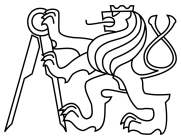
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Systém pro podporu BI-DBS - semestrální práce
Student:	Filip Glazar
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

V předmětu BI-DBS ČVUT FIT, katedry Softwarového inženýrství, mají studenti za úkol vytvořit databázový projekt a s ním i semestrální práci, která jeho tvorbu zachycuje společně s dalšími úkoly. Cílem této práce je navázat na aktuální stav a zkušenosti z týmového projektu realizovaného v rámci BI-SP1 a BI-SP2, kde byl vyvíjen systém pro podporu realizace tvorby semestrálních prací. Současný stav aplikace je funkční, avšak budoucí rozvíjení je díky neuspořádanosti projektu velmi náročné.

Analyzujte současný stav aplikace dostupné na dbs.fit.cvut.cz a její funkcionality týkající se semestrálních prací.
Na základě analýzy navrhněte s velkým důrazem na snadnou údržbu a rozšiřitelnost nové vnitřní uspořádání aplikace.
V rámci návrhu spolupracujte s Petrem Pejšou, který bude přepracovávat část věnující se testování.
Implementujte navržené řešení.
V průběhu vývoje řádně testujte (unit testy apod.).

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 10. ledna 2016

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

Bakalářská práce

Systém pro podporu BI-DBS - semestrální práce

Filip Glazar

Vedoucí práce: Ing. Jiří Hunka

13. května 2016

Poděkování

V první řadě bych chtěl poděkovat panu Ing. Jiřímu Hunkovi za možnost zvolit si toto téma jako svojí bakalářskou práci a za cenné rady při tvorbě této práce. Dále bych chtěl poděkovat Oldřichu Malcovi a Petru Pejšovi za vynikající připomínky k mé práci a spolupráci na vedení týmů v rámci předmětu BI-SP1. Nesmím opomenout ani všechny členy týmů, kterým chci poděkovat za skvěle odvedenou práci. Jmenovitě děkuji Milanu Vlasákovi, Jakubu Novákovi, Jakubu Šterclovi, Ladislavu Zemkovi, Danielu Ondřejovi, Lukáši Krčálovi, Ondřeji Lakomému, Richardu Strnadovi, Juraji Polačkovi, Milanu Vanclovi, Pavlu Kováři a Vojtěchu Bakajovi. V neposlední řádě chci srdečně poděkovat své rodině za neutuchající podporu po celou dobu mého studia.

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 13. května 2016 .....

Odkaz na tuto práci

Glazar, Filip. *Systém pro podporu BI-DBS - semestrální práce.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato bakalářská práce se zabývá vývojem webového portálu pro podporu výuky předmětu Databázové systémy na Fakultě informačních technologií ČVUT. Cílem práce je provést analýzu stávající aplikace a na základě této analýzy vytvořit novou webovou aplikaci, jelikož původní aplikace byla shledána jako nevhodná pro další vývoj. Práce se zabývá zejména tvorbou a kontrolou semestrální práce, která je pro úspěšné absolvování tohoto předmětu povinná. Při návrhu a vývoji systému jsem čerpal hlavně ze zkušenosti s vývojem a provozem stávajícího systému. Důraz byl kladen především na kvalitní návrh a implementaci projektu. Neméně důležitým požadavkem byla modularita systému, snadná rozšiřitelnost a testovatelnost. Samotná webová aplikace je napsána v PHP s použitím velmi populárního českého frameworku Nette. Jak při návrhu, tak při implementaci byly použity moderní technologie a postupy, které jsou v této práci popsány. Podařilo se mi navrhnout a vytvořit aplikaci, která splňuje všechny tyto požadavky a věřím, že bude bude přínosná jak pro studenty, tak pro lektory.

Klíčová slova webový portál, databázové systémy, PHP, analýza, návrh, výuka

Abstract

This thesis deals with the development of a web portal to support the teaching of Database Systems subject at the Faculty of Information Technology, CTU. The aim is to analyze existing application and to create a new web application based on this analysis. Since the original application was found to be unsuitable for further development. Work is mainly focused on the creation and evaluation of semestral project, which is mandatory for the successful completion of this course. For design and development of the new system. I drew mainly from experience in the development and adminsitration of the existing system. Emphasis was placed on high quality design and implementation of the project. An equally important requirement was the system modularity, extensibility and testability. The web application is written in PHP using the very popular czech framework Nette. While designing and implementing the system were used modern technologies and procedures described in this work. I managed to design and build an application that meets all these requirements and I believe it will be beneficial for both students and lecturers.

Keywords website, database systems, PHP, analysis, design, learning

Obsah

Úvod **1**

1 Systém na podporu předmětu BI-DBS **3**
1.1 Předmět Databázové systémy (BI-DBS) 3
1.2 Funkcionalita systému na podporu BI-DBS 3
1.3 Analýza požadavků 5
1.4 Vznik a vývoj 6

2 Analýza původního systému **7**
2.1 Detailní rozbor funkcionality 7
2.2 Požadavky na systém 12
2.3 Nalezené problémy 12

3 Návrh nového systému **17**
3.1 Použité technologie 17
3.2 Struktura projektu 21
3.3 Návrh databáze 21
3.4 Modelový průchod 22

4 Realizace **27**
4.1 Společný základ 27
4.2 Modul semestrální práce 28
4.3 Testování 29
4.4 Dokumentace 30
4.5 Shrnutí 30

Závěr **33**

Literatura **35**

A Seznam použitých zkratek **39**

B Obrázky **42**

C Obsah přiloženého CD **51**

Seznam obrázků

3.1 MVC 19
3.2 Parametry serveru 20
3.3 Struktura zdrojového kódu 21

B.1 Use case model administrace 42
B.2 Use case model semestrální práce 43
B.3 Aktivity diagram semestrální práce 44
B.4 Aktivity diagram semestrální práce editace 45
B.5 Aktivity diagram tvorba testu 46
B.6 Hodnocení studentů 47
B.7 Chybová hláška 48
B.8 Log importu 49
B.9 Dokumentace 50

Seznam tabulek

2.1 Tabulka kategorií dotazů [1] 10
2.2 Požadavky na semestrální práci [2] 11

Úvod

V dnešní době se jen málokteré odvětví lidské činnosti obejde bez automatizace a vzdělávání v žádném případě netvoří výjimku. S počtem studentů studujících daný obor, či předmět stoupají také nároky na organizaci výuky. Především ověřování znalostí studentů a jejich hodnocení klade vysoké časové nároky na lektory. Tento problém se nevyhnul ani předmětu Databázové systémy, který je určen pro studenty bakalářského oboru Bakalářská informatika na Fakultě informačních technologií Českého vysokého technického učení v Praze.

V tomto předmětu musí studenti pro úspěšné absolvování napsat test v semestru, vypracovat semestrální projekt a úspěšně složit zkouškový test. Jejich výstup musí být opraven a ohodnocen. Přitom lze většinu těchto úkonů zautomatizovat. Tak se i stalo a od roku 2014 byl v rámci předmětů SP1 a SP2 na FIT ČVUT pod vedením Ing. Jiřího Hunky vyvíjen Systém pro podporu výuky předmětu Databázové systémy. Tento systém umožňuje tvorbu semestrální práce, psaní testů a zkoušek online. Systém tedy značně automatizuje výuku předmětu.

Já (Filip Glazar, autor této práce) jsem se k vývoji projektu připojil v roce 2015 během mého absolvování předmětu BI-SP2. Věnoval jsem se především vývoji modulu pro správu semestrální práce. Jelikož se tvorbě webových aplikací věnuji již dlouhou dobu, tak pro mě byla tato práce velmi zajímavá. V roce 2016 začal být systém aktivně používán pro výuku a zjistilo se, že systém není ideálně navrhnut, ani implementován. Objevily se problémy s jeho údržbou a dalším rozvojem.

Výše uvedené okolnosti vedly k rozhodnutí, že se systém implementuje úplně od začátku. Funkční požadavky budou zachovány, ale bude kladen velký důraz na modularitu a kvalitu. Tyto skutečnosti mne vedly k rozhodnutí, že si vývoj nového systému a to speciálně část zodpovídající za tvorbu a hodnocení semestrální práce zvolím jako svou bakalářskou práci.

Strukturu této práce jsem zvolil následovně: V kapitole Systém na podporu předmětu BI-DBS popíši stávající webovou aplikaci. Dále, pokračuji analýzou současného stavu existujícího systému a popisem zjištěných nedostatků sys-

tému. V kapitole Návrh nového systému nastíním použité technologie, popíši návrh databáze a představím modelový průchod semestrem. Poslední kapitola je ryze praktická a zabývá se samotnou realizací a implementací nového systému a jsou zmíněny jeho předpokládané výhody oproti systému stávajícímu.

Tato práce pokračuje v bakalářské práci studenta Jana Sýkory [3] a souvisí s bakalářskou prací studenta Petra Pejši, která se podrobněji věnuje modulu pro psaní a vyhodnocování testů.

Kapitola 1

Systém na podporu předmětu BI-DBS

1.1 Předmět Databázové systémy (BI-DBS)

Předmět Databázové systémy je vyučován na Fakultě informačních technologií Českého vysokého učení technického v Praze v rámci bakalářského programu studia.

V rámci tohoto předmětu se studenti seznámí se základními pojmy z oblasti relačních databází. Mimo jiné se základy architektury databázového stroje, způsoby uložení dat, relační algebrou jako teoretickým základem pro jazyk SQL a mnoho dalšího. Naučí se v praxi navrhovat konceptuální model databáze a jeho implementaci[4].

Absolvování tohoto předmětu je povinné pro všechny bakalářské obory a to sebou nese nezanedbatelné časové nároky na lektory v průběhu semestru. Během semestru se píše zápočtový test, studenti vytváří semestrální práci a předmět je zakončen zkouškou. Všechny tyto části byly donedávna vyhodnocovány ručně přesto, že většina těchto úkonů lze automatizovat. Tuto automatizaci zajišťuje webový portál na adrese dbs.fit.cvut.cz.

1.2 Funkcionalita systému na podporu BI-DBS

Pro lepší zasvěcení čtenáře do problematiky uvádíme základní strukturu systému. Dále se v této práci budeme zabývat především modulem pro semestrální projekt a společnými prvky systému, které jsou základními stavebními kameny pro všechny ostatní moduly systému. Pro lepší představu v následujících podkapitolách odkazujeme na ukázkové aktivity diagramy.

1.2.1 Základní funkce

K základním funkcím systému patří správa a přihlašování uživatelů. Každý uživatel má svoji roli a každá role má nějaká oprávnění k daným akcím. Administrátor systému má možnost importovat údaje ze zvoleného semestru jako jsou studenti, rozvrh, paralelky, učitele pro danou paralelku a tak dále. Přihlášení uživatelé si mohou změnit jazyk aplikace. Aktuálně podporované jazyky jsou čeština a angličtina.

1.2.2 Semestrální práce

Tento modul umožňuje tvorbu, editování, odevzdání a opravu semestrální práce (Obrázek B.3). Semestrální práce se skládá z několika základních částí jako je název, popis, konceptuální model, diskuze smyček, relační model, skripty, dotazy v relační algebře a SQL jazyku, závěru a referencí. Odevzdávání semestrálního projektu probíhá ve třech iteracích, kdy student postupně doplňuje jednotlivé komponenty (Obrázek B.4). Po odevzdání iterace proběhne automatická kontrola a práce je předložena učiteli k dokončení opravy. V této práci se budeme zabývat především touto částí systému a to hlavně v kapitolách Analýza původního systému a Realizace.

1.2.2.1 Relační algebra

Jak je uvedeno v [5], *„Velmi silným prostředkem relačního modelu dat pro práci s daty je nástroj, který se nazývá relační algebra. Jde o jazyk vysoké úrovně v tom smyslu, že se v něm nepracuje s n-ticemi relací, nýbrž s celými relacemi. Operátory relační algebry se aplikují na relace a výsledkem vyhodnocení odpovídajících operací jsou opět relace.“*

Relační algebra slouží jako matematický základ jazyku SQL, ačkoli samotná relační algebra není tak silná jako jazyk SQL.

