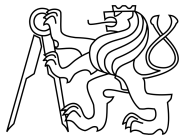
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Systém pro podporu testování studentů v BI-DBS
Student:	Petr Pejša
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, je třeba testovat studenty v průběhu semestru i u zkoušky.
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 tvorby testů a testování studentů. 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 testování studentů.
Na základě analýzy navrhněte s velkým důrazem na snadnou údržbu a rozšiřitelnost nové vnitřní uspořádání aplikace.
V rámci návrhu spolupracujte s Filipem Glazarem, který bude přepracovávat část věnující se semestrální práci.
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 28. listopadu 2015

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

Bakalářská práce

Systém pro podporu testování v BI-DBS

Petr Pejša

Vedoucí práce: Ing. Jiří Hunka

16. května 2016

Poděkování

Chtěl bych zde poděkovat panu Ing. Jiřímu Hunkovi za jeho dobré rady, odborné vedení a vstřícný přístup. Děkuji panu Ing. Ivanu Halaškovi za jeho aktivitu při testování nové aplikace, hlášení nedostatků v aplikaci a za jeho návrhy na možné rozšíření aplikace. Chci poděkovat Filipu Glazarovi, který vypracoval část systému se semestrálními pracemi a společně s Oldřichem Malecem se starají o běh systému na serveru, mé díky patří oběma. Na oba z nich jsem se mohl obrátit s jakýmkoliv problémem. Dále bych chtěl poděkovat všem studentům předmětu BI-SP1 a BI-SP2, kteří se nějak podíleli na projektu. Děkuji za pomoc s jazykovou korekturou textu své mamce Ing. Vladislavě Pejšové.

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 16. května 2016 .....

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Pejša, Petr. *Systém pro podporu testování v BI-DBS*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Tato práce se zabývá tvorbou a automatizovanou opravou testů a zkoušek v předmětu Databázové systémy, vyučovaného na Fakultě informačních technologií ČVUT v Praze. Hlavním cílem bylo nahradit stávající systém pro správu testů v semestru a u zkoušek. Velký důraz byl kladen na snadnou údržbu a rozšiřitelnost. V příloze lze nalézt dokumentaci a zdrojové kódy.

Klíčová slova Webová aplikace, Databázové systémy, automatická oprava, SQL, Relační algebra, Nette, PHP, Grido

Abstract

This work is focused on automatic corrections in the course Database systems which is taught in Faculty of Information Technology at CTU in Prague. The main purpose of this thesis was to replace current system for test management in semester and on exams. The big emphasis was put on easy maintance and extensive use. In attachment is documentation and sourse codes.

Keywords Web aplication, Database systems, automatic correction, SQL, Relational algebra, Nette, PHP, Grido

Obsah

Seznam obrázků

Úvod

Testy a zkoušky slouží k ověřování znalostí studentů. V předmětu Databázové systémy (BI-DBS) se vyučuje především návrh databáze, relační algebra, dotazovací jazyk SQL a teorie. V testech a u zkoušek jsou studenti nuceni prokázat tyto znalosti úspěšným napsáním testu.

Testy se píší především na papír. Na jednotlivé kategorie je obvykle specifický typ odpovědi. Na otázku z návrhu databáze to bývá kreslení schématu databáze, na otázku z relační algebry, resp. jazyka SQL, to bude výraz v relační algebře, resp. SQL dotaz. Na teoretické otázky bývají textové odpovědi, popřípadě výběr z nabízených odpovědí.

V současné době také existuje systém na tvorbu testů pro studenty BI-DBS [1], ale není plně funkční a díky své vnitřní neuspořádanosti je jeho rozšiřitelnost velmi komplikovaná. V tomto systému je tvorba testů zdlouhavá a uživatelsky nepříjemná. Během užívání systému se vyskytuje mnoho chyb, z nichž některé jsou drobné překlepy a jiné znemožňují chod celého systému. Odstraňování těchto chyb a následné celkové doladění je komplikované z důvodu chybějící dokumentace k aplikaci.

