatelskou laboratoř, kde jsou kamery na zaznamenání pohybu uživatele a jednosměrné sklo, pomocí kterého lze pozorovat uživatele. Hlavní výhodou uživatelské laboratoře je oddělení testovaného uživatele od pozorovatele. Uživatel se tak může plně soustředit na zadané úkoly a není ovlivňován pozorovatelem. Ale jinak téměř vůbec nezáleží na tom, kde testování probíhá, pokud jsme schopni zajistit klidné prostředí a hlavně necháme uživatele samostatně pracovat na jeho úkolech. Při uživatelském testování je tedy důležité dodržet, abychom uživatele pouze pozorovali a nezasahovali do testování.[26]

4.7 Čemu věnovat pozornost při testování

Při pozorování bychom neměli poslouchat, co uživatelé říkají, nýbrž bychom měli sledovat co dělají. Doslova, jak říká Jakob Nielsen (doslovný překlad, [27]):

> *„Abyste navrhli co nejlepší uživatelské rozhraní, tak byste měli věnovat pozornost tomu, co uživatelé dělají, ne tomu, co uživatelé říkají. Požadavky hlášené samotnými uživately jsou stejně tak nespolehlivé jako jejich domněnky ohledně budoucího chování rozhraní. Uživatelé nevědí co chtějí.“* [32]

Zde ještě uvedu příklad z [27], na kterém lze krásně demonstrovat citaci uvedenou výše. Představme si situaci, kdy polovina respondentů tvrdí, že by

[32] To design the best UX, pay attention to what users do, not what they say. Self-reported claims are unreliable, as are user speculations about future behavior. Users do not know what they want.

nakupovala více z těch internetových obchodů, které by nabízely 3D náhled na produkt. Znamená to, že musíme rychle implementovat 3D náhledy do naší internetové stránky? Učitě ne. Znamená to pouze, že 3D zní skvěle. Svět je plný neúspěšných obchodních záměrů, které selhaly na postoji lidí k hypotetickým produktům a službám. Ve spekulativních průzkumech lidé jednoduše hádají, jak by se mohli chovat nebo jaké nástroje (featury) by se jim líbily. Neznamená to ale, že by je pak ve skutečnosti používali.

4.8 Dotazník

Dotazník je jeden ze způsobů, jak získat další informace k vylepšení designu. Obecně můžeme získat více informací o cílových uživatelích nebo současných problémech, ale také můžeme získat informace ohledně samotného testování. Cílovou skupinu uživatelů jsem již dobře znal, proto pro mne bylo vhodné získat především dodatečné informace o testování.

Své otázky jsem směřoval tak, aby se testovaní uživatelé mohli vyjádřit především k tomu, co jim dělalo největší potíže při testování, ale také, aby dostali prostor k vlastním návrhům či připomínkám na vylepšení. Nicméně nesmíme zapomenout (viz sekce 4.7), že nejdůležitější jsou naše vlastní postřehy z testování.

4.9 Pilotní testování

Před ostrým testováním jsem ještě provedl pilotní testování. Pilotní testování je vhodné pro hladký průběh ostrého testování. Odzkoušel jsem použité technologie, připravil správná testovací data a hlavně si prošel celý proces testování, abych odhadl přibližnou dobu trvání testování a objevil případné chyby v zadání.

4.10 Průběh testování

Ostré testování probíhalo v předem určených místnostech, kde byl dostatečný klid, aby nebyl při testování uživatel ničím vyrušován. K zaznamenání testu jsem používal kromě vlastních poznámek také nahrávání obrazovky včetně zvukové stopy (využíval jsem nástroj Screenpreso). Mohl jsem si tedy po skončení testování ještě jednou promítnout celý průběh testování znovu a v klidu jej analyzovat.

Uživatelé (cvičící) se nejprve seznámili se zadanými úkoly, a pokud jim bylo vše naprosto jasné, tak se pustili do jejich plnění. Během testování jsem se snažil co nejméně zasahovat do plnění úkolů, abych neovlivnil výsledky testování. Nicméně v některých fázích jsem musel vstoupit do testování, protože bylo potřeba splnit zadaný úkol, aby mohl uživatel pokračovat dále.

Na konci testování jsem uživatelům dal ještě vyplnit dotazník, a tím skončilo celé testování. Jedno testování trvalo průměrně půl hodiny.

4.11 Výsledky testování

4.11.1 Nalezení SP

Cvičící neměli problém nalézt SP připravenou k opravě. Odevzdané SP byly k dispozici na vlastní stránce a ve výchozím nastavení byly vyfiltrovány tak, že se zobrazovaly pouze SP těch studentů, které měl cvičící na svých cvičeních. Cvičícímu se tak nezobrazovaly SP „cizích studentů“ a mohl si rovnou vybrat, kterou SP začne opravovat.

Potíže s nalezením SP však nastaly ve chvíli, kdy cvičící dostali za úkol zobrazit všechny odevzdané SP jednoho studenta. Cílem tohoto úkolu bylo, aby cvičící našel historii všech odevzdání daného studenta. Cvičící nenapadlo hledat tato odevzdání v detailu studenta, nýbrž se je snažily najít přímo mezi odevzdanými SP připravenými k opravě a nebo mezi opravenými SP.

4.11.2 Opravení SP

Stěžejní bod testování, opravení SP, nedělal cvičícím žádné větší problémy. Jednotlivé části SP zůstaly na stejném místě přesně podle XML souboru, takže nalezli hledané položky opravy tam, kde by je očekávali.

Způsob zobrazení SP sice nebyl pro cvičící přehledný (viz např. obrázek C.4), protože jednotlivé části opravy od sebe nebyly vizuálně odděleny (toto se projevilo především při opravě dotazů), ale řádnému opravení to v ničem nebránilo, jelikož neměli problém s nalezením místa, kde se dá přidat komentář, či strhnout body. Jediný problém nastal ve chvíli, kdy očekávali, že po stržení bodů či přidání komentáře by měli tuto akci potvrdit.

Během opravy měl cvičící za úkol uložit provedené změny, opustit opravu, vrátit se k ní a dokončit ji (odeslat výsledky). Cvičí s tímto úkolem neměli mnoho problémů, rychle nalezli, že se tyto akce dají provádět na konci dokumentu (viz obrázek C.5), pouze nevěděli, co přesně tyto akce provedou. Zde by tedy stačilo pouze přidat popisky.

Při jendom z testů se cvičící nechtěně překlikl v průběhu opravy a přišel o všehnu svoji neuloženou práci, aniž by se ho systém na něco zeptal. Toto byl tedy jasný příklad toho, že musíme ošetřit opravování tak, aby cvičící nepřišel jednou drobnou chybou o všechnu svoji provedenou práci.

Celkově tedy nový způsob opravování přinesl cvičícím především možnost komentovat a bodovat jednotlivé části semestrální práce, ale kromě ověření správnosti syntaxe dotazů v SQL a v RA už ale nenabízel oproti současnému způsobu opravování žádné větší výhody.

4.11.3 Povolení znovu odevzdání SP

Během testování měli cvičící největší problém s povolením znovu odevzdání. Povolení znovu odevzdání bylo k dispozici na dvou místech:

- V tabulce opravených SP měla každá opravená SP ve sloupečku akcí možnost povolit znovu odevzdání
- V detailu studenta u všech jeho odevzdaných SP byla možnost povolit mu odevzdání od vybrané iterace

Zde byl problém v tom, že uživatele nenapadlo hledat tuto možnost ani na jednom z uvedených míst. A i když se například dostali po delší době hledání k detailu studenta, tak si zde nevšimli této možnosti.

Dále bylo častým problémem, že během vypisování důvodu, proč povolují znovu odevzdání, nevěděli, komu a do jaké iterace to vůbec povolují (viz obrázek v příloze D.2). Uživateli tedy chyběly důležité informace o akci, kterou prováděl.

4.11.4 Opravení znovu odevzdané SP

Opravení znovuo devzdané SP v našem systému se cvičícím oproti stávajícímu způsobu příliš neulehčilo. Jedinou výhodou bylo, že měli k dispozici zvýrazněný komentář a stržené body z předchozího odevzdání u daných položek opravy a mohli se tak orientovat alespoň podle nich.

Opravení znovu odevzdané SP bylo testováno ve druhé iteraci, kde se kontrolovaly pouze dotazy v přirozeném jazyce a databázové modely. Záměrně jsem vypustil opakovanou opravu pro třetí iteraci, protože bylo jasné, že by v současném provedení byla nepřehledná a nepřinesla by v tomto testování žádný užitek.

4.11.5 Shrnutí

Uživatelské testování 1. prototypu odhalilo nedostatky v některých procesech opravy SP. Odstranění těchto problémů bude vyžadovat lepší návrh uživatelského rozhraní a bude nutný zásah do aktuální implementace. Odstraněním těchto problémů se budu zabývat ve 2. prototypu nástroje na podporu semestrální práce.

Potěšující zprávou z celého testování je ale především to, že i podle vyjádření samotných testovaných cvičících by jim tento 1. prototyp již v této formě výrazně ulehčil práci oproti stávajícímu způsobu opravování SP.

Kapitola 5

Nástroj na podporu semestrální práce - 2. prototyp

Cílem 2. prototypu nástroje na podporu semestrální práce bylo dokončení důležitých funkcí z 1. prototypu, které se nestihly implementovat. Dále jsem se zaměřil u 2. prototypu na kvalitní návrh uživatelského rozhraní, aby se uživatelům s tímto nástrojem pracovalo co nejefektivněji.

Nakonec byl 2. prototyp řádně uživatelsky otestován a nalezené chyby byly zohledněny ve finální aplikaci. Finální aplikace by tedy měla být schopná plného nasazení v následujícím běhu předmětu Databázové systémy.

5.1 Vedení týmu

V letním semestru akademického roku 2014/2015 byl sestaven nový tým studentů z 2. ročníku v předmětu Softwarový týmový projekt 1. Členové týmu byli:

- Oldřich Malec
- Lukáš Vyčítal
- Nhan Van Nguyen
- Petr Balaš (v průběhu semestru ukončena účast na projektu)

Tento tým měl za úkol pokračovat ve vývoji nástroje na automatickou opravu SP. Původně bylo mým úkolem předat projekt a spolupracovat s tímto týmem při návrhu a vývoji 2. prototypu.

Rozhodl jsem se však nad rámec zadání své bakalářské práce zastávat navíc funkci vedoucího týmu. To sice znamenalo, že mi přibyla spousta práce, ale byl jsem ochotný tento čas věnovat do tohoto navazujícího projektu, abych zajistil jeho úspěšné dokončení, nebo aby se alespoň vykonalo co nejvíce práce. Jako vedoucí týmu jsem měl tedy na starost:

- Definování cílů projektu
- Strukturování projektu
- Časové plánování projektu (scheduling)
- Revidování plánu podle změn okolností projektu
- Monitorování a řízení vývoje
- Kontrolu kvality

Kromě vedení týmu jsem ale zároveň pokračoval společně s tímto týmem ve vývoji nástroje na automatickou opravu SP. Mým hlavním úkolem byl návrh nového řešení a dohlédnutí nad jeho realizací. Navíc, dokud se noví členové neseznámili s tímto projektem a použitými technologiemi[33], tak jsem se aktivně podílel i na implementaci.

5.2 Použité technologie

5.2.1 myBalsamiq

K vytváření nových návrhů uživatelského rozhraní jsem používal nástroj myBalsamiq. MyBalsamiq je nástroj na vytváření wireframe (WF). Wireframe v doslovném překladu znamená "drátěný model". Je to návrh, pomocí kterého lze definovat funkci celých webových stránek a prioritu toho, jaké informace se v nich mají zobrazovat.

Hlavní důvod, proč jsem vybral tento nástroj je, že umožňuje vzdálenou spolupráci celého týmu při vytváření jednotlivých návrhů, tzv. mockups. Mockup je obecné označení pro model, který sice není funkční, ale lze pomocí něj definovat vzhled budoucího objektu. V případě vývoje SW může být mockup například pouze tužkou nakreslené schéma webové stránky. V našem případě je mockup vytvořený model jedné webové stránky pomocí kreslícího nástroje myBalsamiq.

Nástroj myBalsamiq je velice komplexní webová aplikace, která umožňuje kromě vzdáleného vytváření WF více uživateli také řízení projektu[34] (rozdělení uživatelů do skupin a jednotlivých projektů, přidělení práv) a řízení verzí (umožňuje evidování historie změn při vytváření mockups, archivace projektů aj.).[28]

Nástroj myBalsamiq je placený, ale pokud je použit pro neziskové projekty, či pro školní účely, tak jej lze používat bezplatně. Mně se povedlo získat tuto bezplatnou licenci. Díky tomu se dá tento nástroj využít i pro další školní

[33] doba trvání byla přibližně do třetiny až poloviny semestru podle schopností a znalostí jednotlivých členů týmu

[34] Projektem se zde myslí pouze projekt na vytváření WF v aplikaci myBalsamiq

projekty[35] a vyučující tak může jednoduše spravovat více školních projektů na jednom místě.

5.2.2 AdminLTE

Pro celkový vzhled systému jsem zvolil webovou šablonu AdminLTE. AdminLTE je open source šablona pro webové aplikace, která je založená na CSS frameworku Bootstrap 3. Implementuje responzivní HTML šablonu, využívá ve svém designu Bootstrap componenty a mění styly běžně užívaných zásuvných modulů (pluginů) a díky tomu vytváří konzistentní design, který může být použit jako uživatelské rozhraní webových aplikací.[29]

AdminLTE je založen na modulárním designu, který může být jednoduše upravený pro naše specifické účely. Díky této šabloně tedy jednoduše docílíme toho, aby se naše aplikace dala používat na různých zařízeních, aniž bychom museli něco složitě programovat a navíc poskytuje estetický design.