1.2.2.2 SQL

Jazyk SQL je neprocedurální programovací jazyk. To znamená, že nepopisuje jak něco udělat, ale popisuje co požadujeme od databáze. Jeho vznik sahá až do roku 1974, kdy zvládal pouze dotazovací část, tedy výběr dat z databáze dle určitých kritérií. Dnes je již jazyk plně standardizován a umožňuje data definovat, aktualizovat a dokonce kontrolovat přístupová práva[5].

1.2.3 Test v semestru a zkouška

Táto část sytému zodpovídá za tvorbu a opravu testů učitelem (Obrázek B.5) a jejich vypracování studenty. V této práci se tomuto tématu budeme věnovat jen velmi okrajově a to z toho důvodu, že tímto tématem se zabývá bakalářská práce studenta Petra Pejši.

1.2.4 Správa databází

Připojení ke vzdálené databázi je nedílnou součástí celé webové aplikace. Využívá se k otestování funkčnosti dotazů v semestrální práci a také při tvorbě testových otázek a jejich automatické opravě. Aktuálně jsou podporovány databázové stroje Oracle. Každý uživatel může mít uložen neomezený počet databází a připojovat se k nim, dle svého uvážení.

1.3 Analýza požadavků

Pro základní definování systému a popis jeho funkcionalit se často v softwarovém inženýrství využívá analýza požadavků. Těchto analýz existuje celá řada např. seznam požadavků, měřitelné cíle, prototypy, use cases, softwarová specifikace požadavků (SRS) a další. Každá z těchto metod má své výhody i nevýhody a jejich podrobný popis a ukázky jsou mimo rozsah této bakalářské práce. Pro naše účely zvolíme use cases, respektive use case modely.

1.3.1 Use case model

Základní myšlenka use case modelingu je jednoduchá. K tomu, abyste věděli co systém má dělat se musíte nejdříve zaměřit na to, kdo nebo co bude systém používat. Dále je potřeba si říct co systém provede a jak, aby to bylo uživatelům k užitku. Základními komponentami use case modelingu jsou: Actors: Reprezentují osoby nebo věci, které nějakým způsoben interagují se systémem. Zaměřujeme se na actors, abychom se ujistili, že systém dělá něco užitečného. Use cases: Reprezentují nějakou činnost nebo hodnotu, kterou systém vykoná pro actors. Use cases nejsou funkce ani vlastnosti a nemohou být dekomponovány. Množina všech actors a use cases popisují systém a je nazývána jako systémový use case model.[6]

1.3.1.1 Administrace

Tímto use case modelem (Obrázek B.1) popisujeme jednu ze základních součástí systému a to administraci. V této části systému můžeme spravovat a importovat uživatele, měnit jejich role a oprávnění. Data při importu uživatelů jsou načítána z KOSApi[7]. Shledali jsme, že tato funkcionalita byla navrhnuta a implementována, bez výrazných chyb a problému. Chceme však podotknout, že jak import uživatelů, tak správa jejich rolí je příliš složitá a uživatelsky nepřívětivá. Možnosti a schopnosti stávajícího systému samozřejmě přesahují rozsah námi uvedených modelů. Cílem této práce není popsat všechny funkce systému, ale zaměřit se na ty hlavní, kterými jsou právě administrace a správa semestrální práce.

1.3.1.2 Semestrální práce

V use case modelu (Obrázek B.2) modulu semestrální práce jsou v roli studenta základními funkčními prvky tvorba, editace a odevzdání. Role učitel může opravovat semestrální práce studentů a nastavovat termíny odevzdání. Po odevzdání práce studentem je provedeno automatické hodnocení ze kterého může učitel vycházet.

1.4 Vznik a vývoj

Vývoj tohoto webové portálu začal v roce 2014 v rámci předmětů BI-SP1 a BI-SP2 vyučovaných na Fakultě informačních technologií Českého vysokého technického učení v Praze pod vedením Ing. Jiřího Hunky.

Tyto dva předměty mají za cíl studenty připravit na vývoj softwaru v praxi. Během letního semestru se činnost studentů soustředí především na návrh projektu a organizaci práce společně s jejím správným a řádným vykazováním. V semestru zimním, kdy probíhá výuka BI-SP2 se většinou studenti zabývají implementací jimi navržených řešení.

Během předchozích dvou let byl vyvinut stávající systém a dokonce se mu věnovalo i několik bakalářských prací například bakalářská práce studenta Jana Sýkory[3]. V letním semestru 2016 byl portál uveden do provozu a začal se plně využívat při výuce BI-DBS. Naneštěstí se objevilo mnoho problémů jak v návrhu, tak samotné implementaci a projekt byl shledán nevhodným pro další rozvoj.

1.4.1 Týmy v BI-SP1

Opět pod vedením Ing. Jiřího Hunky jsme sestavili tři týmy po čtyřech členech, které se aktivně věnovali vývoji nového systému. Náplní práce v semestru těchto týmů bylo v kooperaci s autorem této práce a s Petrem Pejšou, který se také zabývá vývojem nového systému, speciálně části testů, vytvořit systém nový. V první fázi semestru jsme se s týmy věnovali analýze funkčních požadavků starého systému, které se víceméně kryjí se systémem novým. To nám pomohlo odhalit mnohé chyby a slepé uličky v původním návrhu. V druhé polovině semestru probíhal návrh nového systému na základě předešlé analýzy. Nakonec, se začaly všechny týmy věnovat implementaci okrajových částí systému, který byl vytvořen autorem této práce. Předpokládáme, že členové těchto týmů budou pokračovat v rozvoji systému v rámci předmětu BI-SP2. Bylo tedy velmi důležité, aby se poučily ze špatné implementace původního projektu a tyto získané zkušenosti využily při rozvoji nového systému.

Kapitola 2

Analýza původního systému

2.1 Detailní rozbor funkcionality

Před samotnou analýzou jsme detailně rozebrali stávající funkcionalitu systému, abychom mohli zaručit absolutní nezávislost těchto prvků při implementaci portálu nového.

2.1.1 Základní prvky

Základními prvky myslíme všechny společné části aplikace, které jsou nebo mohou být využity dalšími moduly. Přesto, že tato práce není přímo zaměřena na obecné části systému jsou nedílnou součástí celé aplikace a při jejich analýze i návrhu pro nový systém jim byla věnována adekvátní pozornost.

2.1.1.1 Správa uživatelů

Vyjma administrátora aplikace jsou všichni uživatelé do systému importováni skrze KOSApi[7]. Z tohoto API systém dále získá kódy semestrů, kurzů, paralelek a zkoušek. Tyto data, pak slouží jako datový základ celé aplikace.

2.1.1.2 Role uživatelů

V systému vystupují uživatelé v různých rolích. Administrátor, garant, zkoušející, přednášející, cvičící a student. Každá role má svoje oprávnění k různým akcím a funkcím v systému. Jeden uživatel může vystupovat i ve více rolích najednou.

2.1.1.3 Správa databází

V tomto modulu si může každý uživatel vytvořit připojení ke vzdálené databázi. Připojení je pak využíváno při provádění dotazů a testování. Momentálně

podporovaný databázový stroj je pouze Oracle, ale v budoucnu se počítá i s alternativami jako je MySQL nebo PostgreSQL.

2.1.2 Semestrální práce

Semestrální práce se skládá z několika základních komponent, které vytváří student. Většina těchto částí je povinná a jsou hodnoceny v rámci odevzdání. Na každou komponentu jsou v rámci semestru kladeny různé požadavky, které jsou poté hodnoceny při opravě práce.

2.1.2.1 Titulek a popis

Titulkem je myšlen název dané semestrální práce a popis by měl obsahovat základní informace o tom, jaký problém práce řeší a jak k řešení tohoto problému student přistoupil.

2.1.2.2 Konceptuální model

Konceptuální model reprezentuje samotné informace v databázi. Objekty tohoto schématu jsou struktury, operace a integritní omezení datového modelu. V relačních databázích obsahuje konceptuální model tabulky a samotná integritní omezení. V objektově orientovaných databázích obsahuje samotné třídy, které tvoří perzistentní data zahrnující datové struktury a metody dané třídy[8].

Vzhledem k tomu, že hlavní náplní předmětu Databázové systémy jsou především relační databáze, tak je po studentech vyžadován konceptuální model používaný při návrhu relační databáze.

2.1.2.3 Diskuze smyček

Diskuze smyček je silně spjata s konceptuálním modelem. Při návrhu databází je jedním z varovných faktorů, právě vzniklá smyčka, která nám může naznačit, že náš návrh je špatný. Existence smyčky v návrhu však automaticky neimplikuje špatný návrh. Právě proto je vyžadováno, aby student slovně popsal existující smyčky v návrhu a jejich správnost řádně zdůvodnil.

2.1.2.4 Relační model

Relačním modelem je myšlen konkrétní fyzický model databáze. Tento model umí vytvořit nástroj Oracle SQL Data Modeler[9] z modelu konceptuálního. Proto není v semestrální práci povinný.

2.1.2.5 Dotazy

Dotazy jsou jedním ze základních prvků celé semestrální práce. Dotaz se skládá z formulace v přirozeném jazyce, zápisem v relační algebře a vyjádřením v SQL

jazyce. Zápis v relační algebře je možné přeložit pomocí překladače vytvořeném v rámci bakalářské práce studenta Martina Kubiše do jazyka SQL. Poté co je relační algebra přeložena, je dotaz proveden nad aktuálně zvolenou databází. Vykonat lze i zápis v SQL a je zobrazen formátovaný výsledek ve formě tabulky.

2.1.2.6 Kategorie dotazů

Dotazy jsou automaticky rozřazeny do kategorií, které pokrývají. Toto rozřazení provádí systém na základě struktury a výsledku SQL dotazu. Uživatel může některé kategorie přiřadit i manuálně, ale nemůže zrušit kategorie, které systém přidělil automaticky.

Kategorie	Popis
A	pozitivní dotaz nad spojením alespoň dvou tabulek (Vyber seznam studentů, kteří absolvovali (mimo jiné) (alespoň jeden) povinný předmět bakalářského programu Informatika)
B	negativní dotaz nad spojením alespoň dvou tabulek („Vyber seznam studentů, kteří neabsolvovali ani jeden povinný předmět bakalářského programu Informatika“).
C	Vyber ty, kteří mají vztah POUZE k ...(„Vyber seznam studentů, kteří absolvovali pouze povinné předměty bakalářského programu Informatika“)
D1	Vyber ty, kteří mají vztah POUZE k ...(„Vyber seznam studentů, kteří absolvovali pouze povinné předměty bakalářského programu Informatika“)
D2	kontrola výsledku dotazu z kategorie D1
F1	spojení - JOIN ON
F2	spojení - NATURAL JOIN,l JOIN USING
F3	spojení - CROSS JOIN
F4	vnější polospojení - 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 - 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 (příkaz, který vloží najednou množinu řádků, které jsou výsledkem vnořeného poddotazu)
O	UPDATE s vnořeným SELECT příkazem
P	DELETE s vnořeným SELECT příkazem

Tabulka 2.1: Tabulka kategorií dotazů [1]

2.1.2.7 Reference

V této části je studentovým úkolem pouze vypsat reference, které využil při tvorbě semestrální práce. Tato část je nepovinná, pokud nebyly žádné reference použity.

2.1.2.8 Závěr

V závěru semestrálního projektu je po studentech požadováno, aby celkově shrnuli význam jejich semestrální práce a vyjádřili svůj názor na to, co jim absolvování předmětu přineslo.

2.1.2.9 Odevzdání

Odevzdání a kontrola probíhají ve třech iteracích. V každé iteraci jsou kladeny jiné požadavky a podle toho je voleno i bodové ohodnocení. Každá následující iterace vyžaduje splnění požadavků z iterace předchozí. V poslední iteraci je vyžadována kompletní semestrální práce se všemi splněnými náležitostmi.

Iterace	Požadavky
1.	Volba tématu a název práce Textový popis Alespoň tři dotazy v přirozeném jazyku
2.	Konceptuální schéma Diskuze smyček Zdrojový soubor modelu Alespoň 10 dotazů v přirozeném jazyce Dotazy musí pokrývat kategorii C a D
3.	Create a insert skript Alespoň 10 dotazů v relační algebře Minimálně 25 dotazů v SQL jazyce Dotazy musí pokrývat všechny kategorie Závěr případně reference

Tabulka 2.2: Požadavky na semestrální práci [2]

Během odevzdání se automaticky vyhodnotí splněné a nesplněné požadavky a automat vypíše nalezené chyby, či potvrdí správnost řešení. Student má také možnost napsat ke každé položce svůj komentář, který je při opravování zobrazen učiteli.