Současný systém byl vyvíjen v rámci předmětů Softwarový týmový projekt 1 (BI-SP1) a Softwarový týmový projekt 2 (BI-SP2), a to po dobu 2 let, za tu dobu se na něm vystřídaly 4 různé týmy. Právě toto dědění projektu po předchozí skupině bez obstojné dokumentace zapříčinilo neuspořádanost v projektu.

Cílem této bakalářské práce je analyzovat tento systém. Na základě současného stavu navrhnout řešení, které nahradí stávající systém a vyhne se současným problémům. Toto řešení naimplementovat a následně celou aplikaci řádně otestovat a zdokumentovat.

Kapitola 1

Analýza a aktuální stav aplikace pro podporu výuky BI-DBS

Pro úspěšné absolvování předmětu Databázové systémy (BI-DBS), vyučovaného na Fakultě informačních technologií (FIT) ČVUT v Praze, student musí úspěšně napsat 2 testy v semestru, vypracovat semestrální práci a poté, když splní všechny požadavky na zápočet, může být připuštěn ke zkoušce. Pro úspěšné složení zkoušky musí student dosáhnout 50 bodů v součtu ze semestru a ze zkoušky.

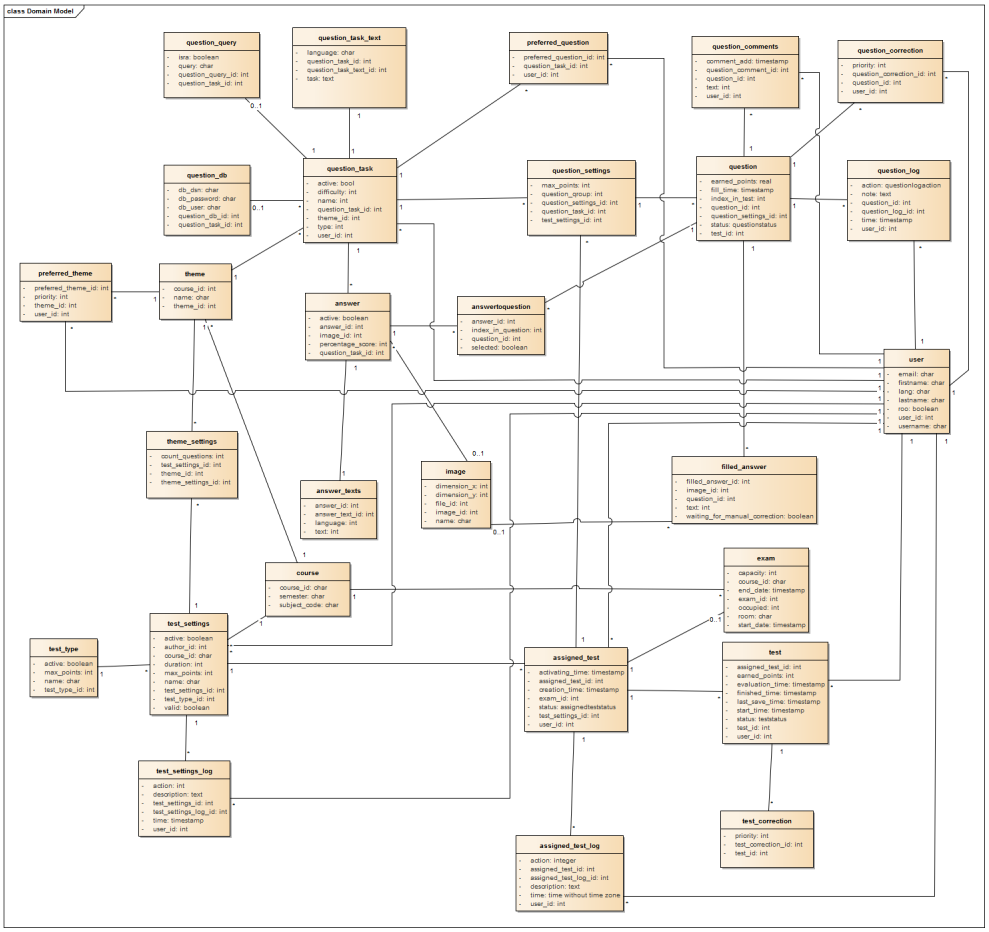
Testy se zpravidla píší na papír. Nicméně někteří vyučující využívají Systém pro podporu výuky BI-DBS [1], vytvořený v rámci předmětů Softwarový týmový projekt 1 (BI-SP1) a Softwarový týmový projekt 2 (BI-SP2). Systém není plně využíván, protože není plně funkční a není kompletně odladěn.