5.2.3 Grido

Grido je komponenta, která se dá využít ve frameworku Nette. Grido umožňuje

- Výpis a formátování dat
- Řazení
- Stránkování
- Filtrování
  - Textové filtrování
  - Checkbox filtr
  - Selectbox filtr
  - Číslený typ
  - Definování vlastních filtrů
- Hromadné akce
- Inline editace
- Export dat do souboru formátu CSV
- Rozšíření klientské části, například:
  - Označování řádků s podporou shiftu

[35] V letním semestru 2015 byl tento nástroj využíván na dalších dvou projektech - Reálný návrh databáze architektonického kulturního dědictví a Webový objednávkový systém na pronájem zařízení

- Invertování výběru řádků
- Odeslání filtru po změně hodnoty
- Url odkaz s parametry
- Odscrollování po změně

Grido podporuje různé typy datových zdrojů: Nette Database, Dibi, Doctrine a Array. Mohli jsme tak jednoduše využívat knihovnu NetteDatabase k získávání dat na naplnění grido tabulek.[30]

Grido jsme využívali k implementaci multifunkčních tabulek. Ukázka použití takové tabulky je například na stránce odevzdaných SP (obrázek 5.6) či na stránce se seznamem studentů a jejich výsledky (obrázek 5.3).

5.3 Menu

5.3.1 (Ne)rozbalovací položky

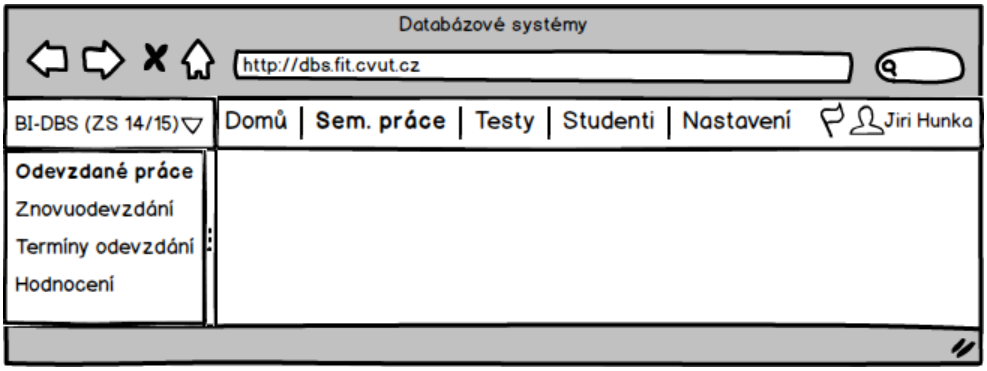
Jeden ze základních bodů, který jsem využil při návrhu zněl, aby menu nebylo rozbalovací, protože podle studie provedené Nielsen Norman Group (viz [31]) rozbalovací menu (drop-down menu) uživatele obtěžuje nebo zdržuje.

Příkladem, proč nepoužívat rozbalovací menu může být špatný výběr položky v menu. Uživateli zabere více času opravit se a vybrat tu správnou položku, než kdyby menu nebylo rozbalovací. Navíc pro uživatele s horší motorikou a sníženou přesností pohybu myší se toto stává ještě více nepříjemné, protože je pro ně obtížnější kolikrát správně rozbalit a najet do tohoto menu.

Dalším důvodem, proč nepoužívat rozbalovací menu je, že uživatel nevidí obsah jednotlivých položek menu. Pokud tedy neví, kde přesně se nachází cílová položka, tak musí postupně rozbalovat hlavní položky menu jednu za druhou.

Ještě jako jeden příklad uvedu, když uživatel prohlíží stránky pomocí dotykového zařízení (typicky mobil, tablet). V dotykovém zařízení totiž nelze přejet přes rozbalovatelné položky menu, aby se automaticky rozbalily a hlavně ve chvíli, kdy si rozbalí položku, která obsahuje větší počet položek nižší úrovně, tak musí scrollovat na svém zařízení kvůli menšímu displeji.

Ale ne vždy platí, že rozbalovací menu je něco, co se nemá používat. V případě, že menu obsahuje velké množství položek, je to například rozsáhlejší webová stránka, se tzv. „mega rozbalovací menu“ dá efektivně využít k logickému členění položek na jednotlivé sekce a za dalšího využití ikonek a tooltipů může napomoci uživateli se lépe orientovat.[32] Avšak v případě systému na podporu DBS se rozbalovacímu menu vyhneme, protože tento systém nedosahuje takové velikosti, aby se vyplatilo jej použít. Hlavní menu jsem tedy navrhl tak, aby bylo pouze jednoúrovňové.



Obrázek 5.1: Menu – Semestrální práce – z pohledu cvičícího / garanta předmětu

5.3.2 Počet položek aneb někdy méně znamená více

Pokud přijde na abstraktní myšlení, tak lidský mozek je velice omezený. Podle [33] je lidská krátkodobá paměť schopna udržet pouze přibližně 7 věcí na maximálně 20 vteřin. To ale automaticky neznamená, že by každé menu mělo striktně obsahovat maximálně 7 položek. Mohou existovat i delší menu, protože uživatelé si nemusí pamatovat celé menu jako seznam. Avšak pokud je to možné, tak je lepší, aby menu obsahovalo méně položek a jejich počet byl orientačně maximálně okolo sedmi.

Pozor si ale musíme dát na extrém, kdy bychom se naopak snažili počet položek co nejvíce redukovat. Pokud bude menu příliš krátké, tak by se to mohlo odrazit v příliš abstraktních a nic nevypovídajících položkách menu.

Z těchto poznatků tedy plyne, že počet položek menu pro malé a středně velké webové stránky, by měl být přibližně mezi 3 až 7 položkami.

5.3.3 Horizontální a vertikální menu

Menu jsem navrhl dvouúrovňové. První, hlavní úroveň menu je horizontální a v případě přihlášeného vyučujícího se skládá z položek: Domů, Sem. práce, Testování, Studenti a Nastavení .

K tomu, abych se vyhnul rozbalovacímu menu jsem využil postranní menu, které je již implementováno jako postranní panel v šabloně AdminLTE. Toto postranní menu se vždy vykreslí v závislosti na tom, v jaké z výše uvedených hlavních sekcí se bude uživatel právě nacházet. Například když bude cvičící v sekci Sem. práce, tak se v postranním menu zobrazí položky druhé úrovně: Odevzdané práce, Znovuodevzdání, Termíny odevzdání, Hodnocení.

Navíc platí, že položka první úrovně odkáže uživatele na stejné místo, jako první položka druhé úrovně. Například při kliknutí na položku první úrovně Sem. práce je cvičící automaticky přesměrován na Odevzdané práce, které jsou první položkou ve druhé úrovni, ve vertikálním menu.

V případě, že se uživatel bude nacházet v sekci, kde není potřeba menu druhé úrovně, tak se postranní panel automaticky schová. Na obrázku 5.6 můžete vidět návrh menu pro sekci Sem. práce/Odevzdané práce.

5.3.4 Řazení položek

Řazení položek v našem případě rozhodně nesmí být podle abecedy. Ani obecně se nedoporučuje řadit položky v menu abcedně. Uživatelé zpravidla nevědí, co přesně hledají, či jak přesně se hledaná položka jmenuje.

Položky jsem tedy seřadil podle frekvence jejich předpokládaného použití a také podle jejich významu. Při návrhu seřazení položek menu je také vhodné vycházet ze zažitých konvencí mezi uživateli, kdy například tlačítko „Domů“ očekávají zpravidla na začátku a tlačítko „Kontakty“ očekávají zase naopak na konci menu.

Řazení položek menu ale také nesmí být podle jejich důležitosti vzestupné či sestupné, nýbrž nejdůležitější položky jsou na začátku a na konci a uprostřed jsou ty nejméně používané položky. Zde se totiž uplatňuje tzv. serial position effect.

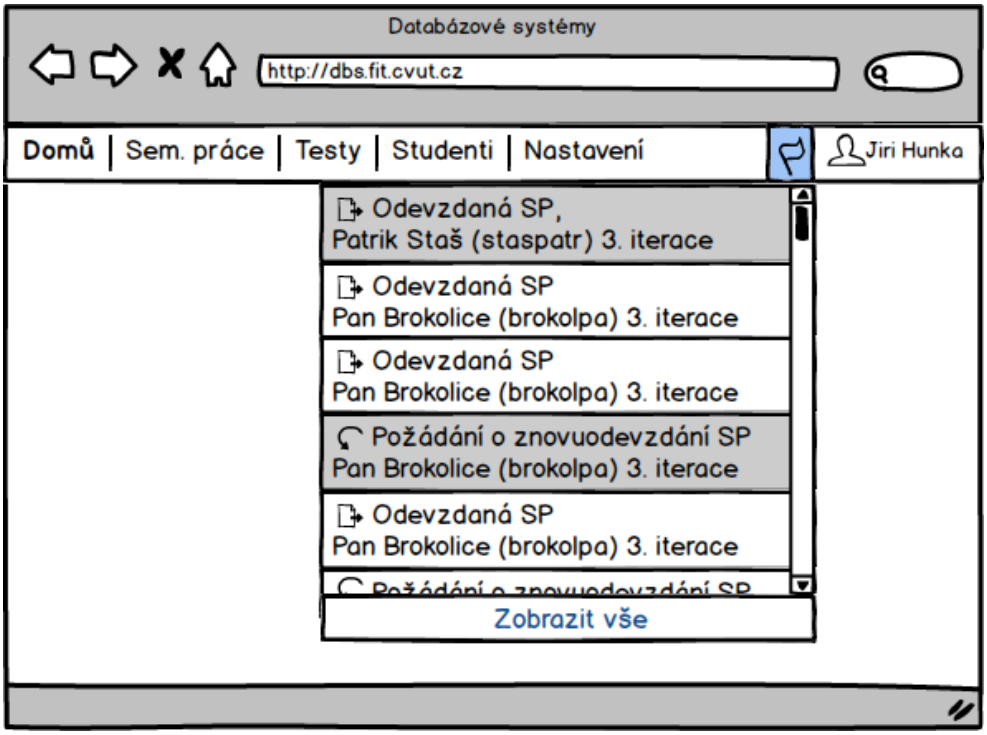
5.4 Domovská stránka

Domovská stránka je zpravidla nejdůležitější stránkou, protože je to první, co uživatel uvidí. V komerční sféře je o to důležitější, protože rozhoduje o tom, jestli potenciální zákazník bude dále pokračovat na stránkách společnosti, nebo odejde.

Na to, jak správně vytvořit domovskou stránku se lze inspirovat například ze seznamu 113 bodů sestaveného Jakobem Nielsenem a Marií Tahirovou ([34]). Uvedu zde pár vybraných bodů, které jsem využil při návrhu domovské stránky:

1. Je vhodné začít domovskou stránku jednou stručnou větou, ve které je popsáno, čím se tento web zabývá a co umožňuje
2. Rozhodně nezačínat úvodní stránku nadpisem „Vítejte ...“ aj.
3. Domovská stránka by měla nabízet okamžitou cestu k nejčastějším úkonům, které mají uživatelé vykonávat
4. Využít významovou grafiku, nepoužívat grafiku pouze pro dekorativní účely (typickým příkladem může být použití ikonek)

Dále domovská stránka zobrazí krátký přehled, statistiky (kolik studentů cvičícímu odevzdalo SP a kolik mu jich ještě zbývá na opravení, kolik % testů má již cvičící opraveno, atd.) a aktuální informace (např. za jak dlouho skončí odevzdávání do nejbližší iterace).



Obrázek 5.2: Notifikace – cvičící

5.5 Notifikace

Notifikace (upozornění) jsou jednoduchým, ale efektivním způsobem, jak uživatele informovat o akcích, které jsou pro něj důležité. Systém notifikací je navrhnut tak, že každá notifikace obsahuje informaci, kdy byla vytvořena, kým byla vytvořena, co je jejím obsahem, komu byla určena (jedna notifikace může být zasláno více uživatelům) a typ notifikace.

Notifikace může obsahovat akci, která se spustí při kliknutí na tuto notifikaci. Ve chvíli, kdy bylo na notifikaci kliknuto, tak změní barvu, aby se vizuálně odlišila od nezobrazených notifikací.

Pro druhý prototyp jsou navržené notifikace upozorňující na níže uvedené procesy v rámci nástroje na podporu SP:

- **Odevzdání SP**
  - **Popis** – jakmile student sám odevzdá SP, tak se jeho cvičícímu zobrazí notifikace, že mu byla odevzdána SP, která je připravena k opravě
  - **Akce** – kliknutím na notifikaci je cvičící přesměrován na tabulku odevzdaných SP a automaticky se mu předfiltruje tato odevzdaná SP

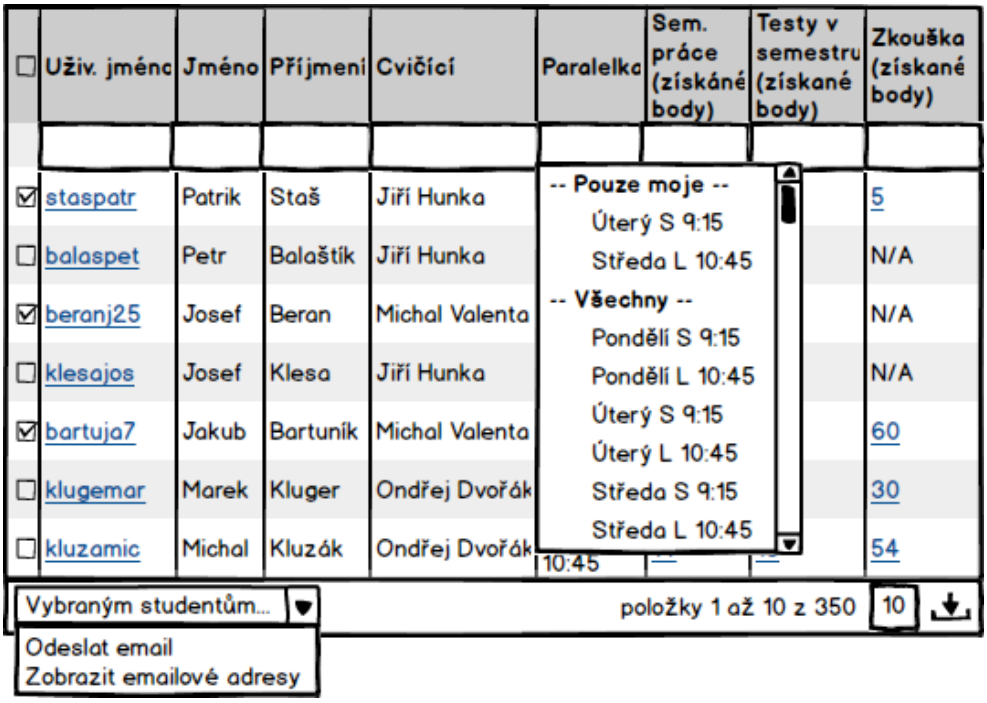
- **Automatické odevzdání SP**
  - **Popis** – pokud student v dané iteraci sám neodevzdá ani jednu SP, tak se po skončení termínu odevzdání automaticky odevzdá jeho aktuálně rozpracovaná SP. Tato notifikace se zobrazí studentovi a jeho cvičícímu.
  - **Akce** – kliknutím na tuto notifikaci bude student přesměrován na své odevzdané SP, kde bude předvyfiltrována tato automaticky odevzdaná SP
- **Opravení SP**
  - **Popis** – jakmile cvičící dokončí opravu odevzdané SP, tak se tato notifikace zobrazí studentovi této SP.
  - **Akce** – kliknutím na tuto notifikaci bude student přesměrován na své odevzdané SP s touto předvyfiltrovanou SP
- **Požádání o povolení znovuodevzdání SP**
  - **Popis** – ve chvíli, kdy student požádá svého cvičícího o povolení znovu odevzdat SP, tak se tato notifikace zobrazí cvičícímu
  - **Akce** – kliknutím na tuto notifikaci bude cvičící přesměrován na dialog povolení / zamítnutí znovuodevzdání (viz 5.12)
- **Povolení / zamítnutí znovuodevzdání SP**
  - **Popis** – studentovi se zobrazí upozornění na výsledek rozhodnutí cvičícího, jestli mu povolil nebo zamítnul žádost o znovuodevzdání
  - **Akce** – studentovi bude po kliknutí na tuto notifikaci zobrazen detail jeho zpracované žádosti, kde bude moci zjistit důvod povolení / zamítnutí a případného nového termínu pro odevzdání

V budoucnu lze přidat další notifikace. Mezi ně budou patřit notifikace z části testování, které budou cvičícímu oznamovat, že má k dispozici testy na opravu a studentovi budou oznamovat, že mu cvičící opravil test.