2.1.2.10 Oprava učitelem

Odevzdanou a automatem vyhodnocenou práci si může učitel zobrazit a upravit její hodnocení, dle vlastního uvážení. Až po provedení opravy učitelem je práce označená jako opravená a výsledek se zobrazí studentovi.

2.2 Požadavky na systém

Požadavky na software jsou jasně definovaná pravidla toho, co systém umožňuje a jaké má mít vlastnosti. V následujících podkapitolách se budeme věnovat požadavkům, které byly na systém kladeny a obzvláště těm, které nebyly splněny. Pro kategorizaci požadavků použijeme metodiku FURPS.

(F)unctional features, capabilities, security.

(U)sability human factor, help, documentation.

(R)reliability frequency of failure, recoverability, predictability.

(P)erformance response times, throughput, accuracy, availability, resource usage.

(S)upportability adaptability, maintability, internationalization, configurability.

[10]

V následující kapitole se budeme těmito požadavky zabývat a speciálně těmi, které považujeme za nedostatečné nebo dokonce úplně nesplněné.

2.3 Nalezené problémy

Na základě informací od studentů a lektorů jsme identifikovali mnoho problémů v sytému. Dalším podstatným zdrojem byla autorova účast na samotném vývoji stávajícího systému. Jedním z největších problému byla udržitelnost systému. Vzhledem k nedostatečné dokumentaci, nekonzistentnímu kódu a nulovému pokrytí testy byla údržba systému velmi náročná a neefektivní.

2.3.1 Dokumentace

Jak je uvedeno v [11] *„Říka se, že zapsání myšlenky nám pomáhá myslet. Je opravdu snadné mít svoji představu v hlavě a tam zní jako dokonalá, ale až samotné zapsání myšlenky vyžaduje zamyšlení. Psaní dokumentace zlepšuje návrh vašeho kódu a umožní vám přemýšlet o návrhu mnohem více formálněji. Dalším a neméně důležitým přínosem dokumentace je daleko snazší zapojení dalších lidí do vývoje.“*

Projekt jako je systém pro podporu výuky předmětu Databázové systémy vyžaduje kvalitní a obsáhlou dokumentaci. Hlavně z toho důvodu, že během vývoje se na projektu vystřídalo mnoho programátorů a předpokládáme, že každý akademický rok bude na projektu pracovat tým nový. Každý programátor má svůj osobitý styl a rozdílný pohled na řešení problému. Ovšem, pokud má nový člen týmu k dispozici dokumentaci může se po jejím přečtení rychle zorientovat a adaptovat se na styl, který je v projektu aktuálně používán.

Předchozí vývojové týmy tvořily v průběhu semestrů vývojovou dokumentaci v aplikaci pro správu projektu Redmine[12]. To mohlo sloužit jako dobrý základ pro projektovou dokumentaci celého projektu. Analýzou obsahu jsme zjistili, že pokrytí projektu a všech jeho vlastností a specifik je nedostatečné.

2.3.2 Nekonzistence a znovu použitelnost kódu

Během psaní této práce je systém aktivně používán při výuce. To znamená, že vyžaduje údržbu a opravy nalezených chyb. Na těchto činnostech se autor této práce společně se studenty předmětů BI-SP1 aktivně podílel. Zjistili jsme, že další práce na projektu je velmi obtížná a to zejména díky rozdílnému návrhu a použitým programovacím technikám v různých modulech systému. Ve zdrojovém kódu se velmi obtížně orientuje a hledaní souvislostí je běh na dlouhou trať i pro zkušeného vývojáře. Těmto komplikacím se lze vyhnout, minimálně omezit jejich počet, aplikací znovupoužitelného kódu.

Podle Naurera a Randella [13], *„software reuse is the procces of creating software systems from existing software rather than building software from scratch.“*

Podle Meyera [14], *„reusability is the ability of software products to be reused, in whole or part, for new applications.“*

Tyto myšlenky považujeme za naprosto základní. Nemusíme je ovšem chápat jako nějaké dogma a můžeme se zaměřit i na daleko menší znovupoužitelné kousky kódu. Objektově orientovaný návrh nám k tomu nabízí mnoho prostředků.

Objektově orientovaný vývoj softwaru (OOP) nám umožňuje dívat se na software tak, aby jsme mohli přemýšlet o vývoji jako o interakci objektů z reálného světa. Základní koncept objektově orientovaných programovacích jazyků leží ve zkombinování dat a funkcí pro práci s nimi do jedné jediné jednotky. Pojem objektově orientovaný znamená, že software je kolekce diskrétních objektů, které zahrnují jak datové struktury, tak jejich chování. Při použití toho přístupu již nerozdělujeme problém na funkce, ale doménu problému si můžeme představit jak interakci mezi objekty. Objektově orientovaný program obsahuje množství objektů, které mezi sebou komunikují voláním členských funkcí objektů ostatních. Hlavní výhody OOP jsou:

Data abstraction It refers to the facility to create user-defined data types for modeling real world object, having similiar properties and a set of permitted operators.

Encapsulation The mechanism of associating the code and the data it manipulates into a single unit and keeping them safe from external interference, is called encapsulation.

Inheritance It involves a creation of new classes (derived classes) from the existing ones (base classes). The new derived class inherits the characteristics of the base class and also has some of its one.

Polymorphism It allows a single name or operator to be associatedwith different operations depending on type of data passed to it.

Message passing It is proccess of invoking an operation on an object.

Extensibility This feature allows extension of the functionality of the existing software components.

Persistence The object (data) outlives the program execution time and survives several executions of a program.

Delegation The child classes send requests to parent classes. In delegation, two objects are involved in handling request.

Genericity It is a technique for defining software components that have more than one interpretation depending on the data type of parameters. Thus, it allows the declaration of data items without specifying their exact type.

[15]

Samotné paradigma objektově orientovaného programování nezaručuje, že při jeho použití bude kód automaticky znovupoužitelný. Vždy to leží na bedrech vývojáře. Shledali jsme, že koncept znovupoužití kódu je sice využíván, ale velmi zřídka. Pokud už je použit, tak většinou špatně. Jedním z mnoha případů je tabulka hodnocení studentů (Obrázek B.6). Tento výpis je požadován ve třech různých částech systému v naprosto totožné podobě. Považujeme za logické, aby byla tabulka implementována pouze jednou. V systému je na každém z těchto míst tabulka implementována znovu.

2.3.3 Uživatelská přívětivost

Uživatelská přívětivost a jednoduchost používání software je jedním ze základních kriterií pro hodnocení úspěchu, či neúspěchu vývoje. Termín uživatelsky přívětivý je naneštěstí subjektivní. Neexistuje dobrá a obecně uznávaná definice tohoto pojmu. Program, který nazveme uživatelsky přívětivý pro nějakého uživatele, může být záhadou pro jiného uživatele. To znamená, že při definování programu ve funkční specifikaci jako uživatelsky přívetivý není určující pro rozsah a vlastnosti projektu. Přesto se tento pojem hojně využívá. Existuje mnoho věcí, které může vývojař, či designér softwaru udělat proto, aby byl uživatel s používáním programu spokojen.[16]

2.3.3.1 Omezení vstupů uživatele

Čím méně toho uživatel musí zadat, tím menší je prostor pro chyby. Další z nepříjemných vlastností může být nutnost opakovaně zadávat stejné údaje. Tento problém jsme nalezli i v systému a to konkrétně u správy databázových připojení. Pro připojení je vyžadováno heslo a toto heslo není v systému žádným způsobem uloženo, to znamená, že při každém přihlášení je uživatel nucen toto heslo vyplnit.

2.3.3.2 Nápomocné chybové hlášky

Obsáhlé a hlavně výstižné chybové hlášky jsou jedním ze základním požadavků na uživatelsky přívětivý software. Jedním z mnoha případů špatné chybové hlášky je sdělení uživateli, že zadal chybné heslo při připojení k databázi (Obrázek B.7). První problém je nesrozumitelnost chybové hlášky a druhým problémem je význam sdělení pro uživatele. Uživatele zajímá pouze to, že zadal špatné heslo. Jedním z dalších problémů je zmizení hlášky po uplynutí několika málo vteřin a uživatel nemusí v tomto intervalu stihnout hlášku přečíst a pochopit.

2.3.4 Nasazení v produkčním režimu a údržba

Cesta softwaru z vývojového prostředí do reálného produkčního řešení bývá trnitá. Za daleko složitější proces je však považována aktualizace již nasazeného řešení. Hrozí riziko výpadků, smazání dat a v nejhorším případě kompletní nefunkčnosti systému. Právě proto existují různé techniky jak tato rizika minimalizovat. Těmto technikám se budeme věnovat v kapitole Realizace.

Aktuálně jsou drženy dvě vývojové větve. Jedna je pro vývojáře a druhá je nasazena na serveru. Tyto větve mají drobné rozdíly, které činí aktualizaci produkce velmi složitou a náchylnou na chyby.

Kapitola 3

Návrh nového systému

Při návrhu nového systému jsme využili naše zkušenosti se systémem předchozím. V potaz byly vzaty především nedostatky, na které jsme během analýzy popsané v kapitole Analýza původního systému narazili. Vzhledem k tomu, že funkční požadavky se kryly s požadavky aktuálního systému zaměřili jsme se především na kvalitu zdrojového kódu, uživatelskou přívětivost, výkon aplikace, dokumentaci a rozšiřitelnost systému. Všechny tyto vlastnosti aktuálnímu systému chybí.

3.1 Použité technologie

Technologie jsme volili ve většině případů shodné se stávajícím systémem. Jedná se o běžné technologie používané při tvorbě webových aplikací, což nám dává výhodu pro čerpání informací z mnoha dostupných zdrojů.

3.1.1 Apache

Apache je open-source implementace HTTP web serveru. Jedná se o jeden z nejrozšířenějších web serverů vůbec. Projekt je vyvíjen již od roku 1995 a jeho vývoj stále pokračuje. Je populární především kvůli tomu, že je velmi robustní a hlavně je dovoleno jeho komerční použití[17].

Apache je používán na více jak 50 procentech všech webových stránek[18]. My jsme tento server zvolili hlavně pro jeho dlouholetou tradici, jednoduchou správu a kompletní dokumentaci pro vývojáře.

3.1.2 PHP

„Následník Perlu navržený pro webové skriptování na straně serveru. Skripty jsou typicky vloženy ve webových stránkách. Byl vyvinut Rasmusem Lerdorfem v roce 1995 za účelem správy jeho osobní domovské internetové stránky. Nyní je jako oficiální název používán rekurzivní akronym (PHP: Hypertext

Preprocessor). Novější verze vyvíjí Andi Gutmans a Zeev Suraski ve spolupráci s Lerdorfem. PHP obsahuje vestavěnou podporu pro široké spektrum internetových protokolů a pro přístup do více jak tuctu komerčních databázových řešení. Verze 5 z roku 2004 přidala podporu pro objektově orientované programování jako je složená dědičnost, iterátory, automatické načítání, strukturované zpracování výjimek, reflexi, přetěžování metod a volitelnou deklaraci typu parametru. "[19]

PHP patří k nejpopulárnějším programovacím jazykům pro vývoj webových aplikací. Jako takový je poslední dobou vytlačován jinými platformami, ale v kombinaci s nějakým dostupným frameworkem, který přidává nové vlastnosti jazyka, je velmi silným nástrojem pro efektivní implementaci komplexních systémů.

3.1.3 Nette Framework

Nette Framework je populární český framework pro PHP. Mezi jeho hlavní přednosti patří zejména čistý objektový návrh, výkon, zabezpečení a rozsáhlá komunita vývojářů. Dalšími a neméně důležitými vlastnostmi tohoto frameworku jsou šablonovací systém a formuláře. Framework je vhodný pro práci v týmu. Podporuje nejmodernější technologie a přirozeným způsobem vede programátora k používání nejnovějších a korektních postupů, které nabízí jazyk PHP. Hlavním důvodem proč jsme zvolili toto řešení je excelentní podpora MVC návrhu. O Nette a jeho kladech i záporech se toho dá napsat velmi mnoho, ale to není cílem této bakalářské práce[20][21].