Systém nabízí možnost automatické opravy otázek, velkou část otázek dokáže plně vyhodnotit a obodovat (viz dále které), dále nabízí možnost opravy více stejných odpovědí na otázku, která se nevyhodnotila automaticky. Čímž opravu výrazně urychlí a zjednoduší, zároveň jsou všichni studenti se stejnou odpovědí ohodnoceni stejně.

Na druhou stranu práce se systémem není uživatelsky příjemná. Je velice chybový a nezvládá zatížení většího počtu připojených uživatelů najednou. To je velký problém, když se píše zkouška s 60 studenty a systém přitom nezvládne více než 10 připojených uživatelů.

Cílem této práce je restrukturalizovat stávající systém a odstranit jeho nedostatky, kvůli nimž není plně využíván. Proto jsem se musel seznámit s aktuálním stavem aplikace a najít její nedostatky. Na začátek bych chtěl uvést, jaké technologie jsou použity ve stávajícím systému a ukázat, jak je navržen databázový model.

Celý systém běží na frameworku Nette (verze 2.3), který je rozšířením programovacího jazyka PHP. Z Nette se používá velká část jejích komponent tj. Nette/Forms, Nette/Database, Nette/Security aj., dále se v systému jednotně používá komponenta Grido (verze 2.1) pro tvorbu tabulek. Pro ukládání



Obrázek 1.1: Schéma staré databáze

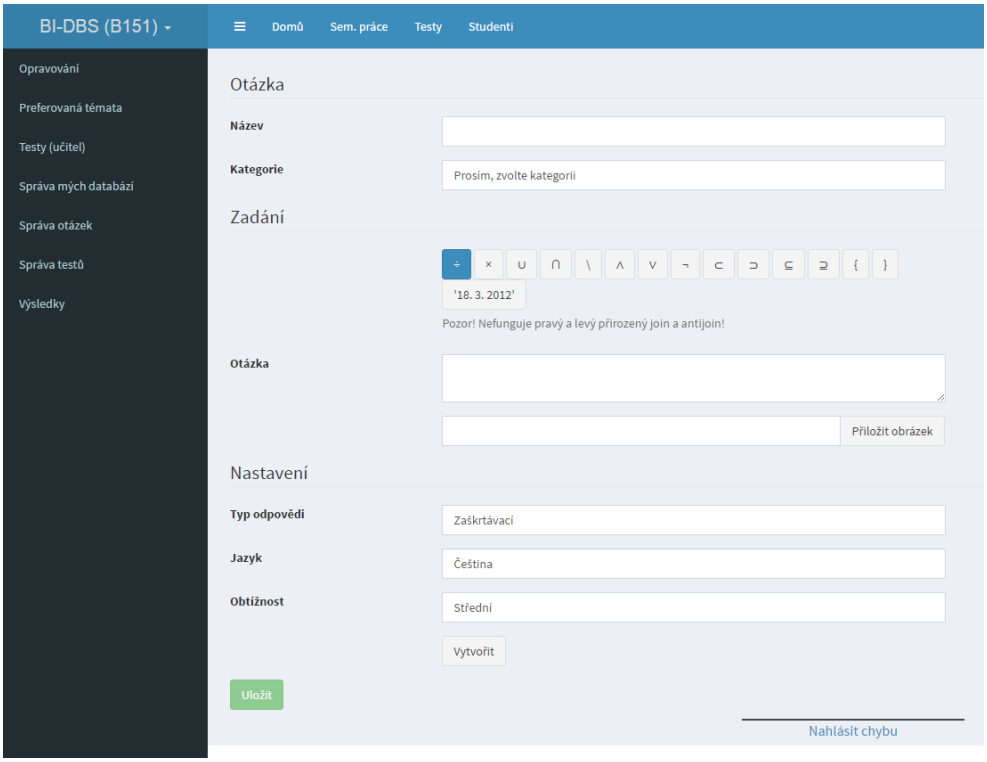
dat je zde databáze PostgreSQL. Všem těmto technologiím se budu věnovat později.