Notifikace o odevzdaných SP (a v budoucnu i notifikace o odevzdaných testech) budou mít svoji vlastní sekci na zobrazení. Předejde se tak zahlcení všech ostatních notifikací ve chvíli, kdy studenti začnou odevzdávat své SP.

5.6 Studenti

V 1. prototypu byla sekce studentů rozdělena na dvě části - Moji studenti a Všichni studenti. Avšak tento způsob členění byl poněkud nešťastným, protože se systém kvůli každé podobné drobnosti stává složitějším a uživatelé se v něm pak hůře orientují.



Obrázek 5.3: Tabulka studentů – použití filtru na paralelky

Logickým vyústěním bylo sloučení těchto dvou sekcí pouze do jedné. Vznikla tedy pouze jedna tabulka studentů, ale aby zůstala zachována možnost rozdělení studentů na ty, kteří patří pod daného cvičícího a na ostatní, tak se přidal do této tabulky atribut Cvičící a k němu filtr, takže se studenti dají filtrovat podle svých cvičících. Navíc ve výchozím nastavení se cvičícímu vyfiltrují právě jeho studenti.

Dále se do tabulky studentů přidal i atribut Paralelka a k němu dvouúrovňový filtr, pomocí kterého lze vyfiltrovat pouze své paralelky, všechny paralelky nebo jednotlivé paralelky.

Tabulka studentů kromě filtrování podle jednotlivých atributů umožňuje spoustu dalších akcí:

- Kliknutím na uživatelské jméno studenta je cvičící je přesměrován na Detail studenta (viz sekce 5.7)
- Kliknutím na počet získaných bodů u studenta za Semestrální práci je cvičící přesměrován na tabulku odevzdaných SP, kde má předvyfiltrované opravené SP studenta (podobně to bude funguvat při kliknutí na počet získaných bodů za test v semestru a při kliknutí na získané body za zkoušku)
- Cvičící může označovat jednotlivé řádky a v levém dolním rohu má možnost těmto vybraným studentům odeslat email (spustí se výchozí emai-

lový klient a předvyplní se pole adresátů emailovými adresami vybraných studentů) nebo pouze zobrazit emailové adresy vybraných studentů v textové podobě

- Cvičící si může exportovat obsah tabulky do souboru formátu CSV

K původnímu návrhu tabulky se ještě později přidaly další sloupečky: Celkem získaných bodů, Nárok na zápočet a Návrh známky. Atributy Nárok na zápočet a Návrh na známku nemají suplovat nebo nahrazovat funkci eduxu, v současném návrhu se předpokládá, že mají pouze informativní charakter.

Posledním dodatečným návrhem v tabulce studentů bylo odebrání sloupečku Email a jeho náhradou je přidání sloupečku Akce, ve kterém je ikonka pro odeslání emailu. Kliknutím na tuto ikonku se odešle email jednomu vybranému studentovi. Toto opatření bylo navrženo z důvodu úspory místa, protože tabulka studentů představuje nejen seznam všech studentů, ale i jejich výsledků a rozrostla se tak o notný kus v horizontální části.

5.7 Detail studenta

Sekce Detail studenta umožňuje cvičícímu nahlédnout na všechny informace jednoho studenta v daném kurzu.

Cvičící zde nalezne především seznam všech odevzdaných SP daného studenta. Rozdíl oproti tabulce Odevzdaných SP všech studentů (viz sekce 5.10) je v tom, že zde jsou vidět úplně všechna odevzdání. Tedy i ta, která nebyla a už nebudou opravena, protože byla nahrazena novějšími odevzdáními ještě před opravením. Cvičící má tedy možnost nahlédnout na historii studentovy práce.

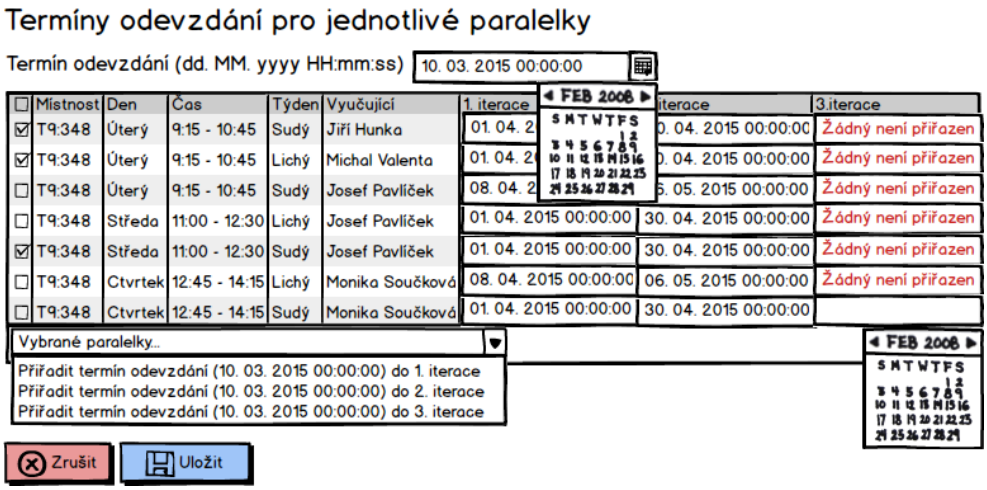
Cvičící má v detailu studenta k dispozici odkaz na náhled na studentovu aktuálně rozpracovanou SP. Díky tomu může pozorovat, co má již student vytvořeno.

V detailu studenta se počítá i s přidáním seznamu všech studentových testů, které vypracoval, avšak ve svém návrhu jsem je blíže nespecifikoval, protože toto je úkolem druhého týmu, který se zabývá testovacím prostředím.

5.8 Termíny odevzdání SP

V 1. prototypu bylo přiřazování termínů odevzdání rozděleno na každou iteraci. Uživatel si tedy nejprve musel vybrat konkrétní iteraci a terpve poté si mohl prohlížet termíny odevzdání pro jednotlivé paralelky. Ve 2. prototypu se toto sjednotilo do jedné tabulky a tento krok by měl tedy v konečném důsledku vést ke zjednodušení uživatelského rozhraní.

Jednotlivé řádky tabulky, představující termíny odevzdání, jsou paralelky. Pro lepší filtrování jsou vytvořeny jako jednotlivé sloupečky všechny parame-



Obrázek 5.4: Termíny odevzdání SP – editace garantem předmětu

try paralelky: Místnost, Den, Čas (začátek až konec cvičení), Týden (sudý / lichý) a Cvičící.

Všichni účastnící kurzu mohou nahlédnout na termíny odevzdání pro všechny paralelky, ale pouze garant může editovat toto nastavení.

V původním návrhu pro 2. prototyp (viz obrázek 5.4) jsem uvažoval možnost inline editace. Garant by tak mohl přímo v tabulce editovat jednotlivé termíny. Komponenta Grido sice umožňuje inline editaci, ale bohužel v tomto případě nešlo inline editaci jednoduše implementovat a vzhledem k tomu, že nastavení iterací proběhne přibližně pouze jednou či dvakrát do roka, tak nemělo smysl věnovat čas pro implementaci této funkčnosti.

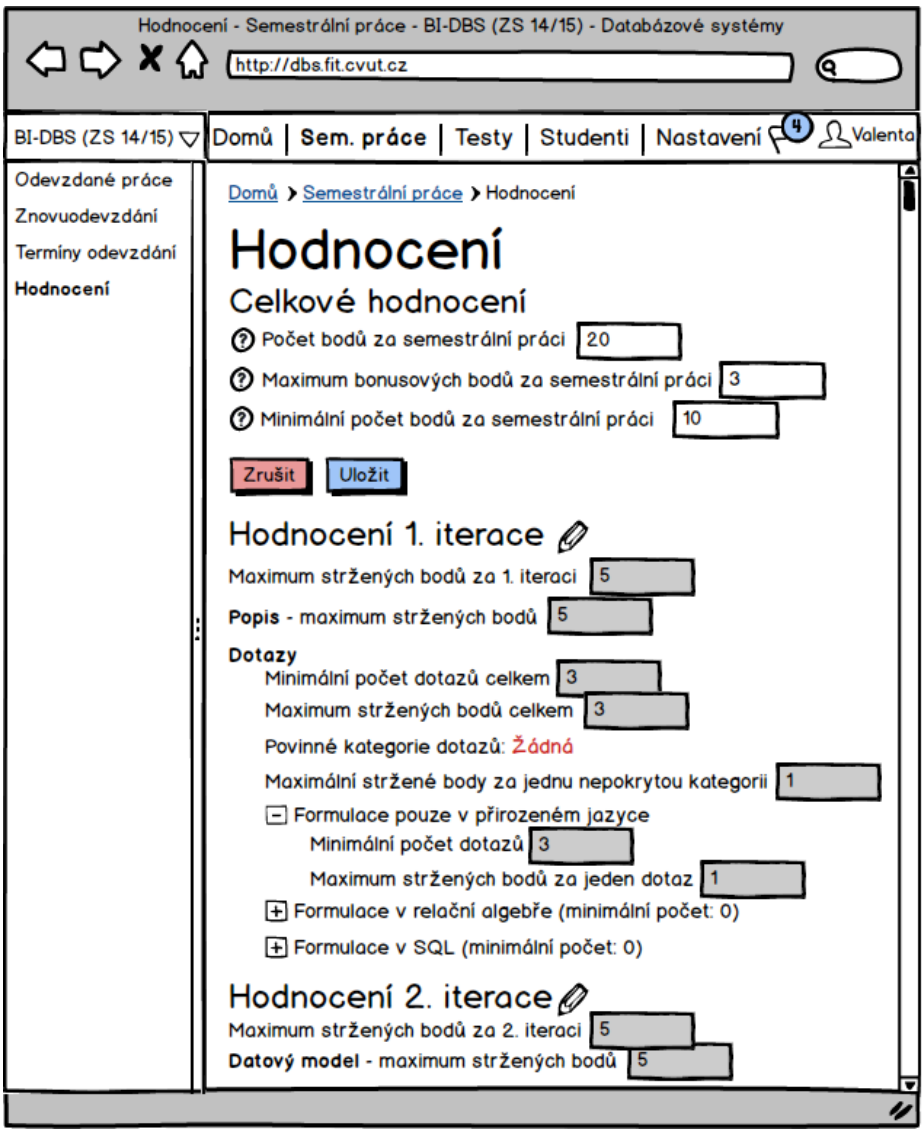
Garant má ale samozřejmě možnost nastavit termíny odevzdání i jiným způsobem. Stejně jako v 1. prototypu si vybere pomocí označování řádků v prvním sloupečku jednotlivé paralelky, zvolí konkrétní datum a čas (k tomu může využít i bootstrap komponentu date picker) a určí, k jaké iteraci se má tento termín vázat.

5.9 Hodnocení SP

5.9.1 Fixní počet iterací

Hodnocení SP prošlo od 1. prototypu několika zásadními změnami. Jednou z nich je zrušení možnosti přidávat další iterace. Počet iterací se stanovil jako fixní na 3 iterace. Důvodem k tomuto kroku je složitost realizace a ani do budoucna se nepočítá se změnou celého konceptu opravování.

Podle nového návrhu je implementována možnost, aby si garant u každé z těchto tří iterací nastavil bodové hodnocení pro jednotlivé kontrolované části a také si bude moci nastavit podrobně kontrolu dotazů v každé iteraci.



Obrázek 5.5: Hodnocení SP - ukázka editace Celkového hodnocení

5.9.2 Doporučené maximální stržené body

Ve 2. prototypu se přidala pro garanta možnost nastavit doporučené maximální stržené body nejen pro celou iteraci, ale i pro každou její kontrolovanou část. Tuto nastavenou hodnotu tedy může cvičící při opravování SP překročit. Tímto jsem chtěl jednak dát cvičícímu možnost co největší flexibility při opravování, ale zároveň se tak vyřešil problém při automatické opravě, kdy by SP obsahovala takový počet chyb, že by přesáhla toto maximum, a pak by byl problém určit, jak nepřekročit stanovené maximum.

5.9.3 Editace jednotlivých částí

Všichni účastníci kurzu (studenti a vyučující) mají možnost nahlédnout na hodnocení SP. Garant má ale dále možnost změnit toto hodnocení. Jak lze vidět podle obrázku 5.5 tak garant může editovat jednotlivé části hodnocení: Celkové hodnocení, 1. iterace, 2. iterace a 3. iterace.

5.9.4 Nastavení dotazů

Zvláštní prostor při návrhu hodnocení byl věnován nastavení dotazů. Díky novému provedení přiřazení požadavků k iteracím lze jeden požadavek kontrolovat v několika iteracích různým způsobem. Toto bylo realizováno především kvůli kontrole dotazů.