3.1.4 MVC

„MVC je softwarový návrhový vzor postavený na vzájemném propojení tří hlavních komponent. V programovacích jazycích jako je PHP často s velmi úzkou vazbou na a objektově orientované programovací paradigma. Těmito třemi komponenty jsou model, view and controller. "[22]

3.1.4.1 Model

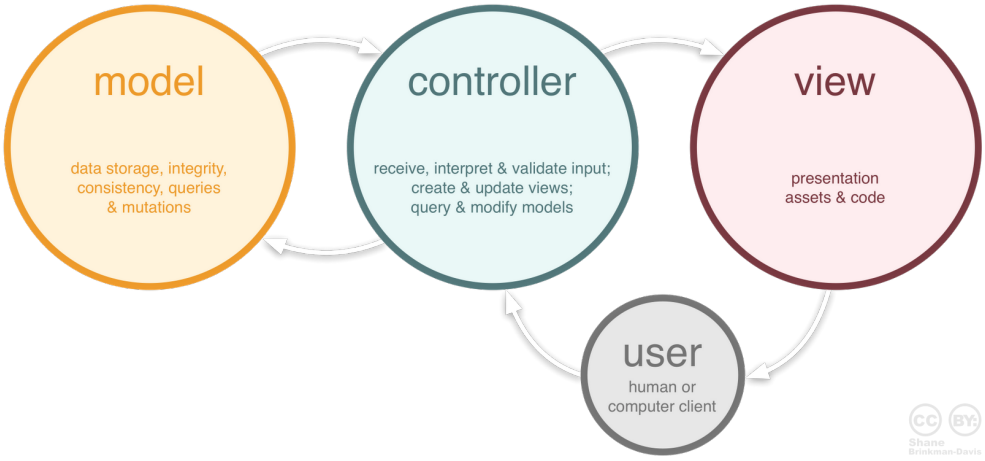
V modelu nalezneme celou logiku aplikace. Je to místo, kde jsou držena data a hlavně funkce systému. Ať už se jedná o přístup a načtení dat z a do databáze, komunikace přes API se systémem třetí strany nebo načtení a zpracování dat ze souboru. Všechny tyto funkcionality by měl zajišťovat právě model[22].

3.1.4.2 View

View neboli pohled je vrstva aplikace zodpovědná za uživatelské rozhrání a prezentaci dat koncovému uživateli. U webových aplikací je tvořen většinou kombinací HTML, CSS a JS[22].

3.1.4.3 Controller

Kontroler zajišťuje propojení modelu a pohledu. Kontroler reaguje na akci uživatele v pohledu a jeho logika rozhodne o tom, jaký pohled a s jakými daty získanými z daného modelu uživateli vrátí[22].



Obrázek 3.1: Model - View - Controller [1]

Tento návrhový vzor je velmi vhodný pro větší projekty. Umožňuje efektivně oddělit různě části aplikace a do vývoje zapojit například jen programátora se specializací na frontend, který se nemusí vůbec zajímat o to jak data, která mu jsou předána do pohledu vzniknou a jak se zpracují. Může se tedy soustředit pouze na správnou funkčnost pohledu a jeho vykreslení například na různých zařízeních.

3.1.5 PostgreSQL

Vybrat správný databázový stroj, který bude aplikace využívat pro nás nebyl jednoduchý úkol. Na databázový stroj jsme kladli pouze jediný požadavek a to, aby byla databáze relační. Hlavně z toho důvodu, že data v systému jsou vhodná pro uložení v relačním typu databáze. Tento požadavek splňuje mnoho databázových strojů od MySQL, přes MariaDB až po Oracle. Nakonec jsme zvolili PostgreSQL.

PostgreSQL je open-source objektově-relační databázový stroj. Je vyvíjen více jak 15 let a má výborné reference. Jeho velkou výhodou je komplexní podpora standardu ANSI-SQL:2008 a vestavěná podpora pro PHP skrze existující rozšíření. Běží na všech hlavních operačních systémech jako je Linux, Windows a OS X[23].

3.1.6 Gitlab

Pokud na projektu pracuje více vývojářů najednou je vhodné nějakým způsobem zajistit slučování a zálohu zdrojových kódu od různých subjektů. K tomuto účelu a nejenom k němu, slouží verzovací systémy.

Verzovací systém je schopný zaznamenávat změny, které byly provedeny v souboru nebo skupině souborů za danou časovou periodu. Umožňuje se vracet ke starším verzím těchto souborů. Formálněji můžeme říci, že verzovací systém je softwarový balíček, který po inicializaci monitoruje změny v souborech a umožňuje označit změny na různých úrovních, ke kterým je možno se vracet nebo je slučovat s jinými[24].

Těchto verzovacích systémů existuje několik typů. Jedním z nejpopulárnějším je distribuovaný verzovací systém Git[25]. Tento systém byl zvolen mimo jiné proto, že FIT ČVUT nabízí studentům vlastní verzovací řešení na bázi Gitlabu[26], to znamená, že každý další student, který se v budoucnu zapojí do vývoje má automaticky k tomuto verzovacímu sytému přístup, což je pro nás velká výhoda.

3.1.7 Produkční server

Systém poběží na stejném školním serveru jako systém aktuální. Pro nový projekt jsme zvolili doménu dbs2.fit.cvut.cz, protože se počítá s dočasným paralelním během obou systému. Předpokládáme, že později se starý systém vypne a nový bude přesunut na doménu dbs.fit.cvut.cz. Z tohoto důvodu je velmi vhodné, aby byly oba systémy nasazeny na stejném serveru. V rámci udržování aktuálnosti softwaru na serveru jsme aktualizovali základní programy na jejich nejnovější stabilní verze. Tento podpůrný software využívá jak původní, tak stávající systém. Jejich kompatibilitu s oběma verzemi jsme důkladně otestovali. K datu 27. 4. 2016 jsou parametry serveru jsou následující:

CPU Intel® Xeon® CPU E5-2630 @ 2.30 GHz

RAM 8 GB

HDD 50 GB

OS Debian GNU/Linux 8.4 64-bit

Web server Apache 2.4.10

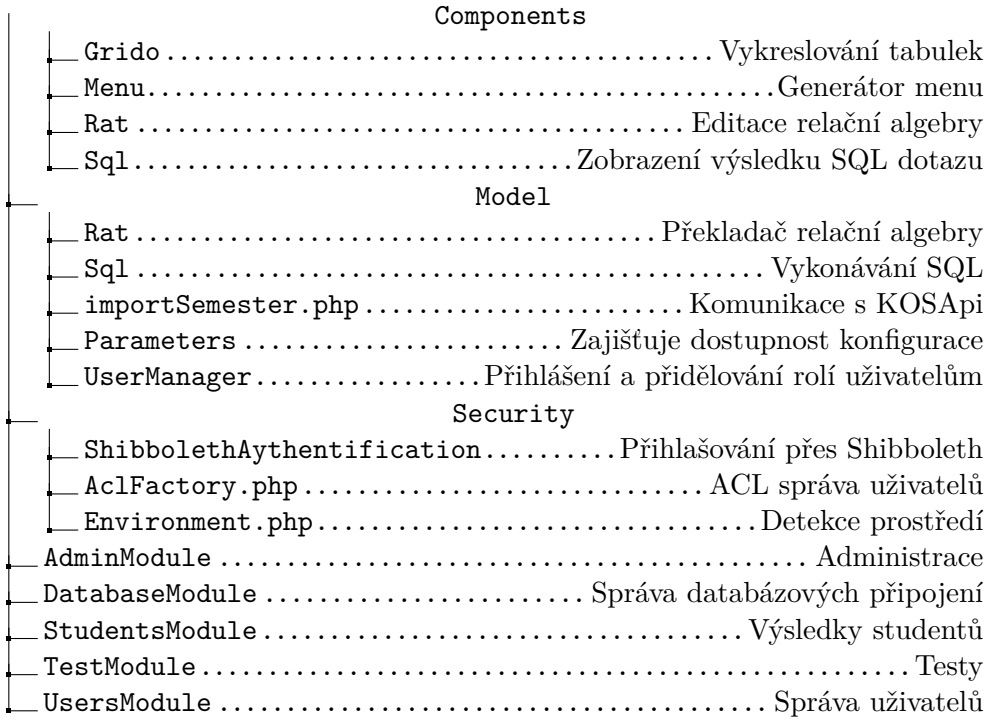
Databáze PostgreSQL 9.4.4

PHP 5.6.19

Obrázek 3.2: Parametry produkčního serveru

3.2 Struktura projektu

K zajištění snadné rozšiřitelnosti systému jsme systém rozdělili do jednotlivých modulů. Vytvořili jsme základní funkcionality jako je přihlašování do systému, správa uživatelů, jejich role, oprávnění k různým částem systému a sdílené podpůrné prostředky.



Obrázek 3.3: Strom zdrojového kódu

Struktura zobrazena výše je pouze částí struktury celého projektu. Důležité je říci, že všechny komponenty projektu respektují rozdělení na model, view a controller. Tento přístup pomůže při orientaci v projektu nezainteresovaným vývojářům, což byl jeden z našich hlavních cílů.

3.3 Návrh databáze

Návrh datsbáze jsme shledali jako jednu z největších slabin původního systému. Hlavním důvodem nekorektnosti návrhu byl způsob vývoje systému. Dva hlavní moduly, kterými jsou modul pro semestrální práci a modul pro testování byly navrhnuty a vyvíjeny dvěma nezávislými týmy. Po sloučení těchto projektů došlo ke zbytečné redundanci dat. Následující úpravy tento stav ještě zhoršily.

Při návrhu databáze pro nový systém jsme se snažili o co nejmenší duplikaci dat a homogenní pojmenování všech objektů v databázi. Pro zaručení jednotnosti pojmenování jsme zvolili prefixy. Tabulky, které obsahují data společná pro celou aplikaci jsou bez prefixu. Všechny ostatní tabulky jsou pojmenovány s prefixem dle modulu pod který spadají, takže například objekty související s modulem semestrální práce mají prefix `sw_` a pro modul databázových připojení `db_`. Oproti původnímu návrhu došlo k prokazatelnému zjednodušení o čemž vypovídá i počet tabulek, který klesl z 87 na 47.

Základní entitou pro uložení semestrální práce je tabulka `sw_sw`, na ní jsou přes cizí klíč napojeny všechny komponenty semestrální práce. Každá komponenta má svojí vlastní tabulku pro jednoduchou orientaci v datové struktuře. Druhou velmi důležitou entitou je odevzdání, které reprezentuje tabulka `sw_turn`, na kterou jsou opět navázány tabulky pro každou část semestrální práce s předponou `sw_turn_`. Do této skupiny tabulek se překlopí aktuální stav semestrální práce při odevzdání a díky tomuto návrhu nedojde ke ztrátě žádných dat při odevzdání. Samozřejmě za předpokladu správné implementace aplikační logiky.

Vzhledem k velikosti diagramu, který popisuje a jasně definuje celou strukturu návrhu ho nebudeme vkládat přímo do této práce, ale je dostupný na přiloženém datovém médiu ve formátu SVG.

3.4 Modelový průchod

Pro úplné pochopení všech součástí a funkcionalit systému popíšeme modelový průchod semestrem pro jednotlivé uživatele v daných rolích. Některé funkce systému jsou společné pro více rolí a budou popsány pouze u jedné role.

3.4.1 Administrátor

Administrátor, neboli root je jediný implicitní uživatel v systému. Root má oprávnění na všechny akce v systému. Jeho hlavním úkolem je naimportovat aktuální semestr. Poté co je semestr naimportován má možnost upravovat role uživatelů v daném kurzu. Každá role má svá oprávnění, které je možno také upravit. Předpokládáme však, že změna i úprava rolí nebude v průběhu semestru potřeba, protože v případě změny paralelky studenta, učitele dané paralelky, či termínu zkoušky v KOSu může root jedním kliknutím celý semestr zaktualizovat a změny v systému se provedou opravdu jen tam kde mají. Obvykle před začátkem následujícího semestru může root tento semestr naimportovat do systému. Po každé aktualizaci dat z KOSu se zobrazí přehledný log (Obrázek B.8).

Při importu semestru dojde i k vložení jednotlivých iterací s jejich požadavky na semestrální práci. Tyto požadavky jsou definovány v přehledném konfiguračním souboru aplikace. Defaultní požadavky pro každou iteraci odpo-

vídají aktuálnímu semestru (B152) a jsou kompletně editovatelné z přehledné administrace v průběhu semestru.