Aktuální databázové schéma je na obrázku 1.1 (Schéma nepopisuje kompletní databázi, ale pouze část zabývající se testy, tzn. vynechává část se semestrálními pracemi). Schéma je poměrně rozsáhlé a komplikované, pokusím se ho objasnit po částech v následujících kapitolách.

1.1 Tvorba otázky

Před tvorbou testu je nejprve nutné vytvořit sadu otázek, ze kterých se následně sestaví test. Tímto způsobem lze jednu otázku přiřadit do více testů. Kdyby byl nejprve vytvořen test a následně k němu teprve vytvářena otázka, tak by se musely tvořit pro každý test nové otázky.

Ve stávajícím systému se vytvoří otázka vyplněním formuláře - viz ob-



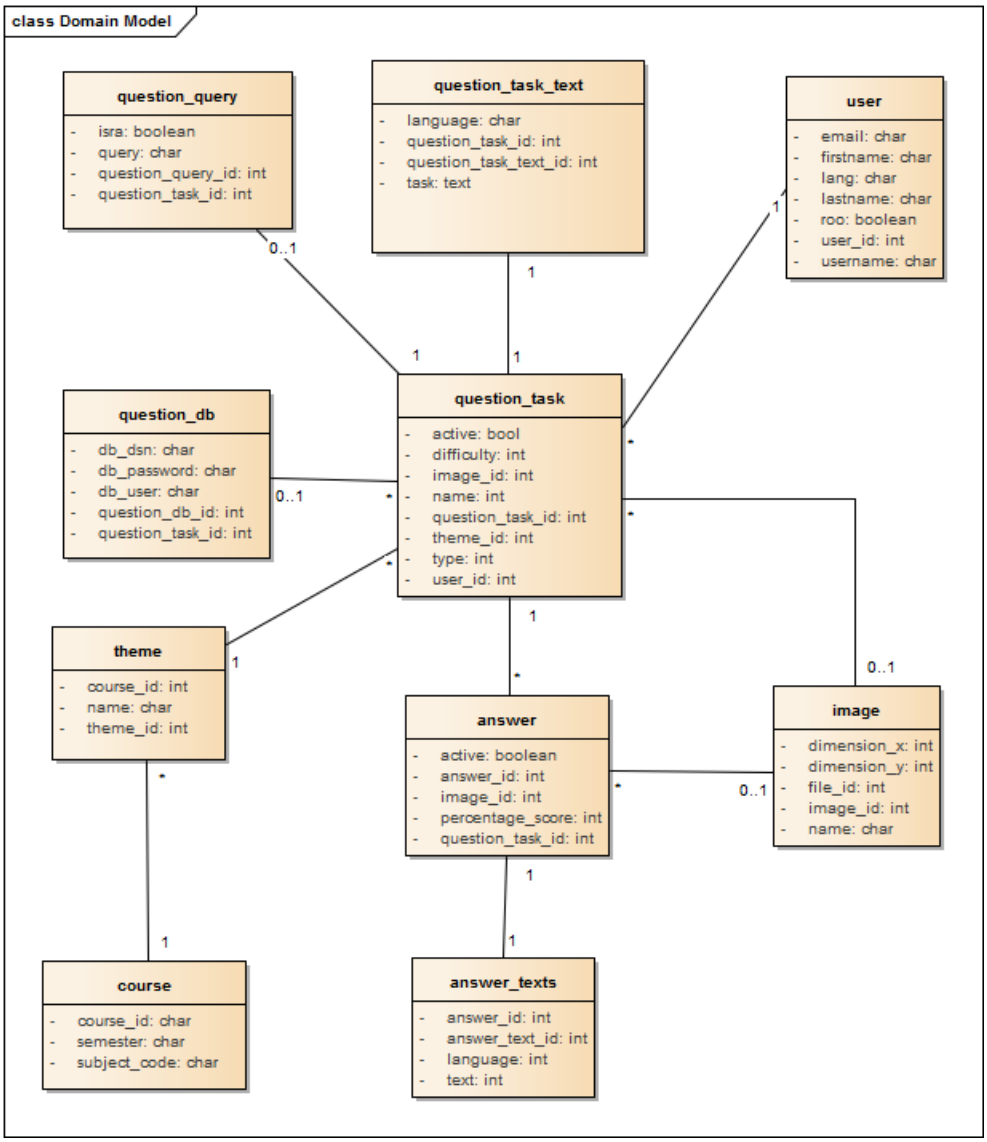
Obrázek 1.2: Tvorba otázky

rázek 1.2. Všechny volby formuláře jsou povinné a musí být vyplněny. Pokud nebudou všechny vyplněny, systém otázku nevytvoří a zahlásí chybu. Ve volbě kategorie máme na výběr z těchto možností:

- jazyk SQL
- relační algebra
- databázový model
- teoretická otázka

V typu odpovědi jsou na výběr následující možnosti:

- zaškrtávací s jednou správnou odpovědí
- zaškrtávací s více správnými odpověďmi
- textová odpověď
- kreslící nástroj
- odpověď v jazyce SQL



Obrázek 1.3: Schéma staré databáze (otázka a odpověď)

- odpověď v relační algebře

Jazyk otázky je buď anglický, nebo český. Obtížnost otázky lze vybrat z možností lehká, střední a těžká.

Na obrázku 1.3 je schéma části databáze, která se stará o ukládání otázek a souvisejících částí.