U dotazů se dají ve 2. prototypu pro každou iteraci nastavit tyto podmínky:

1. Minimální počet dotazů a doporučený počet stržených bodů za jeden chybějící dotaz[36]

2. Povinné kategorie dotazu a doporučený počet stržených bodů za jednu nepokrytou kategorii

3. Minimální počet dotazů v RA a doporučený počet stržených bodů za jeden dotaz v RA

4. Minimální počet dotazů v SQL a doporučený počet stržených bodů za jeden dotaz v SQL

Díky tomu má garant nástroj, pomocí kterého může změnit kontrolu dotazů v každé iteraci. V případě, že nechce, aby se například v 1. a 2. iteraci nekontrolovaly dotazy v RA či v SQL, tak stačí nastavit jejich minimální počet na 0.

[36]Dotazem se obecně myslí objekt, který se skládá ze tří komponent: Slovní popis (formulace v přirozeném jazyce), formulace v relační algebře, formulace v jazyku SQL.

5.10 Odevzdané SP

Ve 2. prototypu se upustilo od rozdělení odevzdaných SP na ty, které jsou připravené k opravě a na ty, které jsou již opravené. Teď jsou odevzdané SP v jedné přehledné tabulce (viz obrázek 5.6).

Filtrování SP podle iterací a paralelek pomocí tlačítek (viz příloha – obrázek C.3) se zrušilo a místo toho přibyly jako nové sloupečky. Dále se díky přidání atributu Paralelka zrušily její jednotlivé atributy jako sloupečky (den, čas, místnost a týden) a přidaly se atributy Cvičící, Stav (Opravena / Neopravena).

Díky použitým filtrům si mohou cvičící jednoduše vyfiltrovat jimi hledané SP. Ve výchozím nastavení se cvičícímu předfiltrují odevzdané SP pouze od jeho studentů a seřadí se nejprve podle stavu (neopravené SP se zobrazí před těmi již opravenými) a poté podle času odevzdání (SP, která byla odevzdána jako první se také zobrazí jako první).

Ve sloupečku akce se navíc změnily slovní popisky akcí na ikonky (při najetí myši na ikonku se zobrazí její popis, popřípadě je vysvětlení ikonek ještě v postranním menu v části navigace). Díky tomu se uštřilo místo a navíc je pro cvičící intuitivnější, co která ikonka provede za akci.

5.11 Oprava SP

5.11.1 Uzamčení opravování

Ve 2. prototypu byla nově realizována funkce uzamykání opravy SP. Jakmile cvičící začne opravu, tak nejen, že studentovi uzamkne odevzdávání, ale zároveň se uzamkne oprava i před ostatními cvičícími, aby dva cvičící neopravovali v jednu chvíli jednu SP.

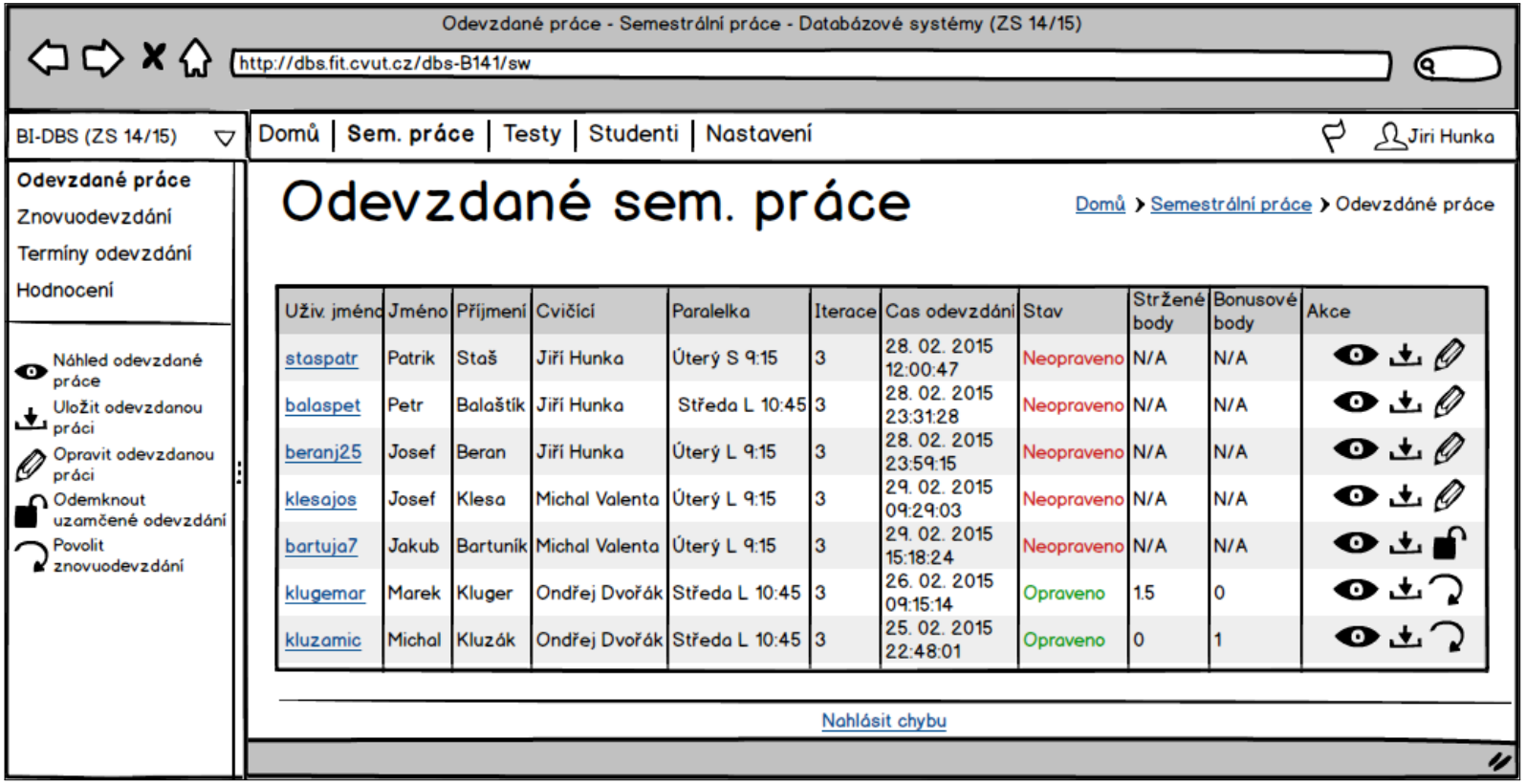
Cvičící, který si uzamknul opravu SP ji může opět odemknout pro ostatní. V případě, že by sám cvičící z nějakého důvodu nemohl odemknout opravu bude mít ještě garant předmětu možnost odemknout opravu SP.

5.11.2 Položka opravy jako komponenta

Z pohledu UX došlo k jednomu z největších posunů oproti 1. prototypu u opravování SP. Všechny kontrolovatelné položky jsou navržené jako komponenty skládající se ze dvou částí: Obsah kontrolované položky a editovatelná část, ve které se dají strhávat body a přidávat komentáře.[37]

Obsah kontrolované položky zůstává stále stejný. Buďto je jím jakýkoli text (ať je to textový popis, nebo dotaz v RA či v SQL) nebo obrázek. U SQL dotazů se navíc přidalo zvýraznění syntaxe. U dotazů v relační algebře se také předpokládá přidání zvýraznění syntaxe, ale to bude moci být uskutečněno až ve chvíli, kdy bude hotový syntaktický analyzátor nástroje RAT.

[37] Ukázka komponenty dotaz viz obrázek 5.7



Uživ. jméno	Jméno	Příjmení	Cvičící	Paralelka	Iterace	Cas odevzdání	Stav	Stržené body	Bonusové body	Akce
staspatr	Patrik	Staš	Jiří Hunka	Úterý S 9:15	3	28. 02. 2015 12:00:47	Neopraveno	N/A	N/A	
balaspet	Petr	Balaštík	Jiří Hunka	Středa L 10:45	3	28. 02. 2015 23:31:28	Neopraveno	N/A	N/A	
beranj25	Josef	Beran	Jiří Hunka	Úterý L 9:15	3	28. 02. 2015 23:59:15	Neopraveno	N/A	N/A	
klesajos	Josef	Klesa	Michal Valenta	Úterý L 9:15	3	29. 02. 2015 09:29:03	Neopraveno	N/A	N/A	
bartuja7	Jakub	Bartuník	Michal Valenta	Úterý L 9:15	3	29. 02. 2015 15:18:24	Neopraveno	N/A	N/A	
klugemar	Marek	Kluger	Ondřej Dvořák	Středa L 10:45	3	26. 02. 2015 09:15:14	Opraveno	1.5	0	
kluzamic	Michal	Kluzák	Ondřej Dvořák	Středa L 10:45	3	25. 02. 2015 22:48:01	Opraveno	0	1	

Obrázek 5.6: Přehled studentů - filtrování podle paralelek a příslušnosti k přihlášenému vyučujícímu

5.11.3 Stržení bodů a přidání komentáře

Stejně jako u 1. prototypu tak i ve 2. prototypu zůstalo, že pokud je během automatické opravy nalezena chyba, tak se u dané položky strhne doporučený maximální počet bodů a do komentáře se přidá popisek chyby.

Pokud nebyly strženy body během automatické opravy, tak je část pro strhávání bodů a přidávání komentáře zabalená, protože se předpokládá, že větší část opravovaných položek bude správně i po faktické stránce a cvičící jim tedy nebude strhávat žádné body, či přidávat komentáře. Tím, že schováme editační část položky, bude pro cvičícího opravování přehlednější.

Naopak ve chvíli, kdy byla nalezena alespoň nějaká chyba, tak se část pro editaci stržených bodů a komentářů rozbalí, protože se předpokládá, že cvičící při opravě bude chtít přidat svůj vlastní komentář, či změnit stržené body.

U ručního strhávání bodů cvičícím se přidala 4 pomocná tlačítka, aby cvičící nemusel pokaždé dávat ruku pryč z myši na klávesnici. Tato tlačítka obsahují hodnoty: 0, -0.5, +0.5 a doporučené maximum. Cvičící má možnost překročit doporučené maximum.

5.11.4 Barevné značení

Každá položka opravy má určenu svoji barvu podle následující definice:

Šedivá - opraveno v jedné z předchozích iterací, nelze editovat

Zelená - prošlo v pořádku automatickou kontrolou a nebyly stržené žádné body

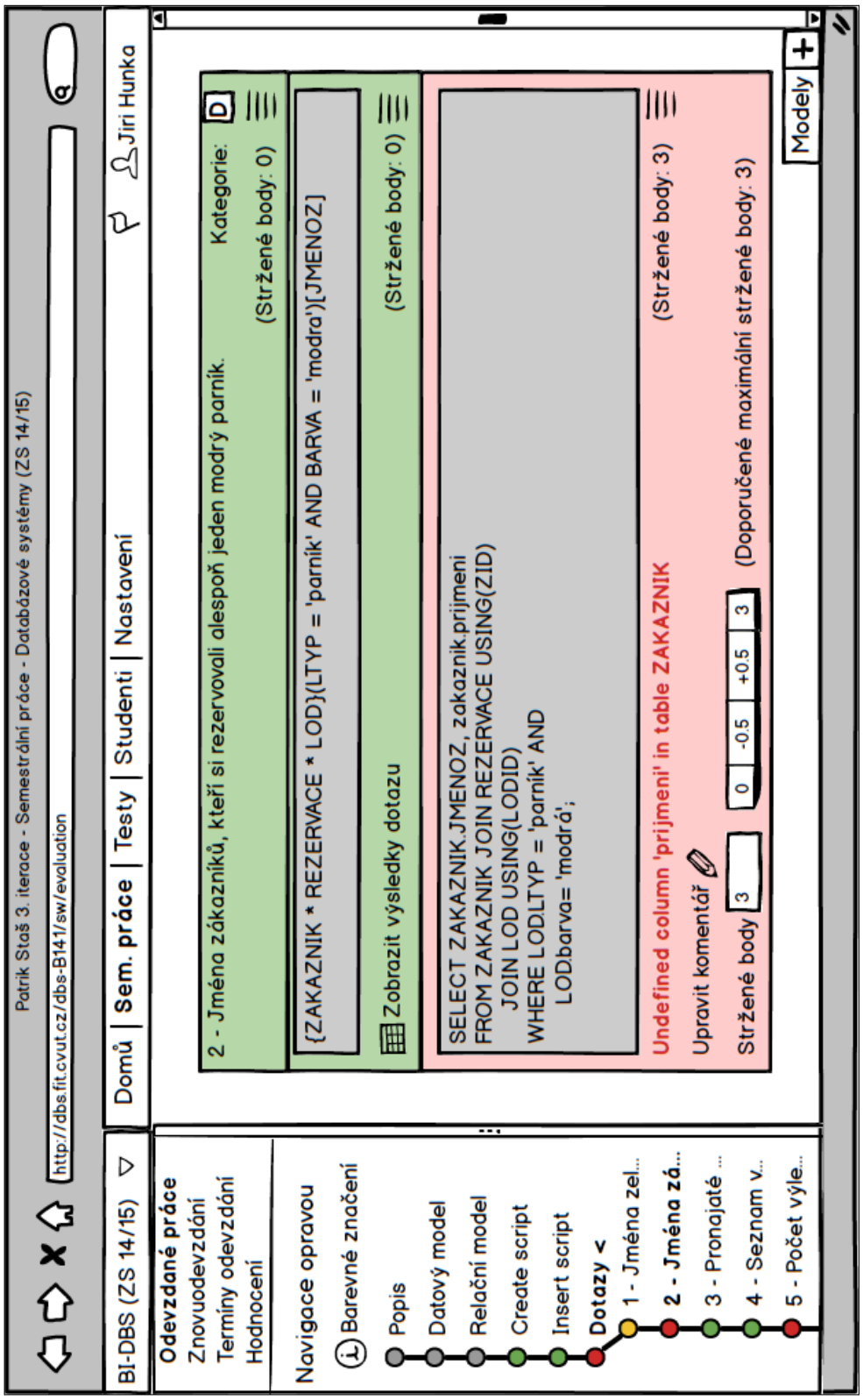
Červená - byla nalezena chyba během automatické kontroly a/nebo byly strženy nějaké body

Oranžová - zvyšte pozornost při opravování. Používá se například při znovu odevzdání, kdy v minulém odevzdání byla nalezena chyba a v novém, opakovaném odevzdání, již žádná chyba nebyla nalezena. Cvičící se tak při opravě může rovnou zaměřit na ty položky, které měl student opravit. Druhý případ použití je u dotazů, které mají od studenta manuálně přiřazenou kategorii.