Z hlediska bezpečnosti systému je důležité, aby se jako root nemohl přihlásit běžný uživatel. Proto jsme vytvořili speciální přihlašování jen pro administrátora systému. Jeho heslo není uloženo v databázi aplikace, ale přímo ve zdrojovém kódu systému a to v nečitelné formě v tak zvané hash formě. Tímto jsme velmi ztížili možnost krádeže hesla, či jeho změny přímo na produkčním serveru a následné přihlášení útočníka.

3.4.2 Student

Po prvním přihlášení uživatele s rolí student se automaticky vybere nejaktuálnější kurz ve kterém je student zapsán. Pokud je student zapsán ve více kurzech může si samozřejmě kurz změnit a tato možnost je dostupná z roletkového menu v celém systému. Dále, se bez nutnosti jakékoliv interakce studenta automaticky vytvoří pracovní verze semestrální práce vázaná na daný kurz a student může okamžitě začít na své semestrální práci pracovat.

3.4.2.1 Tvorba semestrální práce

Pro usnadnění tvorby semestrální práce má student možnost naimportovat ukázkovou semestrální práci ve formátu XML nebo jakoukoliv jinou ve správném formátu. Tento import není vyžadován a student může začít tvorbu i bez něj. Dále je také možné naimportovat ZIP soubor projektu z nástroje Oracle SQL Developer Data Modeler[9], který je momentálně vyžadován hlavně pro kontrolu počtu entit v konceptuálním modelu. Celou tvorbu semestrální práce zajišťuje editor. Editor je rozdělen na jednotlivé sekce, které odpovídají požadovaným komponentám semestrální práce. Pokud uživatel provede nějakou akci v editoru dostane vždy od systému informaci o úspěchu, či neúspěchu dané operace.

Předpokládejme, že student bude pracovat na semestrální práci v souladu s danými požadavky na jednotlivé iterace. V této kapitole se budeme zabývat pouze tvorbou. Samotné odevzdání v každé iterace je rozebráno v kapitole Odevzdání. Tyto požadavky zjistí na EDUXu nebo přímo v systému, kde jsou zobrazeny i s podrobným popisem. Pro splnění první iterace vyplní titulek a popisek. Dále vytvoří textový popis tří dotazů. Kompletní stav své práce si může zobrazit na domovské stránce modulu semestrální práce, kde je možno práci exportovat do formátu, který je v souladu s požadavky na odevzdání v předchozích bězích předmětu Databázové systémy.

V druhé iteraci je jedním z požadavků obrázek konceptuálního modelu. Tento obrázek student nahraje v jakémkoliv podporovaném formátu a po nahrání vidí i jeho náhled přímo v editoru. Pokud je to nutné vyplní diskuzi smyček u které je zobrazen i konceptuální model a nahraje zdrojový soubor z Oracle SQL Data Modeleru. V budoucích verzí aplikace počítáme i s mož-

ností tvorby konceptuálního modelu pomocí nástroje, který byl vytvořen studentem Jiřím Slavotínkem v rámci jeho bakalářské práce. Dále musí vytvořit minimálně dalších 7 dotazů v přirozeném jazyce. Některé z těchto dotazů by měly pokrývat kategorie C a D. Nastavení těchto kategorií musí student zatím provést ručně. Automatický analyzátor kategorií dotazů tyto kategorie neumí přidělit, ale na jeho vylepšení a schopnosti pokrýt všechny požadované kategorie aktuálně usilovně pracujeme v rámci předmětu BI-SP1.

Ve třetí iteraci je hodnocena kompletní semestrální práce. Zde je vyžadováno minimálně 25 dotazů v SQL jazyce a 10 v relační algebře. V této iteraci se již naplno projeví výhody systému. Úkolem studenta je vložit skript, který vytvoří tabulky a další elementy v databázi a také skript, který jí naplní daty. Tyto skripty mají za úkol připravit databázi na vykonávání dotazů. V případě, že student skript provede, dostane jasné informace o tom, jak vykonání skriptu proběhlo a může si ověřit jeho správnost. V editoru dotazů student zadá například formulaci dotazu v relační algebře. Systém je schopen dotaz přeložit do SQL a provést dotaz na studentem zvolené databázi. Více o databázových připojení v kapitole Připojení k databázi. Student může provádět dotazy nad databází jak pomocí formulace v relační algebře, tak pomocí SQL. Pokud provede obojí najednou, systém porovná výsledky a sdělí mu jestli jsou stejné nebo ne.

3.4.2.2 Odevzdání

Odevzdání je rozděleno na dvě fáze. První z nich je zkušební odevzdání. Student zvolí iteraci, ve které chce svoji semestrální práci odevzdat a systém provede simulaci reálného odevzdání. Studentovi vypíše všechny splněné i nesplněné požadavky dané iterace. Pokud je student s výsledkem spokojen může práci předat k hodnocení. Touto akcí přejde odevzdání do druhé fáze, kdy je kompletní stav semestrální práce uložen i s výsledky dotazů. Student může ke každé části semestrální práce přidat svůj komentář. Takto odevzdaná práce se i s komentáři od studenta zobrazí učiteli. Pokud není student spokojen se svým posledním odevzdáním, může učitele požádat o povolení odevzdat práci znovu. Této žádosti může a nemusí být vyhověno.

Všechny své předchozí odevzdání si může student zobrazit v sekci předešlá odevzdání. Zde nalezne samotné hodnocení v dané iteraci a případné komentáře k opravě od učitele.

3.4.3 Učitel

Uživatel v roli učitele má několik specifických funkcionalit. Jednou z těchto funkcí je možnost přihlásit se za jakéhokoliv studenta. Tato možnost je určena pro co nejefektivnější řešení problému, na které můžou studenti narazit. V momentě, kdy uživatel narazí na nějaký problém jak se samotným systémem, tak se svou semestrální prací, kontaktuje učitele a ten se může přihlásit jako daný

uživatel. V této chvíli získává učitel naprosto stejné prostředí jako student a může velmi rychle odhalit problém.

Učitel má dostupnou kompletní tabulku hodnocení jednotlivých studentů, ve které lze snadno filtrovat, ať už podle jména nebo dané paralelky. Uživatel v této roli může také nastavit termíny jednotlivých iterací pro paralelky, které vyučuje. Toto nastavení probíhá dvěma způsoby. Prvním z nich je manuální nastavení každé paralelky zvlášť. Druhý způsob je mnohem efektivnější a to tím, že systém akceptuje soubor ve formátu CSV a dle daného formátu tohoto souboru nastaví termíny odevzdání dávkově.

Nejdůležitější činností učitele je samotná oprava semestrálních prací. Pokud proběhne odevzdání semestrální práce některým ze studentů, kteří spadají pod paralelku daného učitele, zobrazí se toto odevzdání učiteli. Před tím, než učitel zahájí opravu si může semestrální práci prohlédnout v náhledu nebo ji stáhnout jako archiv ve formátu, který koresponduje s předešlou formou odevzdávání v předmětu Databázové systémy. Jakmile přistoupí lektor k opravě, odevzdání se uzamkne. V opravě se zobrazí kompletní semestrální práce i s historií předešlých odevzdání u jednotlivých komponent semestrální práce. Pokud nevyhovuje hodnocení od automatu, může toto hodnocení učitel změnit a v případě potřeby přidat komentář, který se zobrazí studentovi v hodnocení.

Učitel ve výchozím nastavení vidí pouze své paralelky a odevzdané práce svých studentů. Systém umožňuje v případě nutnosti jako je například nemoc, zobrazit a opravit práce i jiných, než svých studentů.

3.4.4 Garant

Garant předmětu může upravovat požadavky a celkové maximální bodové hodnocení semestrální práce a zobrazit si výsledky všech studentů. Množinu jeho oprávnění tvoří všechny možnosti, které mají ostatní role, kromě role administrátora.

3.4.5 Připojení k databázi

Modul připojení k databázi slouží jak učitelům, tak studentům. Umožňuje každému uživateli vytvořit a uložit několik připojení ke vzdáleným databázím. Tyto databáze poté student používá při testování své semestrální práce a učitel při vytváření testů při automatickém vyhodnocení správnosti odpovědi. Jedna databáze může být nastavena jako výchozí a připojení k této databázi proběhne automaticky při přihlášení uživatele.

Pro uložení databáze je vyžadováno i přístupové heslo. Toto heslo je uloženo v zašifrované formě a tím je tedy minimalizované riziko, že by třetí strana mohla získat přístupové údaje k dané databázi. Výhodou tohoto řešení je, že uživatel nemusí při každém připojení vyplňovat heslo znovu.

Kapitola 4

Realizace

V této kapitole se budeme podrobně věnovat implementaci a výhodám jednotlivých komponent systému oproti původnímu řešení. Cílem této kapitoly není popsat kompletní implementaci všech součástí systému, ale zmínit ty nejdůležitější změny oproti stávající verzi.

4.1 Společný základ

Společným základem jsou myšleny ty funkcionality systému, které jsou naprosto nezbytné pro funkčnost sytému jako takového a je u nich předpoklad, že budou využity jak v existujících modulech systému, tak v budoucích rozšířeních. To byl hlavní důvod, aby jsme tyto komponenty systému implementovali obecně a naprosto nezávisle na ostatních částech systému.

4.1.1 Import dat z KOSApi

V původním systému byla implementace modelu pro získání dat z KOSApi[7] rozdělena do několika samostatných podmodelů. Navíc, bylo možno importovat zvlášť paralelky, uživatele a další subdata. Při tomto postupu často docházelo ke ztrátě integrity dat a k vytváření duplicit v databázi. Proto jsme vytvořili jeden model, který naimportuje kurz v semestru jako celek a je schopen data jednoduše aktualizovat. Celá správa kurzu v daném semestru se tedy zúžila na jednu jedinou akci administrátora. Model je schopen si velmi sofistikovaně všechny konflikty a aktualizace vyřešit sám. Všechny úpravy, které model provede jsou zaznamenány do strukturovaného logu, který se v systému po provedení importu administrátorovi zobrazí.

4.1.2 Connector

Třídu `Connector` jsme vytvořili jako abstraktní třídu, která slouží jako základ pro další specifické konektory. Tyto konektory, například `OracleConnector`

jsou navrženy k připojení ke vzdáleným databázím. Třída byla definována jako abstraktní, protože musí implementovat základní metody pro práci s obecným připojením, jako je odstranění komentářů z SQL dotazu, převod kódování a tak dále. Zároveň však musí u derivovaných tříd požadovat implementaci nezbytných metod, jako je `execute()`, která vykoná příkaz nad databází a je různá u rozličných databázových strojů. Tímto přístupem jsme zajistili jednoduché a univerzální začlenění jakékoliv externí databáze do projektu. V budoucnu se počítá například s MySQL.

4.1.3 Result

Třída `Result` reprezentuje obecný výsledek SQL dotazu nad databází. Tento výsledek je různý pro dané typy dotazů. Například u SELECT dotazu požadujeme zobrazení výsledku v přehledné tabulce nebo v dotazu typu UPDATE nás zajímá pouze úspěch či neúspěch operace. Proto jsme vytvořili tuto třídu jako základ pro všechny standardní typy dotazů. Dále jsme naimplementovali třídy, které zodpovídají každá za svůj druh dotazu a jejich vnitřní chování je specifické pro daný typ dotazu, zatímco vykreslení může být jednotné.

4.2 Modul semestrální práce

Tento modul byl jedním z hlavních témat této bakalářské práce a proto jsme dali jeho návrhu i implementaci jednu z největších priorit v celém projektu. Jak je zmíněno v kapitole Semestrální práce, celá semestrální práce se skládá z několika komponent, proto jsme vytvořili jeden základní model `SemesterWork` a pro každou dílčí komponentu zvláštní model, který reprezentuje danou entitu v semestrální práci. Toto rozčlenění umožňuje snadné přidání dalších entit v případě nutnosti rozšířit požadavky předmětu Databázové systémy na semestrální práci, protože jednotlivé komponenty jsou na sobě absolutně nezávislé. Samotný model `SemesterWork` je dokonce nezávislý i na zdroji dat, ze kterého je načítán a posléze ukládán. Za tímto účelem byly vytvořeny modely `SemesterWorkDb` zajišťující spolupráci s databázovým úložištěm, `SemesterWorkFs` umožňující načtení a uložení práce ze souborového systému a `XmlParser`, který slouží k importu semestrální práce z XML dokumentu ve kterém se odevzdávala semestrální práce v minulých bězích předmětu BI-DBS.

4.2.1 Komponenty

Všechny logické celky semestrální práce jsme rozdělili na jednotlivé komponenty, které mohou být použity kdekoliv v rámci aplikace. V případě potřeby například zobrazení dotazu vývojář pouze definuje, že se zde má tento prvek zobrazit a všechny akce, které je možné s danou komponentou provádět jsou implementovány jen jednou a na jednom konkrétním místě. Toto rozdělení

na komponenty považujeme za jeden z největších přínosů oproti stávajícímu systému, protože velmi usnadní budoucí vývoj portálu.

4.2.2 Odevzdání

Odevzdání patří k jedné z nejsložitějších a nejkomplexnějších částí této bakalářské práce. Při požadavku na odevzdání, či zkušební otestování je požadavek zařazen do fronty. Tuto frontu zpracovává model `Submitter`, který je vyvolán každým požadavkem na odevzdání a zároveň automaticky v intervalu 10 minut. Tento interval je jednoduše nastavitelný a předpokládáme, že se jeho hodnota bude ještě měnit, dle zkušeností při provozu systému v reálném prostředí.

Pokud fronta není prázdná zkopíruje `Submitter` všechna data, která souvisí s danou semestrální prací do oddělené databázové struktury, která je blíže popsaná v kapitole Návrh databáze. Poté si vyžádá připojení k databázi. Momentálně má aplikace vyhrazených deset systémových databází na serveru oracle.fit.cvut.cz a je tímto tedy omezena na tento počet možných paralelních odevzdání. Pokud jsou všechny databáze obsazené, tak systém nic nedělá. V případě, že je nějaká databáze dostupná, tak ji systém zamkne, aby nedošlo k žádným konfliktům a provede dané operace nad touto databází. K těmto operacím patří spuštění create a insert scriptu a provedení všech studentových dotazů. Výsledky těchto dotazů jsou uloženy ve formě serializovaných datových objektů na server. Tím je zaručeno, že stav semestrální práce je kompletně zachycen v daný moment a je možné dohledat a zobrazit historii všech odevzdání dané semestrální práce. Pokud vše proběhne bez chyby, je odevzdání označeno jako odevzdané a vyjmuté z fronty. V případě jakékoliv chyby v celém procesu je tato chyba uložena do logu a systém se pokusí o zpracování požadavku z fronty později.

4.3 Testování

Testování bylo v původním projektu úplně opomenuto a to hlavně z časových důvodů. Přitom se jedná o jeden ze základním nástrojů pro minimalizaci chyb při úpravách a dalším vývoji existujícího softwarového řešení. Testování lze samozřejmě provádět ručně. To je však velmi neefektivní jak časově, tak z hlediska důvěryhodnosti výsledku testování. Ideálním řešením je automatické testování. Pro automatické testování se využívá mnoho postupů a technologií. Pro tento projekt jsme zvolili unit testy a Selenium IDE. Pro spouštění testů jsme vytvořili jednotný skript, který postupně provede jednotlivé testy a vypíše informace o jejich úspěchu nebo případném neúspěchu. Tento skript je v této chvíli připraven na spuštění po nahrání úprav na produkční server. Při vložení nového obsahu do repozitáře se spustí tento script a v případě úspěchu se server aktualizuje. V případě opačném se vytvoří log o neúspěchu a server nebude aktualizován.

4.3.1 Unit testy

Unit testy jak již název napovídá slouží k otestování nějaké jednotky. Za tuto jednotku může být považována třída, jedna její funkce nebo funkční celek aplikace. Ke spouštění těchto testů jsme zvolili Nette Tester[27]. Výhodou tohoto řešení je jeho integrace přímo v samotném Nette frameworku, na kterém je aplikace postavena. Těmito testy jsme pokryli některé části aplikace, jako je import semestru, přihlašování a odhlašování uživatelů, import semestrální práce a její odevzdání. Momentálně není těmito testy pokrytá celá aplikace, ale předpokládáme, že tyto testy budou postupně doimplementovány a rozšiřovány. Testy probíhají nezávisle na sobě a nejsou závislé na aktuálních datech ani jiných částech aplikace. To je samozřejmě jistou výhodou, ale tím pádem nemohou zajistit komplexní otestování aplikace jako celku. Nakonec, to není ani jejich účelem. K tomu jsme využili Selenium IDE.

4.3.2 Selenium IDE

Selenium IDE[28] je nástroj umožňující spuštění předem vytvořených scénářů, které provedou nějakou akci v aplikaci, stejně jako by to provedl uživatel. Tento test probíhá přímo v prohlížeči a je tedy maximálně totožný s interakcí reálného uživatele se systémem. Těmito testy jsme pokryli komplexnější funkce systému, jako je třeba kompletní tvorba semestrální práce. Po spuštění těchto testů je dokonce možné jejich průběh přímo sledovat v nově otevřeném okně prohlížeče. Po skončení testů je vygenerován textový log stejně jako v případě unit testů.

4.4 Dokumentace

Dokumentace je automaticky generována přímo ze zdrojových kódů. Ke generování dokumentace systém používá nástroj Apigen[29]. Tento systém byl vybrán z několika různých generátorů na základě přehlednosti a jednoduchosti výsledné dokumentace. Dokumentace se vygeneruje jako HTML stránka. Dokumentaci jsme rozdělili na dvě části. První část pokrývá pouze vlastní kód aplikace (Obrázek B.9) a je tedy velmi přehledná. Druhá část je kompletní dokumentací. Generuje se i ze zdrojových kódů všech použitých knihoven a externích nástrojů včetně samotného Nette frameworku. Důvodem toho rozdělení byla jednodušší orientace vývojářů v aplikační dokumentaci při řešení problému se samotnou aplikací.

4.5 Shrnutí

Všechny tyto jednotlivé součásti implementace a podpůrné nástroje jsme velmi pečlivě zdokumentovali v projektové wiki. Výše popsané prvky jsou základním

stavebním kamenem pro snadný budoucí rozvoj systému, ke kterému jsme směřovali.

Závěr

Cílem této bakalářské práce bylo provést analýzu již existujícího systému na podporu výuky v předmětu Databázové systémy a na základě této analýzy navrhnout a implementovat systém nový, který zodpovídá ze semestrální práci. Systém měl být navíc pokrytý testy a připravený na další vývoj v rámci budoucích běhů předmětů BI-SP1 a BI-SP2. Tyto cíle byly splněny.

Povedlo se nám společně s Petrem Pejšou a všemi členy týmu vyvinout jednotný a robustní systém, který je dobře zdokumentován. Hlavní výhodu oproti systému stávajícímu vidíme v kvalitní implementaci a čistém zdrojovém kódu bez zbytečných a nesrozumitelných částí.

Předpokládáme, že systém bude nasazen již v příštím zimním semestru akademického roku 2016/2017, kdy dojde k nalezení a opravě posledních chyb na základě zkušeností z reálného provozu. Věříme, že těchto nedostatků nebude nalezeno mnoho, protože většina systému a všechny jeho důležité komponenty jsou pokryty automatickými testy, které již nyní zaručují otestování funkcionality hlavních částí systému.

Kromě samotné implementace jsme vytvořili unifikovaný proces vývoje a upravování produkční verze, který minimalizuje možnost nefunkčnosti systému po případné úpravě vývojářem. Celý tento proces a kompletní dokumentace od nastavení vývojového prostředí, až po samotné nahrání úpravy na server jsou přehledně zdokumentovány v projektové wiki.

Během návrhu a vývoje nás napadla mnohá vylepšení a rozšíření funkcionalit oproti původnímu návrhu. Například implementace SQL parseru pro lepší zpracování dotazů, automatické ukládání rozpracované semestrální práce, rozšíření podpory na více databázových strojů a mnoho dalších. Tyto nápady jsme sepsali v projektové wiki a na jejich implementaci se budou podílet budoucí týmy.

Vzhledem k mé velké zainteresovanosti jak na vývoji tohoto systému, tak na údržbě systému původního, bych rád u projektu zůstal, alespoň jako konzultant.

Literatura

[1] Ing. Ivan Halaška: Tabulka pokrytí dotazů [BI-DBS Databázové systémy] [online]. Říjen 2015, [vid. 29. 3. 2016]. Dostupné z: `https://edux.fit.cvut.cz/courses/BI-DBS/project/queries`

[2] Ing. Jiří Hunka: Jiří Hunka [BI-DBS Databázové systémy] [online]. Únor 2016, [vid. 17. 2. 2016]. Dostupné z: `https://edux.fit.cvut.cz/courses/BI-DBS/teacher/hunkajir/start`

[3] Sýkora, J.: *Podpora automatizované kontroly semestrální práce z předmětu Databázové systémy*. Bakalářská práce, České vysoké učení technické v Praze, 2015.

[4] Autor neznámý: Popis předmětu - BI-DBS [online]. Duben 2016, [vid. 12. 4. 2016]. Dostupné z: `http://bk.fit.cvut.cz/cz/predmety/00/00/00/00/00/00/01/12/24/p1122406.html`

[5] Pokorný, J.; Valenta, M.: *Databázove systémy*. Praha: České vysoké učení technické v Praze, 2013, ISBN 9788001052129.

[6] Bittner, K.: *Use case modeling*. Boston, MA: Addison Wesley, 2003, ISBN 0201709139, 3–4 s.

[7] FIT ČVUT: KOSapi [online]. 2016, [vid. 29. 3. 2016]. Dostupné z: `https://kosapi.fit.cvut.cz`

[8] Muller, R.: *Database design for smarties : using UML for data modeling*. San Francisco, Calif: Morgan Kaufmann Publishers, 1999, ISBN 1-55860-515-0, 15 s.

[9] Autor neznámý: SQL Developer Data Modeler [online]. 2016, [vid. 4. 5. 2016]. Dostupné z: `http://www.oracle.com/technetwork/developer-tools/datamodeler/overview/index.html`

[10] Larman, C.: *Applying UML and patterns : an introduction to object-oriented analysis and design and iterative development*. Upper Saddle River, N.J: Prentice Hall PTR, 2005, ISBN 0131489062, 110 s.

[11] Holscher, E.; Howard, T.: A beginners guide to writing documentation [online]. 2013, [vid. 6. 4. 2016]. Dostupné z: `http://docs.writethedocs.org/writing/beginners-guide-to-docs/`

[12] Lang, J.-P.: .

[13] Naur, P.; Randell, B.: *Software Engineering*. Brusel, Belgie: Report on a Conference by the NATO Science Committee. NATO Scientific Affairs Division, 1968.

[14] Meyer, B.: *Object-oriented software construction*. Upper Saddle River, N.J: Prentice Hall PTR, 1997, ISBN 978-0136291558.

[15] ISRD: *Introduction to object oriented programming and C*. New Delhi: Tatamcgraw Hill, 2007, ISBN 0-07-061683-3, 3 s.

[16] Brevoort, M.: *Growing better software*. Theale: Marc Brevoort, 2008, ISBN 978-0-9559824-0-8.

[17] Autor neznámý: About the Apache HTTP Server Project - The Apache HTTP Server Project [online]. 2016, [vid. 3. 5. 2016]. Dostupné z: `https://httpd.apache.org/ABOUT_APACHE.html`

[18] Autor neznámý: Usage Statistics and Market Share of Apache for Websites, May 2016 [online]. 2016, [vid. 3. 5. 2016]. Dostupné z: `http://w3techs.com/technologies/details/ws-apache/all/all`

[19] Scott, M.: *Programming language pragmatics*. Amsterdam Boston: Elsevier/Morgan Kaufmann Pub, 2009, ISBN 978-0-12-374514-9, 826 s.

[20] Autor neznámý: Seznámení s Nette frameworkem [online]. 2016, [vid. 13. 4. 2016]. Dostupné z: `https://doc.nette.org/cs/2.3/getting-started`

[21] Autor neznámý: Rychlý a pohodlný vývoj webových aplikací v PHP | Nette Framework [online]. 2016, [vid. 13. 4. 2016]. Dostupné z: `https://nette.org/cs/`

[22] Pitt, C.: *Pro PHP MVC*. Berkeley, CA New York: Apress Distributed to the Book trade worldwide by Springer, 2012, ISBN 978-1430241645.