Během procesu opravy se tyto barvy mohou měnit. Typickém příkladem je, když cvičící strhne bod u nějaké položky, která prošla v pořádku autoamtickou kontrolou. Tato položka tak změní barvu ze zelené na červenou.

5.11.5 Postranní navigace

Postranní navigace má sloužit k větší přehlednosti při zobrazení SP. Uplatní se zejména při opravování ve 3. iteraci, kdy je obsah SP rozsáhlý oproti předchozím dvěma iteracím.



Obrázek 5.7: Oprava SP ve 3. iteraci

U postranní navigace se také barevně označují jednotlivé položky opravy. Cvičící tak nemusí procházet celou semestrální práci, aby našel položky, které vyžadují jeho zvýšenou pozornost (například oranžové položky při opakovaném odevzdání, nebo chybné červené položky).

Postranní navigace také zvýrazní prvek, u kterého se aktuálně cvičící nachází a kliknutím na prvek v navigaci se cvičící k tomuto prvku okamžitě přesune.

5.11.6 Ukončení opravy – uspořádání tlačítek

V novém návrhu opravování SP se změnilo postavení tlačítek na přerušení nebo ukončení opravy.

V původním návrhu pro 1. iteraci byla všechna tlačítka společně na levé straně a byla seřazena zleva doprava od kladného k zápornému (viz obrázek C.5 v příloze). V novém návrhu jsem toto pořadí otočil, záporné tlačítko umístil úplně vlevo a kladná tlačítka úplně vpravo.

Oba dva způsoby řazení mají svoji logiku.

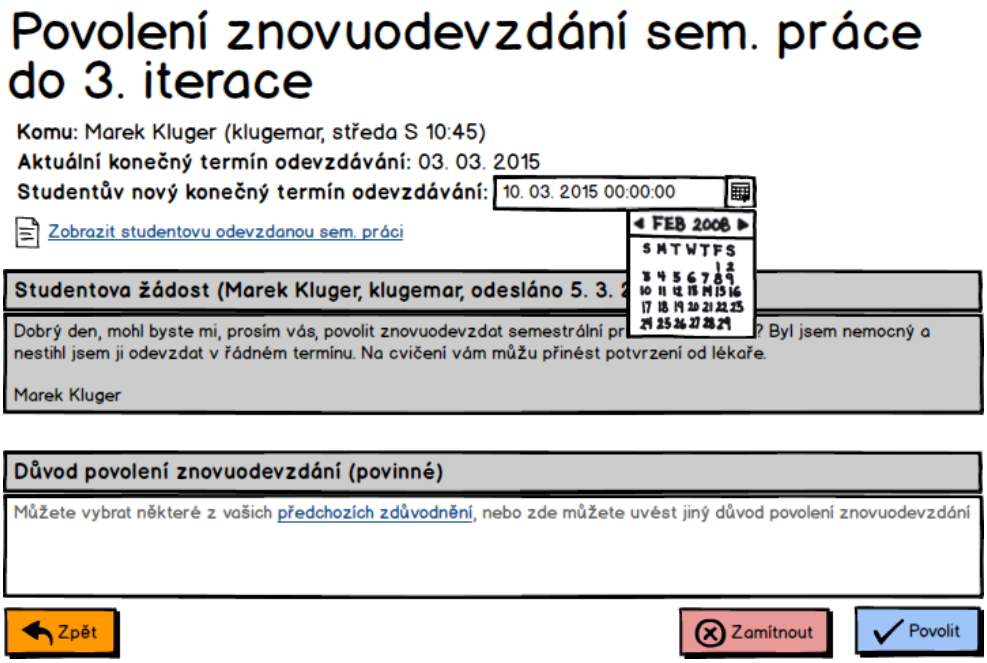
- Postavení tlačítek od kladného k zápornému zleva doprava vychází z myšlenky, že lidé v západním světě čtou zleva doprava a vždy začínáme kladnou odpovědí a končíme zápornou (říkáme například „Ano / Ne“, ne naopak). Navíc uživatelé častěji volí kladnou odpověď místo záporně, tedy při opravě SP ji cvičící zpravidla dokončí a nepřeruší (tedy zvolí kladnou možnost „Dokončit opravu“) a nemusí tak při výběru možností pokračovat čtením dalších možností. Tento způsob pořadí tlačítek je typický pro systémy Windows.

- Na druhou stranu způsob postavení tlačítek od záporného ke kladnému podporuje logiku průchodu, protože záporná odpověď nás vrátí zpět doleva a výběrem kladné odpovědi pokračujeme doprava. V uživateli by to tedy mělo evokovat funkci tlačítek Zpět a Další. Tento způsob použití je typický pro systémy iOS.

Pro druhý způsob postavení tlačítek jsem se rozhodl z důvodu, že při opravě SP se tlačítko na přerušení opravy jmenuje Zpět a není tedy podle mě intuitivní, aby toto tlačítko bylo vpravo[38].

Podle Jakoba Nielsena ([35]) v podstatě nezáleží na tom, jaké pořadí zvolíme, ale musíme dodržet, aby toto pořadí bylo dodrženo všude v celém systému. Musíme tedy zachovat konzistenci.

[38] Něco jiného by bylo, kdyby se tlačítko Zpět jmenovalo například Zrušit, pak by se dalo uvažovat o druhém způsobu pořadí tlačítek.



Obrázek 5.8: Dialog povolení znovuodevzdání SP

5.12 Znovu odevzdání SP

V 1. prototypu mohl být proces znovuodevzdání iniciován pouze učitelem a chyběla podpora ve studentském rozhrání. Podle návrhu pro 2. prototyp má teď student nově možnost požádat svého cvičícího o znovuodevzdání pomocí systému notifikací (viz sekce 5.5). Při podání žádosti musí student uvést důvod, proč žádá o dodatečné povolení.

Cvičícímu tedy může buďto přijít žádost od studenta, nebo může povolit znovuodevzdání ze své vůle (například při předčasném odevzdání chce studentovi umožnit, aby mohl opravit chyby, které mu našel při opravě). Nově také přibyla možnost zamítnout studentovu žádost o znovuodevzdání.

V novém návrhu musí navíc cvičící specifikovat nový termín odevzdání. Nový termín odevzdání bude přednastaven o 7 dní později, než je aktuální termín odevzdávání, ale cvičící jej může samozřejmě změnit.

Dále přibyla v dialogu znovuodevzdání bližší specifikace ake, kterou cvičící provádí, možnost nechat si zobrazit SP, kterou opravil studentovi, aby ji nemusel zbytečně vyhledávat mezi opravenými SP a při vyplnění důvodu znovuodevzdání bude mít k dispozici výběr svých předchozích zdůvodnění.

Podle uživatelského testování 1. prototypu měli cvičící problém vůbec najít, kde mohou povolit znovuodevzdání SP. Tento problém se tedy vyřešil jednak přes systém notifikací a jednak přibyla celá jedna sekce v nástroji SP, která se jmenuje Znovuodevzdání.

V sekci Znovuodevzdání bude mít cvičící možnost vybrat si jakéhokoli studenta (přednostně budou seřazeni na začátku jeho studenti) a u něj bude vidět ve sloupečcích 1. iterace, 2. iterace a 3. iterace aktuální stav znovuodevzdání jeho SP a případné akce:

- Prázdné - není co povolovat, protože student nemá opravenou žádnou SP ve vybrané iteraci
- Opravená SP + akce Povolit znovuodevzdání - v případě, že chce studentovi povolit znovuodevzdání, ikdyž jej o to student nepožádal a přesune cvičícího do dialogového okna na povolení znovuodevzdání (toto dialogové okno bude samozřejmě ochuzené o zdůvodnění studenta a možnost zamítnout oproti dialogovému oknu z obrázku 5.8)
- Opravená SP + akce Zpracovat žádost - přesune cvičícího do dialogového okna z obrázku 5.8
- Zamítnuto + akce Povolit znovuodevzdání - cvičící může i po zamítnutí povolit znovuodevzdání SP
- Povoleno

5.13 Oprava znovuodevzdané SP

Opravování znovu odevzdané SP ve stejné iteraci vychází z konceptu nového způsobu opravování. Navíc se u částí, které to vyžadují, zobrazí i staré odevzdání, využije se barevné značení jednotlivých částí opravy a k dispozici bude i náhled na provedené změny.

- Ty části, kterým nebyl v minulém odevzdání stržen žádný bod a nebyl jim přidán ani žádný komentář a zároveň prošla úspěšně automatickou kontrolou a zároveň se v novém odevzdání u těchto částí nic nezměnilo, tak tyto části se označí zelenou barvou.
- Pokud byly dané části v minulém odevzdání strženy nějaké body a/nebo jí byl přidán komentář a v novém odevzdání prošla úspěšně automatickou opravou (nebyl jí stržen žádný bod během automatické opravy), tak taková část bude označena oranžovou barvou. Pro cvičícího je tak jasně označeno, které části opravy má projít pečlivě, protože zpravidla kvůli těmto částem student znovu odevzdával svoji SP.
- Pokud byly i v novém odevzdání některé části strženy body již při automatické opravě, tak se tato část označí červeně.

V případě, že došlo ke změně obsahu u dané kontrolované jednotky, tak se tato část nejen označí barevně, ale cvičící bude mít u textových položek k dispozici i vyznačené rozdíly mezi jednotlivými odevzdáními (využije se nástroj na vyznačení rozdílů po slovech a řádcích mezi dvěma textovými řetězci).

5.14 Uživatelské testování 2. prototypu

Uživatelské testování 2. prototypu se v principu vůbec nelišilo od uživatelského testování 1. prototypu, byla ta pouze další iterace.

Cílem tohoto testování bylo nalezení všech chyb v uživatelském rozhraní, tj. chyb, které nebyly nalezeny v minulém testování ale i nových chyb, které mohly vzniknout ve 2. prototypu.

5.14.1 Testovaní uživatelé

Na testování jsem opět vybral 4 cvičící předmětu Databázové systémy. 2 z testovaných uživatelů se zúčastnili předchozího testování a zbylí 2 uživatelé tento nově vzniklý nástroj testovali poprvé.

Výhodou tohoto rozložení testovaných uživatelů bylo, že ti, kteří nebyli ovlivněni testováním 1. prototypu, mohli nalézt chyby, které se nepodařilo objevit v přechozím testování. A u uživatelů, kteří byli testováni podruhé se dalo pozorovat zlepšení či zhoršení práce s tímto nástrojem.

5.14.2 Úkoly

Původní úkoly se téměř nezměnily a pouze přibyly úkoly na testování nových funkcí 2. prototypu:

1. Najděte, co se má v každé iteraci hodnotit, jaké jsou požadavky na hodnocení SP

2. Najděte, jaké jsou termíny odevzdání (deadliny) pro všechny vaše paralelky

3. Zobrazte si všechny odevzdané SP, i těch studentů, které nemáte na cvičení

4. Prohlédněte si libovolnou odevzdanou SP (nezáleží, jestli je opravená, nebo čeká na opravu)

5. Opravte SP z 1. iterace od studenta Jakub Truneček (truneja2)

    a) Vyzkoušejte si během opravy cokoli budete chtít (strhávání bodů, přidávání komentářů, uložení hodnocení, dočasné ukončení hodnocení a vrácení se zpět k opravě, využití postranní navigace aj.)

6. Zběžně opravte SP z 2. iterace od studenta Jakub Truneček (truneja2) (stačí si pouze prohlédnout jednotlivé položky opravy, neřešte jejich správnost)

7. Opravte SP studenta Jakub Truneček (truneja2) ze 3. iterace

a) Opravte řádně dotaz 1 (například zkontrolujte, že odpovídá struktuře databáze podle modelů, podívejte se, co vrací dotazy za hodnoty)

b) Prohlédněte si zběžně zbylé dotazy a ostatní části, které se mají opravovat a dokončete opravu

8. Studentu Jakub Truneček (truneja2) povolte znovuodevzdání SP do 3. iterace (aby mohl opravit chyby, které jste mu právě opravil/a)

9. Zběžně opravte znovuodevzdanou SP z 3. iterace od studenta Jakub Truneček (truneja2)

10. Odešlete email všem studentům, které máte na cvičeních (pouze sudentům z vašich paralelek)

5.14.3 Průběh testování

Testování opět probíhalo po předchozí domluvě s jednotlivými cvičícími na smluveném místě. K zaznamenání zvukové a video stopy testu jsem využil nástroj Screenpresso.

Jediná změna v průběhu testování oproti testování 1. prototypu byla v tom, že jsem již cvičícím nedával vyplnit dotazník, protože jak se ukázalo z předchozího testování, nepřínesl téměř žádnou přidanou hodnotu.

5.14.4 Výsledky testování

Ani jsem nečekal, že by to mohlo být tak dobré. Celé to na mě působí velice příjemně, hlavně se mi líbí menu a jak je u opravy semestrální práce v navigaci ihned vidět, co je špatně.

Ing. Pavel Krejčí, doktorand

Na úvod výsledků testování jsem si dovolil citovat jednoho z tetovaných cvičících během testování. Podobně pozitivně reagovali i další testovaní uživatelé, což svědčí o tom, že výsledný 2. prototyp splňuje naše očekávání a cvičícím jistě výrazně ulehčí kontrolu semestrálních prací.

Cvičící neměli téměř s žádným úkolem výrazný problém. V systému se dobře orientovali, nový design na ně působil příjemně a byli vždy schopni nalézt potřebné informace či provést požadovanou akci. Při testování samozřejmě narazili cvičící na některé problémy, což je ale naprosto pochopitelné, protože žádný prodkut není bezchybný a nám to pomůže doladit poslední chybičky.

Při opravování narazili cvičící na tyto problémy:

- Při strhávání bodů očekávali, že když budou strhávat body, tak by měli zadat záporné hodnoty, ale ve 2. protypu fungovalo strhávání bodů

v kladných hodnotách. Tento způsob strhávání bodů pro ně nebyl intuitivní.