[23] Autor neznámý: PostgreSQL: About[online]. 2016, [vid. 1. 5. 2016]. Dostupné z: `http://www.postgresql.org/about/`

[24] Somasundaram, R.: *Git version control for everyone beginner's guide : the non-coder's guide to everyday version control for increased efficiency and productivity*. Birmingham: Packt Pub, 2013, ISBN 978-1-84951-752-2.

[25] Autor neznámý: Git [online]. 2016, [vid. 3. 3. 2016]. Dostupné z: `https://git-scm.com`

[26] Autor neznámý: Code, test, and deploy together [online]. 2016, [vid. 3. 3. 2016]. Dostupné z: `https://about.gitlab.com`

[27] Autor neznámý: Nete Tester [online]. 2016, [vid. 4. 5. 2016]. Dostupné z: `https://tester.nette.org/cs/`

[28] Autor neznámý: Selenium IDE plugin [online]. 2016, [vid. 4. 5. 2016]. Dostupné z: `http://www.seleniumhq.org/projects/ide/`

[29] Autor neznámý: Apigen [online]. 2016, [vid. 4. 5. 2016]. Dostupné z: `http://www.apigen.org`

Příloha **A**

Seznam použitých zkratek

PHP Personal Home Page

XML Extensible markup language

BI-DBS Bakalářská informatika - Databázové systémy

MVC Model-view-controller architektura

GIT Typ distribuovaného verzovacího systému

OOP Objektově orientované programování

HTML HyperText Markup Language

CSS Cascading Style Sheets

JS JavaScript

BI-SP1 Bakalářská informatika - Softwarový týmový projekt 1

BI-SP2 Bakalářská informatika - Softwarový týmový projekt 2

SRS Software Requirements Specification

SQL Structured Query Language

FIT ČVUT Fakulta informačních technologií českého vysokého technického učení v Praze

ZIP souborový formát pro kompresi dat

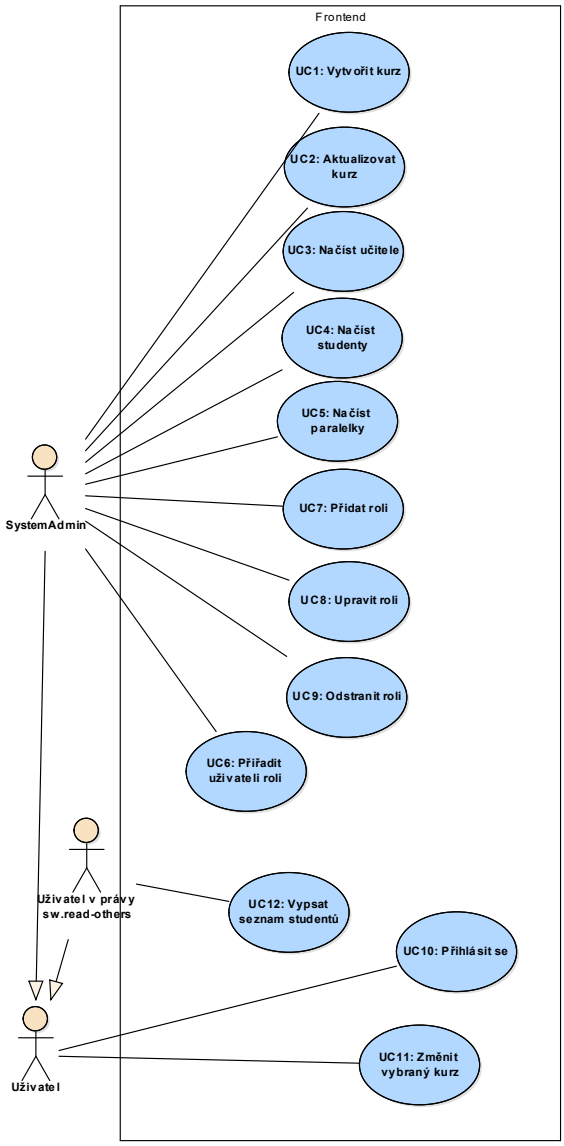
API Application Programming Interface

KOS Studijní informační systém ČVUT

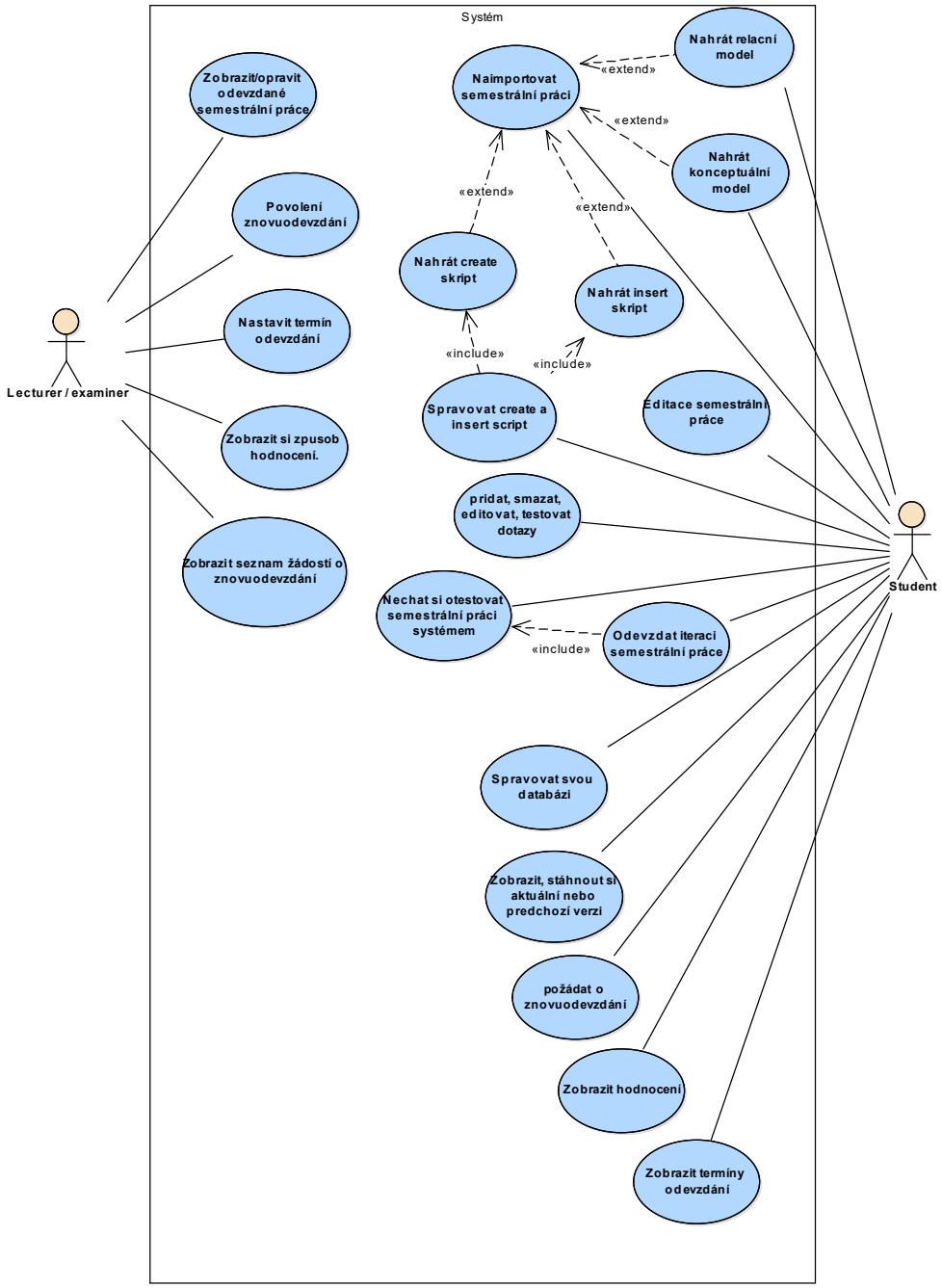
CSV Comma Separated Values

Příloha B

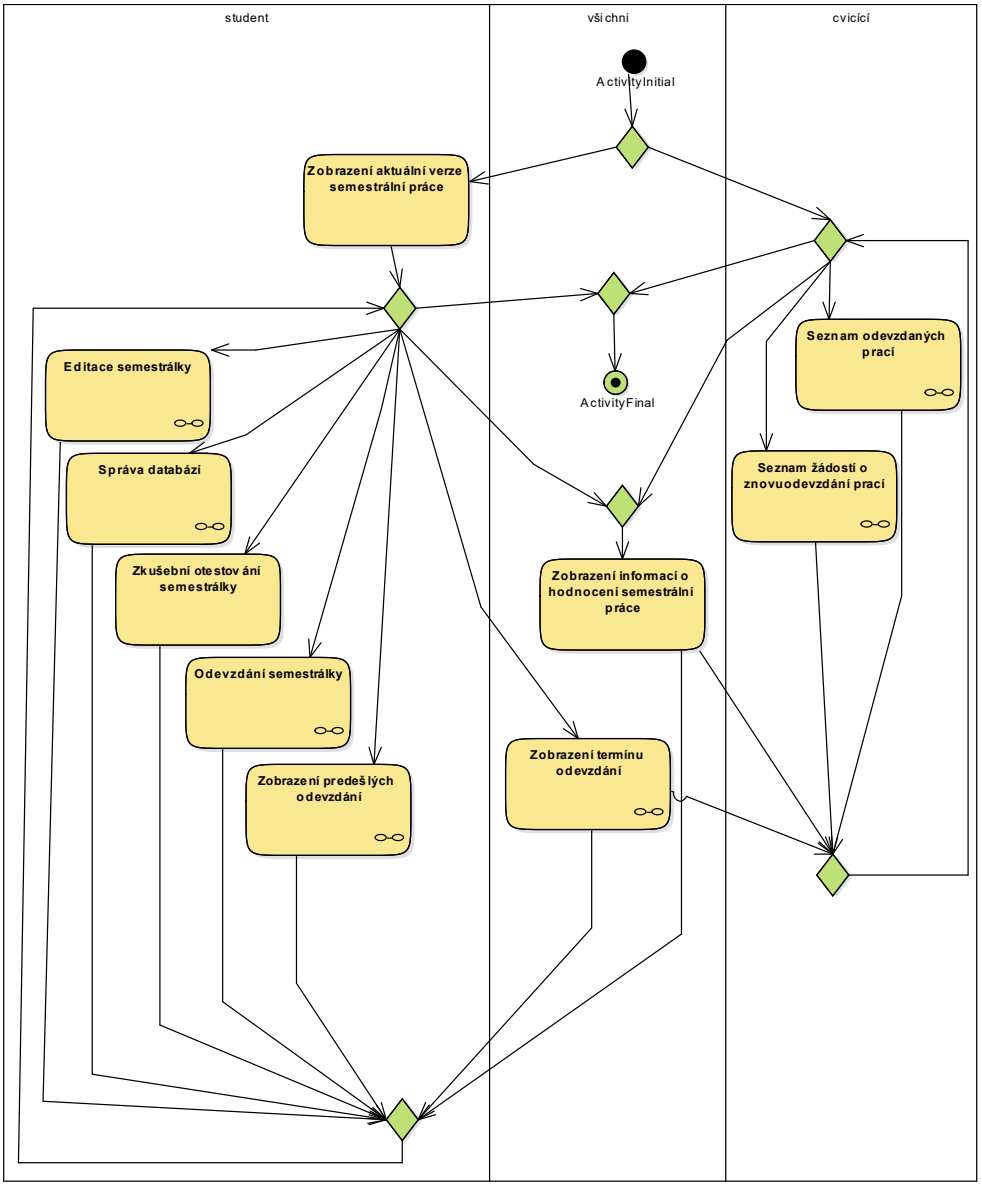
Obrázky



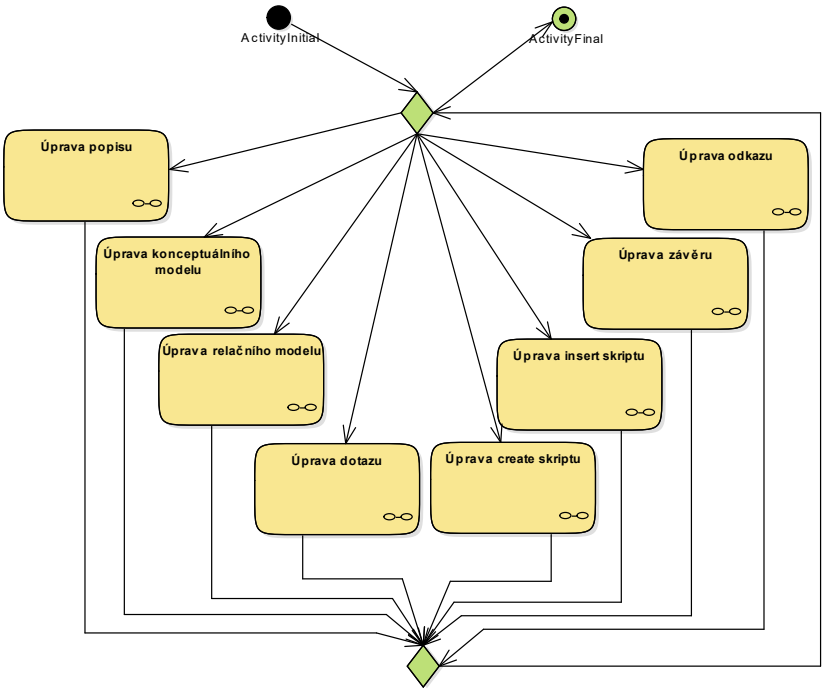
Obrázek B.1: Use case model administrace



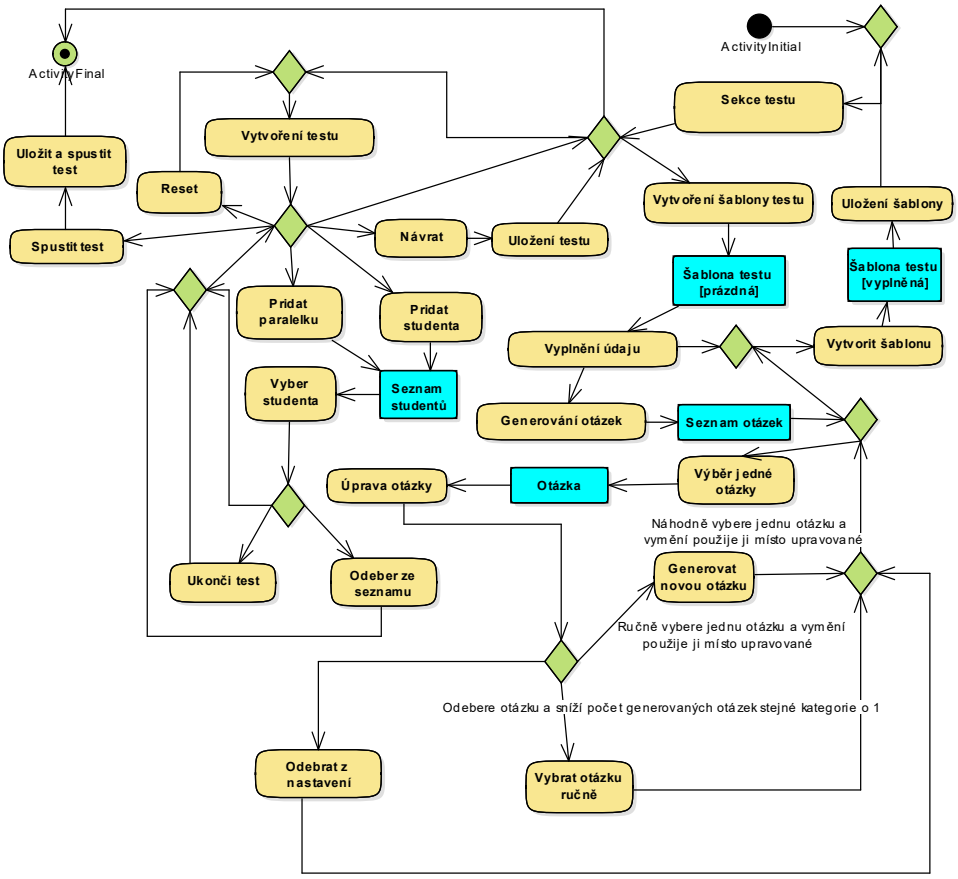
Obrázek B.2: Usecase model semestrální práce



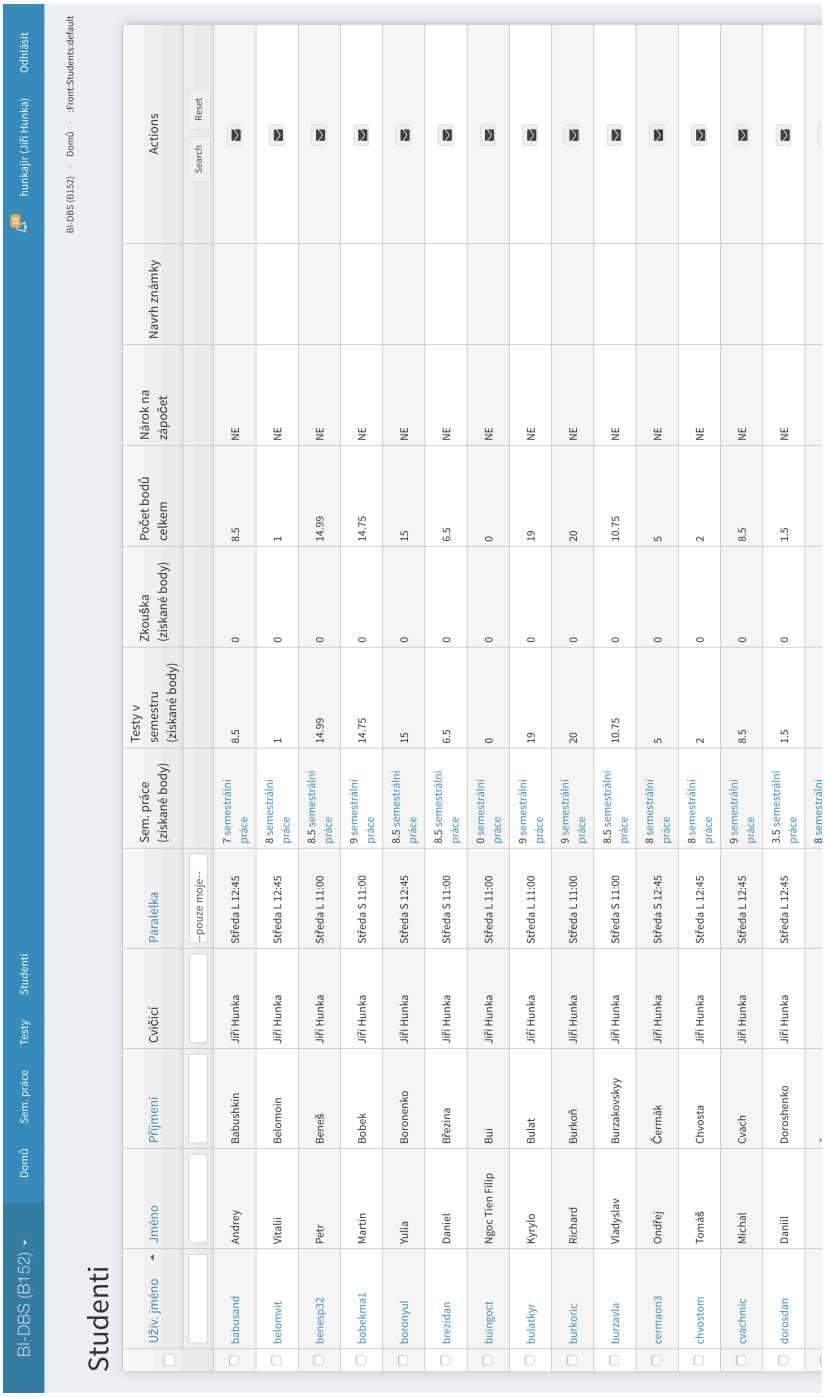
Obrázek B.3: Aktivity diagram semestrální práce



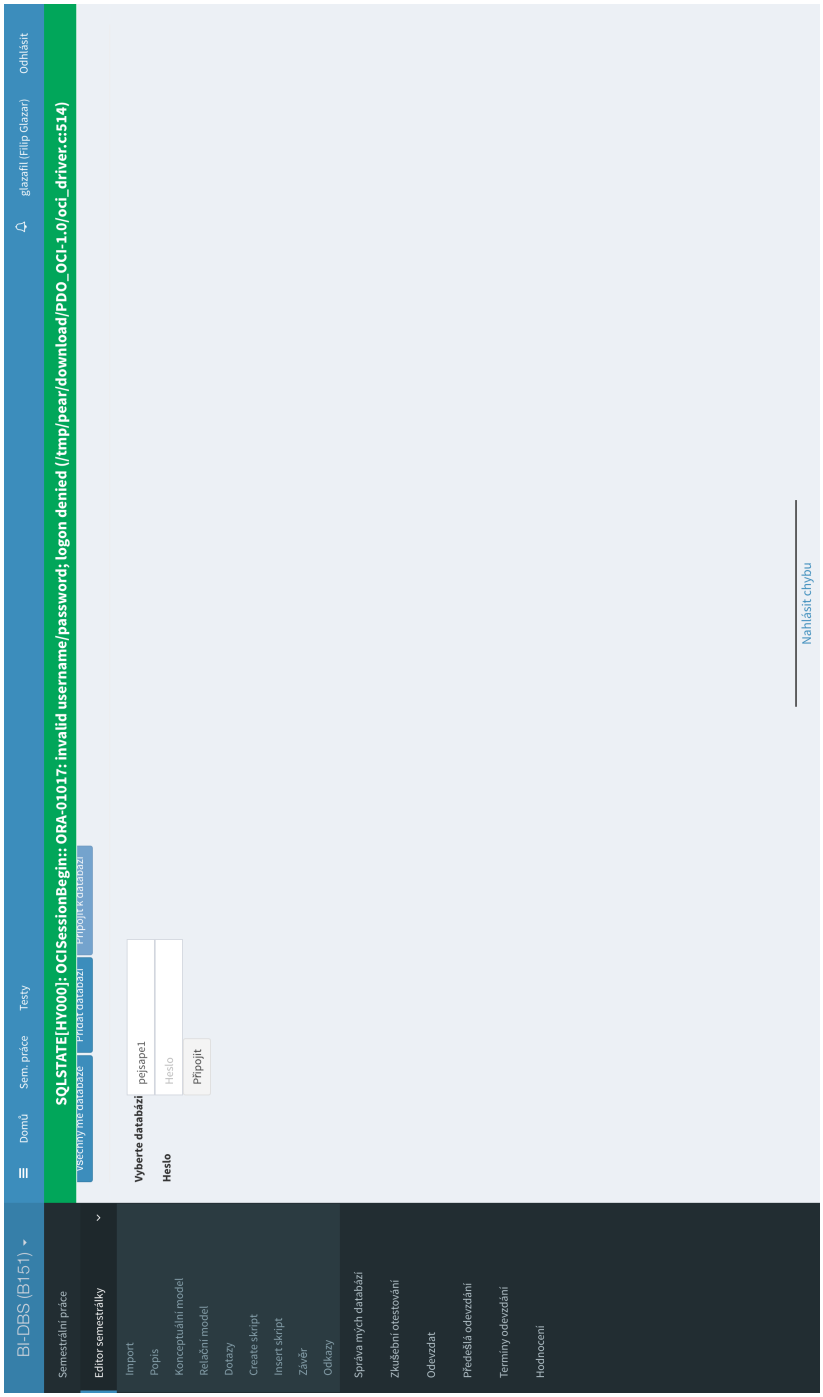
Obrázek B.4: Aktivity diagram semestrální práce manuální editace



Obrázek B.5: Aktivity diagram tvorba testu



Obrázek B.6: Tabulka hodnocení studentů



Obrázek B.7: Chybová hláška: Špatné heslo

Obrázek B.8: Log po naimportování semestru

Obrázek B.9: Aplikační dokumentace

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace - GIT
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `BP_Glazar_Filip_2016.pdf` ..... text práce ve formátu PDF
- `docs`
  - `app` ..... aplikační dokumentace
  - `full` ..... kompletní dokumentace
- `database`
  - `diagram.svg` ..... diagram systémové databáze
- `screenshots` ..... srovnání starého a nového systému