- Při strhávání bodů a přidávání komentáře očekávali, že mají změněné hodnoty potvrdit a nevěděli tedy, jestli jimi provedené změny se opravdu uloží nebo ne.

- Při opravování ve 3. iteraci si nikdo z cvičících nevšiml v pravém dolním rohu tlačítka na zobrazení konceptuálního a relačního modelu.

Kromě těchto problémů ale pro ně byl nový způsob opravování jenom přínosem. Pomocí postranní navigace a barevného značení pro ně bylo opravování velcie přeheldné a díky schovávání nepříliš důležitých informací nebyli zahlceni spoustou zbytečných dat.

Dalším drobným kazem bylo nevýrazné zobrazení notifikací. Důsledkem tedy bylo, že si jich nikdo nevšiml a tím pádem je ani nevyužil, ale díky kvalitnímu návrhu celého systému to pro cvičící i tak nepředstavovalo žádný výrazný problém při plnění úkolů.

Při testování se narazilo na drobné problémy s Grido tabulkami

- Tlačítko pro výběr všech řádků v tabulce fungovalo na principu inverze (tzn. že vždy označilo nevybrané řádky, ne všechny řádky), což bylo pro uživatele neintuitivní

- Plně nefungovalo tlačítko reset

- Odeslání emailu více studentům - tato akce se měla provádět pomocí tabulky grido, ale nikoho to nenapadlo, a ikdyž pak byli nasměrováni na příslušnou tabulku studentů, tak stejně nebyli schopni přijít na způsob, jak poslat studentům hromadný email.

Jinak ale Grido tabulky byly opět velkým přínosem díky okamžitému filtrování (po vložení filtrované hodnoty nebylo nutné kliknout na tlačítko Hledat nebo Enter) a implementovanému case-insensitive vyhledávání.

Celkově byli cvičící nadmíru spokojeni s výsledným nástrojem a po úpravě posledních drobných chyb bude možné tento nástroj plně nasadit v následujícím běhu předmětu Databázové systémy.

Závěr

Cílem bakalářské práce bylo vytvoření nástroje na automatickou kontrolu semestrálních prací v předmětu Databázové systémy, který umožní cvičícím efektivní kontrolu semestrálních prací. Tento nástroj měl být ještě řádně uživatelsky otestován. Zadané cíle byly splněny a navíc bylo dosaženo mnoha dalších výsledků nad rámec zadání této bakalářské práce.

Výsledným produktem není pouze jeden samostatný program na kontrolu semestrálních prací, ale celý systém na podporu předmětu Databázové systémy, který ulehčí práci nejen vyučujícím tohoto předmětu, ale i studentům. Tento systém obsahuje jako jeden ze svých dílčích nástrojů právě nástroj na kontrolu semestrálních prací.

Díky zasazení nástroje na kontrolu semestrálních prací v systému na podporu celého předmětu, je možné využít při kontrole semestrálních prací i další podpůrné činnosti. Například garant předmětu může nastavit termíny odevzdání pro jednotlivé paralelky, či může nastavit bodové hodnocení jednotlivých částí semestrální práce.

K maximální efektivitě samotné kontroly semestrální práce přispívá celá řada funkcí. Nejdůležitější z nich je automatická kontrola dodržení termínů pro odevzdávání, automatická kontrola všech odevzdávaných částí semestrální práce a jejich předpřipravení k ruční opravě a zcela nové rozhraní pro ruční opravu odevzdané semestrální práce.

Pod pojmem automatické kontroly semestrální práce se skrývá spousta pojmů, největší přínos z nich má především oprava dotazů. U dotazů se kontroluje jejich minimální stanovený počet, pokrytí předem požadovaných kategorií (i určování kategorií dotazů je až na výjimky automatické), správnost syntaxe dotazů v relační algebře a v SQL, či se kontrolují vrácené hodnoty těchto dotazů. Navíc je implementována podpora pro opakované odevzdávání, takže je přesně vidět, co student změnil od minulého odevzdání. Cvičící se tak mohou při opravě semestrálních prací zaměřit na ty části, které vyžadují jejich největší pozornost.

Kromě vývoje nástroje na automatickou kontrolu jsem se podílel na návrhu

a vývoji téměř všech důležitých částí vytvořeného systému a během druhého roku vývoje tohoto systému jsem zastával funkci vedoucího nově nabraného týmu studentů.

Výsledný nástroj na kontrolu semestrálních prací, respektive celý systém, bude nasazen v testovacím režimu v nadcházejícím zimním semestru akademického roku 2015/2016 v předmětu Databázové systémy. Cílem bude nalézt a opravit zbylé chyby, protože se má tento nástroj naostro spustit v dalším běhu tohoto předmětu, v letním semestru akademického roku 2015/2016.

Nasazení a údržbu tohoto systému bude mít na starost tým studentů, který jsem vedl v rámci této bakalářské práce. Až tento tým skončí, tak se nabere nový tým studentů, který opět bude pokračovat v práci předchozích týmů. Nemělo by se tedy stát, že by se tento systém, a tím i nástroj na automatickou opravu semestrálních prací, nepoužíval.

Projekt na podporu předmětu Databázové systémy patří rozhodně mezi nejrozsáhlejí na Fakultě informačních technologií ČVUT v Praze. Bohužel i přes veškerou moji snahu a snahu mých předešlých a současných týmových kolegů, se nepodařilo dokončit nejen všechny části tohoto systému, ale i některé méně důležité části nástroje na podporu semestrálních prací právě z důvodu požadovaného rozsahu výsledného produktu.

U tohoto systému se v budoucnosti nepočítá s rozšířením podpory pro další předměty kromě několika výjimek. Tento systém sice umožňuje přidání dalších předmětů, ale v současné chvíli se reálně uvažuje o rozšíření podpory pouze pro předmět Jazyk SQL, kde se v podstatě pokračuje v práci započaté v předmětu Databázové systémy.

Problematika automatické opravy semestrálních prací, respektive problematika celého systému na podporu předmětu Databázové systémy, mě zaujala už v rámci předchozího týmového projektu. Myšlenka toho, že se tento nástroj bude skutečně používat, ušetří tak práci všem jeho uživatelům a já budu moci přispět k chodu své fakulty, mě motivovala k tomu, abych v jeho vývoji pokračoval i v této bakalářské práci. O tom, že mne tento projekt skutečně zaujal svědčí i fakt, že jsem nad ním strávil více než 700 hodin. A vzhledem k tomu, že je zde stále co rozvíjet, tak pokud to bude možné, rád bych pokračoval ve vývoji tohoto systému i ve své diplomové práci.

Literatura

[1] Brabenec, M.: *Analýza a kompletace webové aplikace RAT*. 2015, [cit. 2015-04-11].

[2] Semestrální práce: Popis a dokumentace. *BI-DBS - Databázové systémy* [online]. 2013, [cit. 2014-11-08]. Dostupné z: `https://edux.fit.cvut.cz/courses/BI-DBS/project/description`

[3] Kardinalita vztahu. *Informační technologie* [online]. 2009, [cit. 2014-11-17]. Dostupné z: `http://informacni-technologie.studentske.cz/2009/02/kardinalita-vztahu.html`

[4] *MoodleDocs* [online]. 2010, [cit. 2014-11-08]. Dostupné z: `https://docs.moodle.org/archive/cs`

[5] Modul Úkol. *MoodleDocs* [online]. 2007, [cit. 2014-11-08]. Dostupné z: `https://docs.moodle.org/archive/cs/Modul_%C3%9Akol`

[6] Klimperová, L.: *Automatizovaná kontrola semestrálních prací z DBS*. 2010, [cit. 2014-11-08]. Dostupné z: `https://dip.felk.cvut.cz/browse/pdfcache/klimpmar_2010bach.pdf`

[7] Mlejnek, J.: *Analýza a sběr požadavků* [online]. Praha. 2015, [cit. 2015-04-13]. Dostupné z: `https://edux.fit.cvut.cz/courses/BI-SI1/_media/lectures/03/03.prednaska.pdf`

[8] Nielsen, J.: Diagram případů užití: Základní charakteristika. *Příklady použití diagramů UML 2.0* [online]. 2009, [cit. 2015-04-10]. Dostupné z: `http://uml.czweb.org/pripad_uziti.htm`

[9] Shibboleth. ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE. *Výpočetní a informační centrum ČVUT* [online]. 2014, [cit. 2015-04-12]. Dostupné z: `https://www.civ.cvut.cz/info/info.php?id=207`

[10] SemesterFilter. *KOSapi* [online]. 2012, [cit. 2015-04-12]. Dostupné z: https://kosapi.fit.cvut.cz/projects/kosapi/wiki/SemesterFilter

[11] *Redmine* [online]. 2014, [cit. 2015-04-12]. Dostupné z: http://www.redmine.org/

[12] Nielsen, J.: Usability 101: Introduction to Usability. *Nielsen Norman Group* [online]. 2012, [cit. 2015-04-24]. Dostupné z: http://www.nngroup.com/articles/usability-101-introduction-to-usability/

[13] Enterprise Architect. *Sparx systems* [online]. 2015, [cit. 2015-04-16]. Dostupné z: http://www.sparxsystems.com.au/products/ea/index.html

[14] Zabezpečení před zranitelnostmi. *Nette* [online]. 2015, [cit. 2015-04-13]. Dostupné z: http://doc.nette.org/cs/2.3/vulnerability-protection

[15] Debugování a zpracování chyb. *Nette* [online]. 2015, [cit. 2015-04-13]. Dostupné z: http://doc.nette.org/cs/2.3/debugging

[16] Databáze. *Nette* [online]. 2015, [cit. 2015-04-13]. Dostupné z: http://doc.nette.org/cs/2.3/database

[17] About. *Git* [online]. 2015, [cit. 2015-04-16]. Dostupné z: http://git-scm.com/about

[18] Model-View-Presenter (MVP). *Nette* [online]. 2015, [cit. 2015-04-18]. Dostupné z: http://doc.nette.org/cs/0.9/model-view-presenter

[19] MVC or MVP Pattern – Whats the difference?. *Infragistics* [online]. 2010, [cit. 2015-04-18]. Dostupné z: http://www.infragistics.com/community/blogs/todd_snyder/archive/2007/10/17/mvc-or-mvp-pattern-whats-the-difference.aspx

[20] Basic Design Patterns and Groups. *Difference between ASP.NET 3.5 and ASP.NET 4.0* [online]. 2010, [cit. 2015-04-18]. Dostupné z: http://arunspdreamz.blogspot.cz/2010/03/design-patterns-for-aspnet-developers.html

[21] Nielsen, J.: Why You Only Need to Test with 5 Users. *Nielsen Norman Group* [online]. 2000, [cit. 2015-04-24]. Dostupné z: http://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/

[22] Nielsen, J.: Recruiting Test Participants for Usability Studies. *Nielsen Norman Group* [online]. 2003, [cit. 2015-04-24]. Dostupné z: http://www.nngroup.com/articles/recruiting-test-participants-for-usability-studies/

[23] Nielsen, J.: Quantitative Studies: How Many Users to Test?. *Nielsen Norman Group* [online]. 2006, [cit. 2015-04-24]. Dostupné z: http://www.nngroup.com/articles/quantitative-studies-how-many-users/

[24] Nielsen, J.: Card Sorting: How Many Users to Test. *Nielsen Norman Group* [online]. 2004, [cit. 2015-04-24]. Dostupné z: http://www.nngroup.com/articles/card-sorting-how-many-users-to-test/

[25] What is a Gaze Plot and a Heat Map? *Tobii dynavox* [online]. 2014, [cit. 2015-04-24]. Dostupné z: http://www.tobii.com/en/assistive-technology/global/support-and-downloads/faqs/501a0000000kLH7/

[26] Nielsen, J.: Traveling Usability Lab. *Nielsen Norman Group* [online]. 2012, [cit. 2015-04-25]. Dostupné z: http://www.nngroup.com/articles/traveling-usability-lab/

[27] Nielsen, J.: First Rule of Usability? Don't Listen to Users. *Nielsen Norman Group* [online]. 2001, [cit. 2015-04-25]. Dostupné z: http://www.nngroup.com/articles/first-rule-of-usability-dont-listen-to-users/

[28] MyBalsamiq. *Balsamiq* [online]. [cit. 2015-04-30]. Dostupné z: https://balsamiq.com/products/mockups/mybalsamiq/

[29] AdminLTE Documentation. *Almsaeed studio* [online]. 2015, [cit. 2015-04-30]. Dostupné z: https://almsaeedstudio.com/themes/AdminLTE/documentation/index.html

[30] Bugyík, P.: Documentation. *Grido* [online]. 2015, [cit. 2015-04-30]. Dostupné z: http://o5.github.io/grido-examples/documentation.cs.html

[31] Nielsen, J.: Does User Annoyance Matter?. *Nielsen Norman Group* [online]. 2007, [cit. 2015-05-01]. Dostupné z: http://www.nngroup.com/articles/does-user-annoyance-matter/

[32] Nielsen, J.: Mega-Menus Gone Wrong. *Nielsen Norman Group* [online]. 2010, [cit. 2015-05-01]. Dostupné z: http://www.nngroup.com/articles/does-user-annoyance-matter/

[33] Nielsen, J.: Short-Term Memory and Web Usability. *Nielsen Norman Group* [online]. 2009, [cit. 2015-05-03]. Dostupné z: http://www.nngroup.com/articles/short-term-memory-and-web-usability/

[34] Nielsen, J.: 113 Design Guidelines for Homepage Usability. *Nielsen Norman Group* [online]. 2001, [cit. 2015-05-02]. Dostupné z: http://www.nngroup.com/articles/113-design-guidelines-homepage-usability/

[35] Nielsen, J.: OK-Cancel or Cancel-OK?. *Nielsen Norman Group* [online]. 2008, [cit. 2015-05-03]. Dostupné z: http://www.nngroup.com/articles/ok-cancel-or-cancel-ok/

Příloha **A**

Seznam použitých zkratek

DBS Databázové systémy

SP1 Softwarový týmový projekt 1

CSS Cascade style sheet

CSV Comma-separated values

MVC Model-View-Controller

MVP Model-View-Presenter

RA Relational algebra (relační algebra)

RAT Relational Algebra Translator

SP Semestrální práce

SQL Structured Query Language

SSO Single Sign On

UI User interface

UX User Experience

WF Wireframe webu

XML Extensible markup language

Příloha B

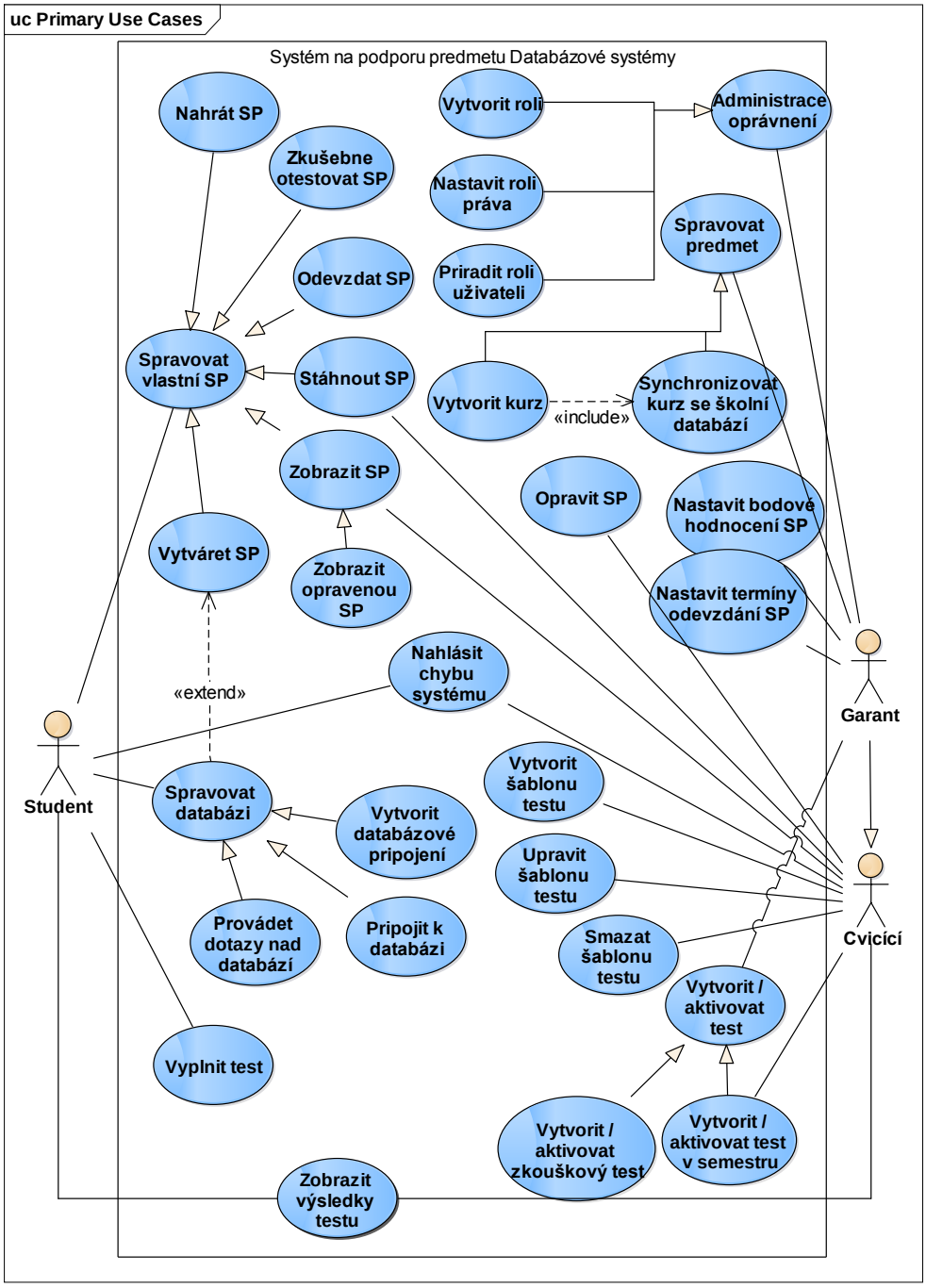
Semestrální práce

Kategorie	Charakteristika dotazu
A	pozitivní dotaz nad spojením alespoň dvou tabulek (Seznam kateder - id, název, jejichž učitelé učil/učili v předmětech, které garantuje katedra KKKK)
B	negativní dotaz nad spojením alespoň dvou tabulek (seznam semestrů - id, název, ve kterých NEzkoušel nikdo z katedry Zubních trhaček)
C	Vyber ty, kteří mají vztah POUZE k (vyber seznam kateder učitelů, kteří učí/učili POUZE v letních semestrech)
D	Vyber ty, kteří/které jsou ve vztahu se všemi - dotaz s univerzální kvantifikací (seznam učitelů - id, jmeno, prijmeni, titul, kteří přednášeli ve VŠECH semestrech počínaje zimním semestrem 2001/2002 až do letního semestru 2007/2008 včetně)
F	spojení - JOIN ON
F2	spojení - NATURAL JOIN | JOIN USING
F3	spojení - CROSS JOIN
F4	polospojení (vnější) - LEFT | RIGHT OUTER JOIN
F5	plné (vnější) spojení - FULL (OUTER) JOIN
G1	vnořený dotaz v klauzuli WHERE
G2	vnořený dotaz v klauzuli FROM
G3	vnořený dotaz v klauzuli SELECT
G4	vztažený vnořený dotaz (EXISTS | NOT EXISTS)
H1	množinové sjednocení - UNION
H2	množinový rozdíl - MINUS (v Oracle)
H3	množinový průnik - INTERSECT
I1	agregační funkce (count | sum | min | max| avg)
I2	agregační funkce nad seskupenými řádky - GROUP BY (HAVING)
J	stejný dotaz ve třech různých formulacích SQL
K	všechny klauzule - SELECT FROM WHERE GROUP BY HAVING ORDER BY
L	pohled VIEW
M	dotaz nad pohledem
N	INSERT .... SELECT příkaz
O	UPDATE s vnořeným SELECT příkazem
P	DELETE s vnořeným SELECT příkazem

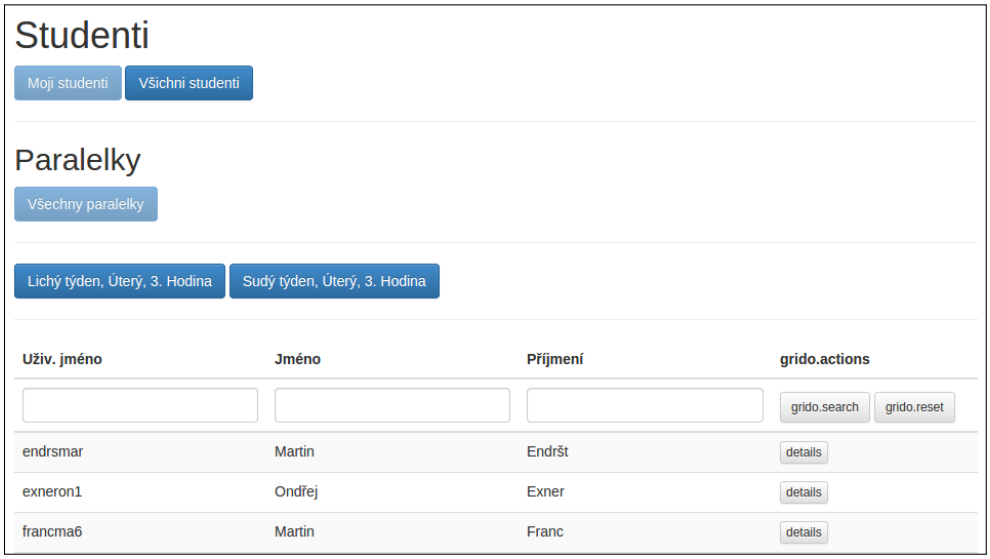
Tabulka B.1: Kategorie dotazů

Příloha C

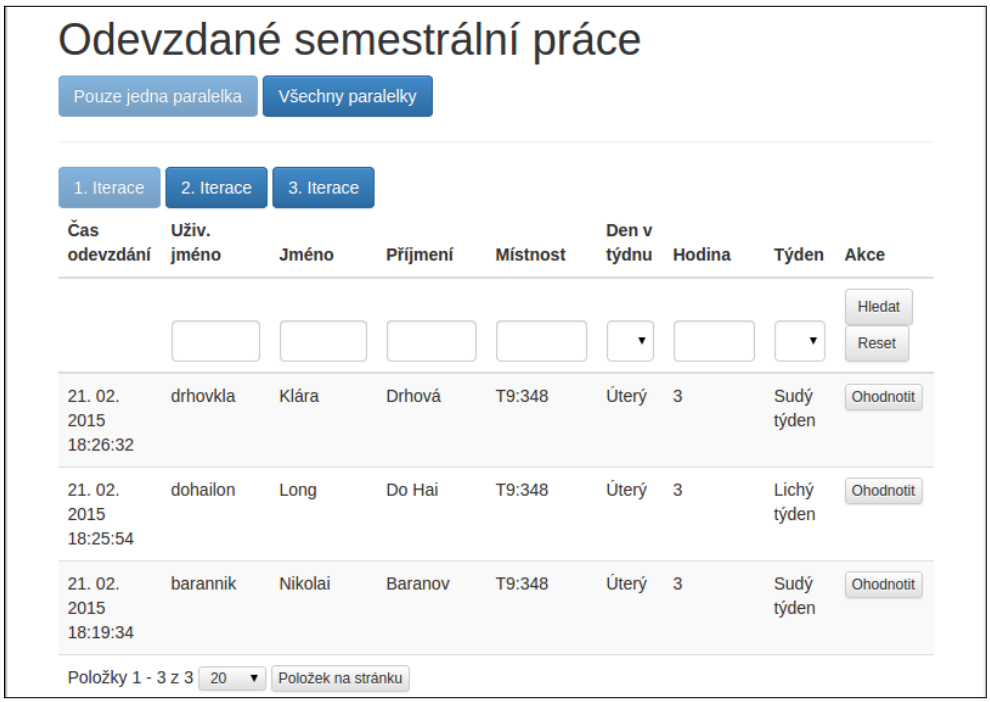
Systém na podporu předmětu DBS – 1. prototyp



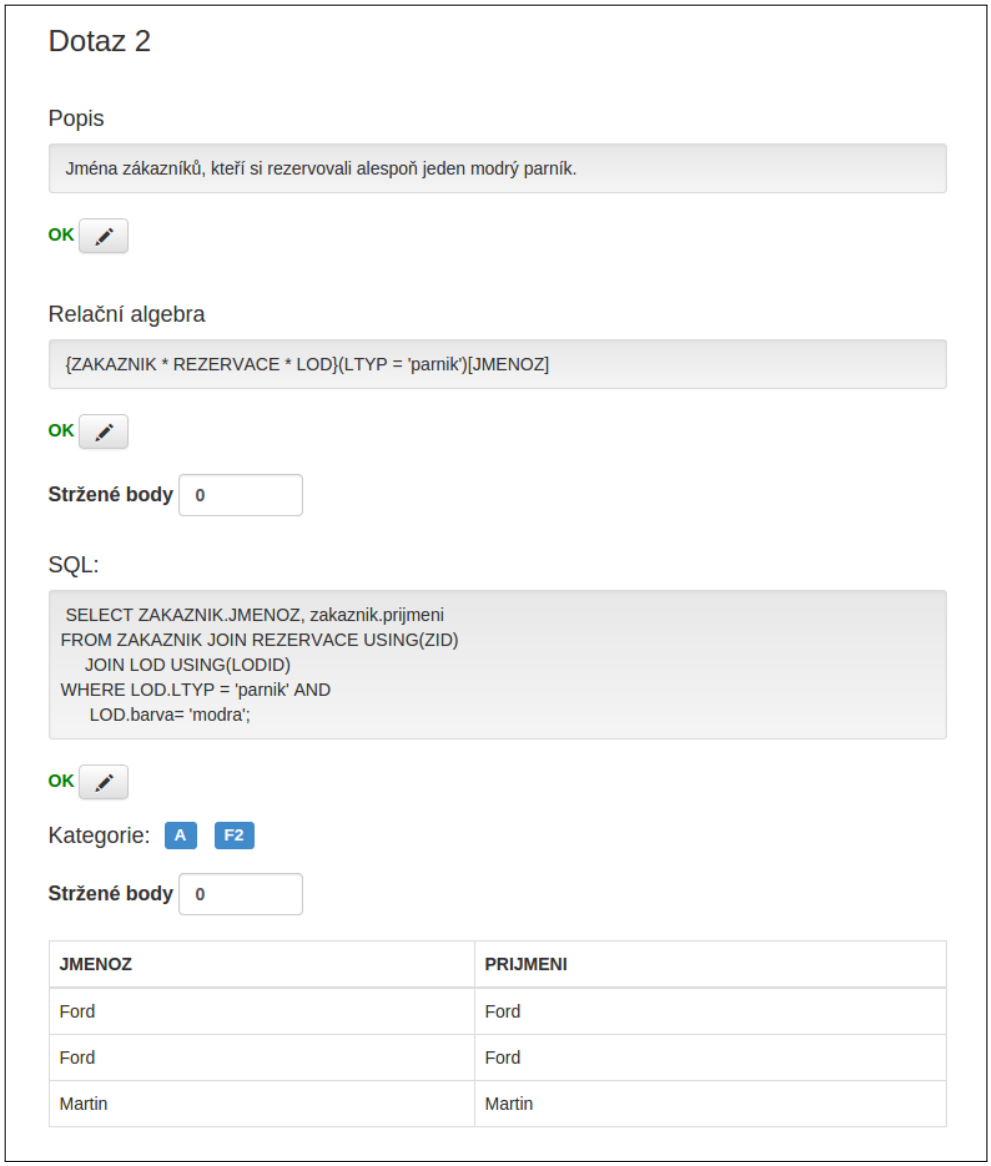
Obrázek C.1: Podrobný diagram případů užití systému na podporu předmětu Databázové systémy



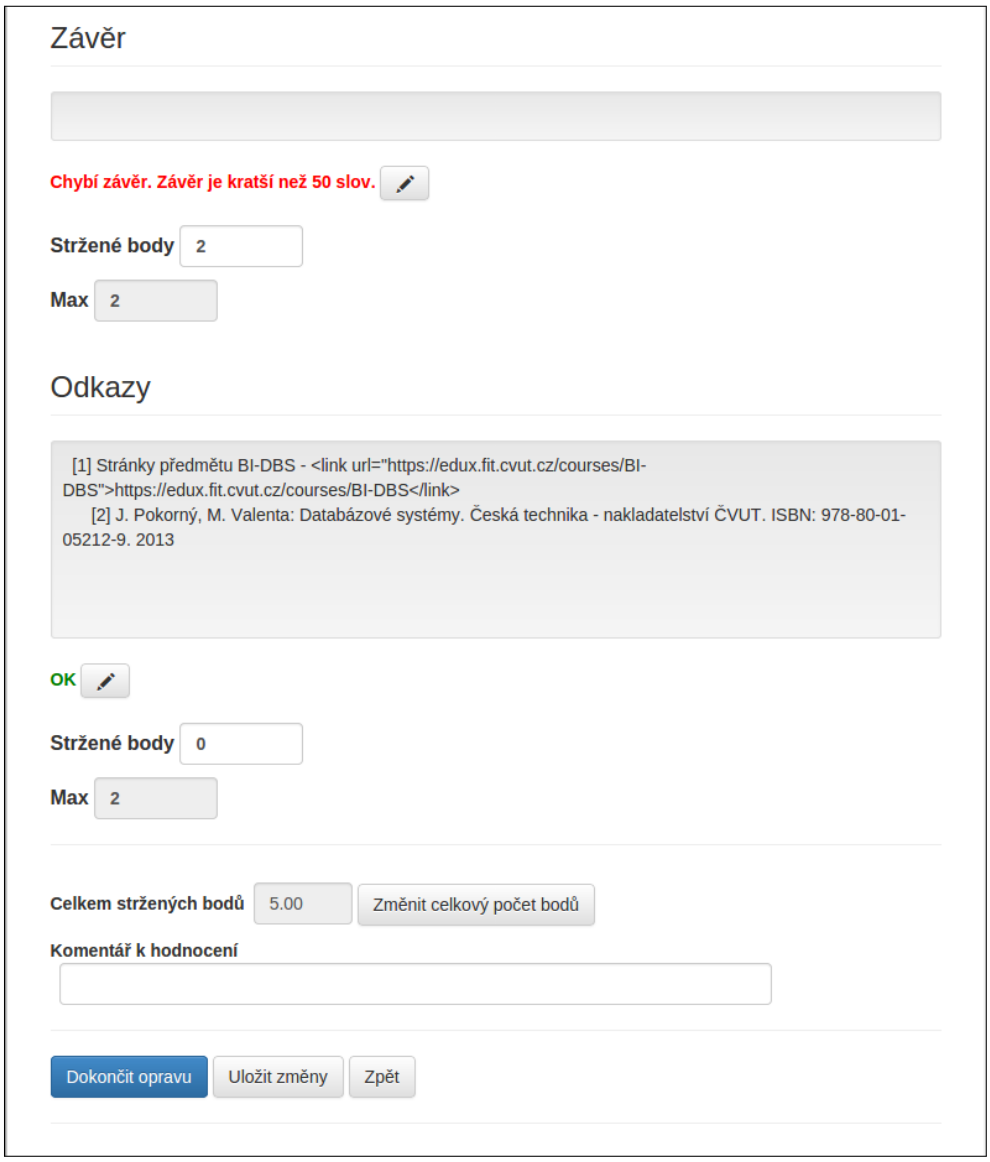
Obrázek C.2: Přehled studentů - filtrování podle paralelek a příslušnosti k přihlášenému vyučujícímu



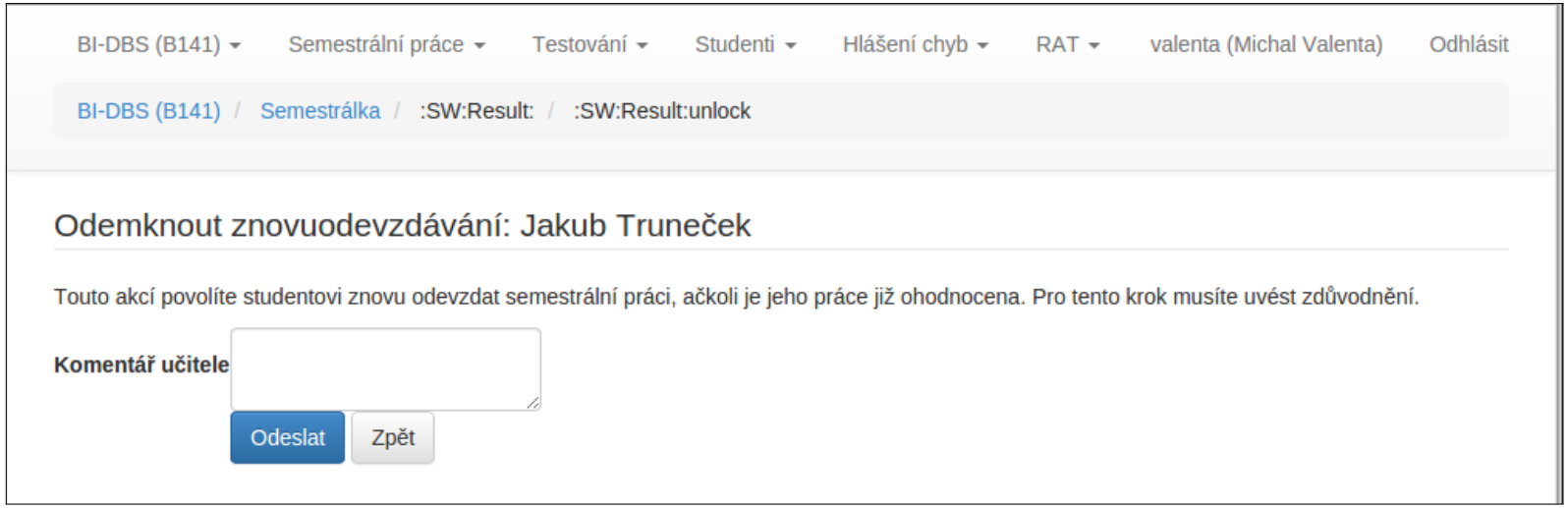
Obrázek C.3: Odevzdané SP připravené k opravě



Obrázek C.4: Opravování 3. iterace – dotazy



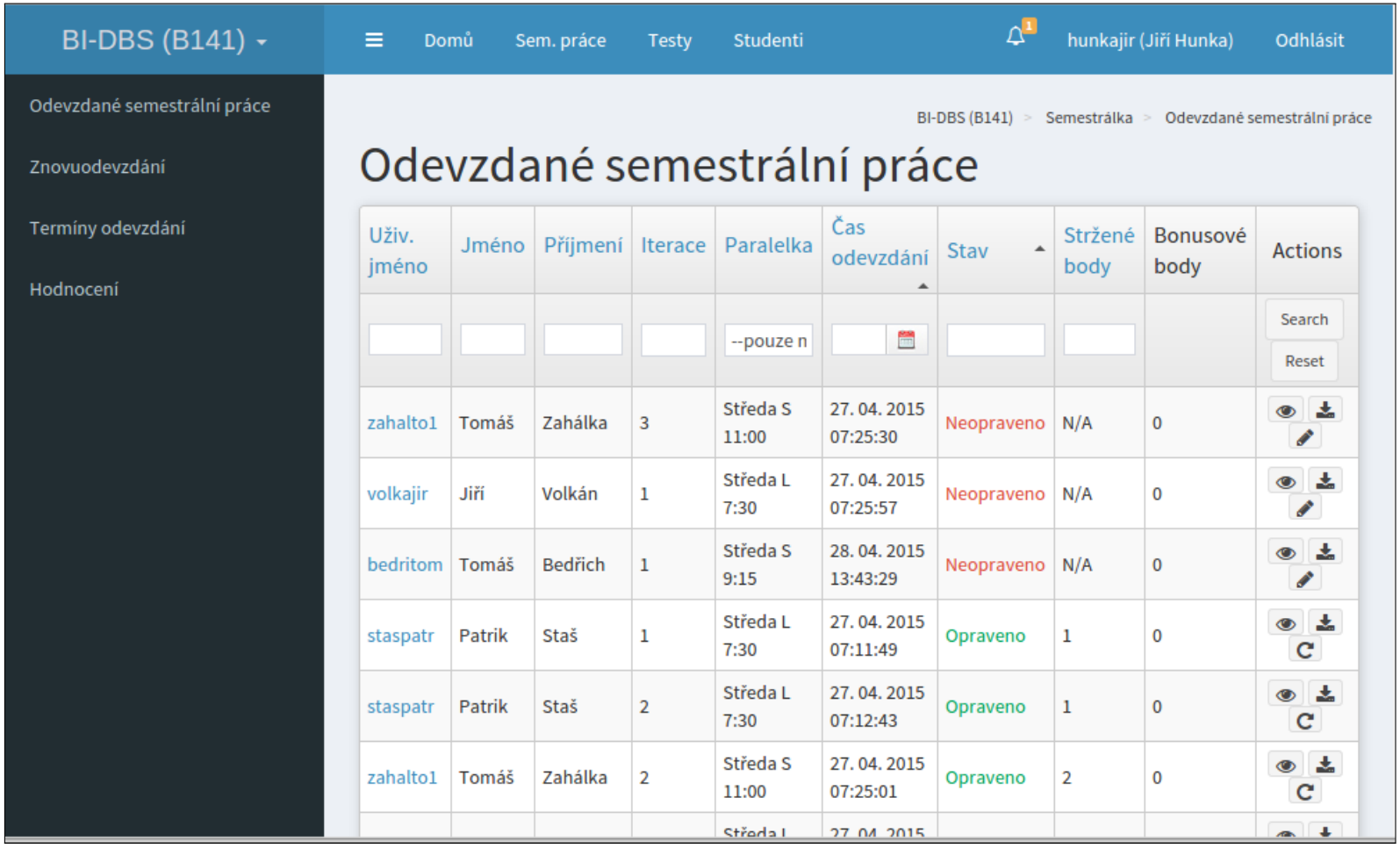
Obrázek C.5: Opravování 3. iterace – ukončení opravy



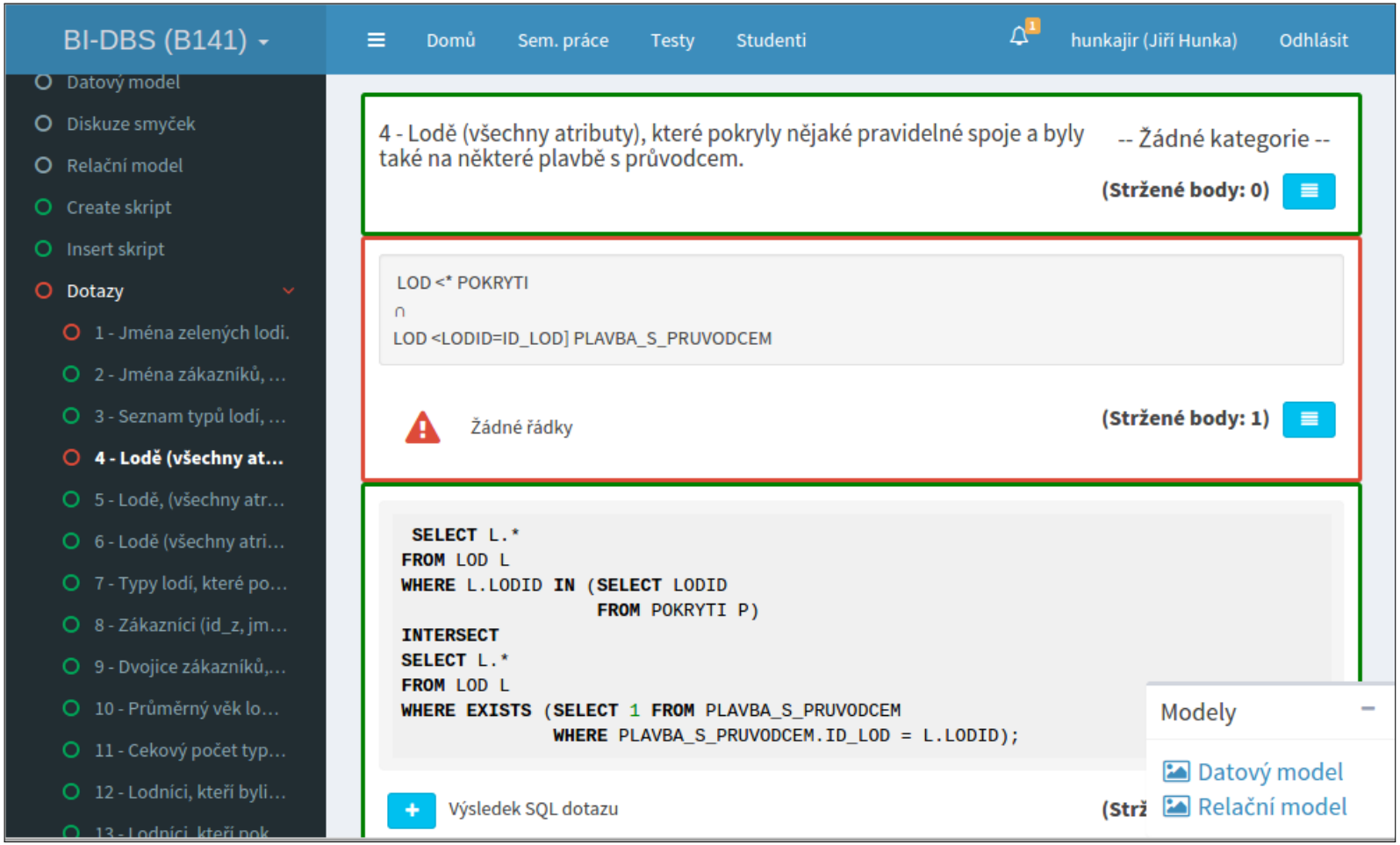
Obrázek C.6: Povolení znovuodevzdání

Příloha D

Systém na podporu předmětu DBS – 2. prototyp



Obrázek D.1: Odevzdané SP



Obrázek D.2: Opravování 3. iterace – dotazy

Příloha **E**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace - 2. prototyp
  - `latex` ..... zdrojová forma práce (obsahuje soubor LaTeX)
- `text` ..... text práce
  - `BP_Sýkora_Jan_2015.pdf` ..... text práce ve formátu PDF
- `ukazka-local` ..... Ukázková semestrální práce
- `tests`
  - `prototype1` ..... Videa z uživatelského testování 1. prototypu
  - `prototype2` ..... Videa z uživatelského testování 2. prototypu