Při tvorbě otázky se uloží zadané hodnoty do databáze. Do tabulky *question_task* se uloží *název otázky*, *typ odpovědi* a *obtížnost*, do této tabulky se také uloží pravdivostní hodnota (boolean) *active*, která značí, jestli byla otázka

použita v testu (při tvorbě nové otázky je false). Pro *kategorii otázky* se vytvoří vazba na hodnotu k tabulce *theme*, stejně tak pro *user* hodnotu, která značí autora nově vytvořené otázky. Zadání a jazyk otázky se uloží do tabulky *question_task_text*. Případně se uloží obrázek do tabulky *image*, pokud byl při tvoření otázky zadán.

Model na uložení otázky je poměrně rozsáhlý, jelikož při tvorbě odpovědí se vytvoří vazby na tabulky *question_query* a *question_db*.

1.2 Tvorba odpovědi

V systému se tvorba odpovědi provádí společně s tvorbou otázky (viz obrázek 1.4). Popis tvorby odpovědi dále v textu rozděluji na kapitoly z důvodu přehlednosti.

Aby se otázka dala automaticky opravit, je potřeba zadat referenční správnou odpověď. Zde se nabídka dělí podle výběru z typu odpovědi u tvorby otázky. Pro různé typy se tvoří různé odpovědi.

1.2.1 Zaškrtávací s jednou správnou odpovědí

Zde musíme vytvořit více odpovědí, ze kterých následně budou studenti vybírat ty správné. V této otázce student odpověď nevyplňuje, ale vybírá odpověď z nabídky. Proto zde vyučující musí vyplnit více odpovědí a jednu označit jako správnou.

V aktuálním systému to začíná být nepřehledné. Celé vytvoření otázky, vytvoření, nebo odebrání odpovědí a následné vybírání správných odpovědí se provádí v jednom formuláři. Tvorba tohoto typu odpovědi je zachycena na obrázku 1.4.

1.2.2 Zaškrtávací s více správnými odpověďmi

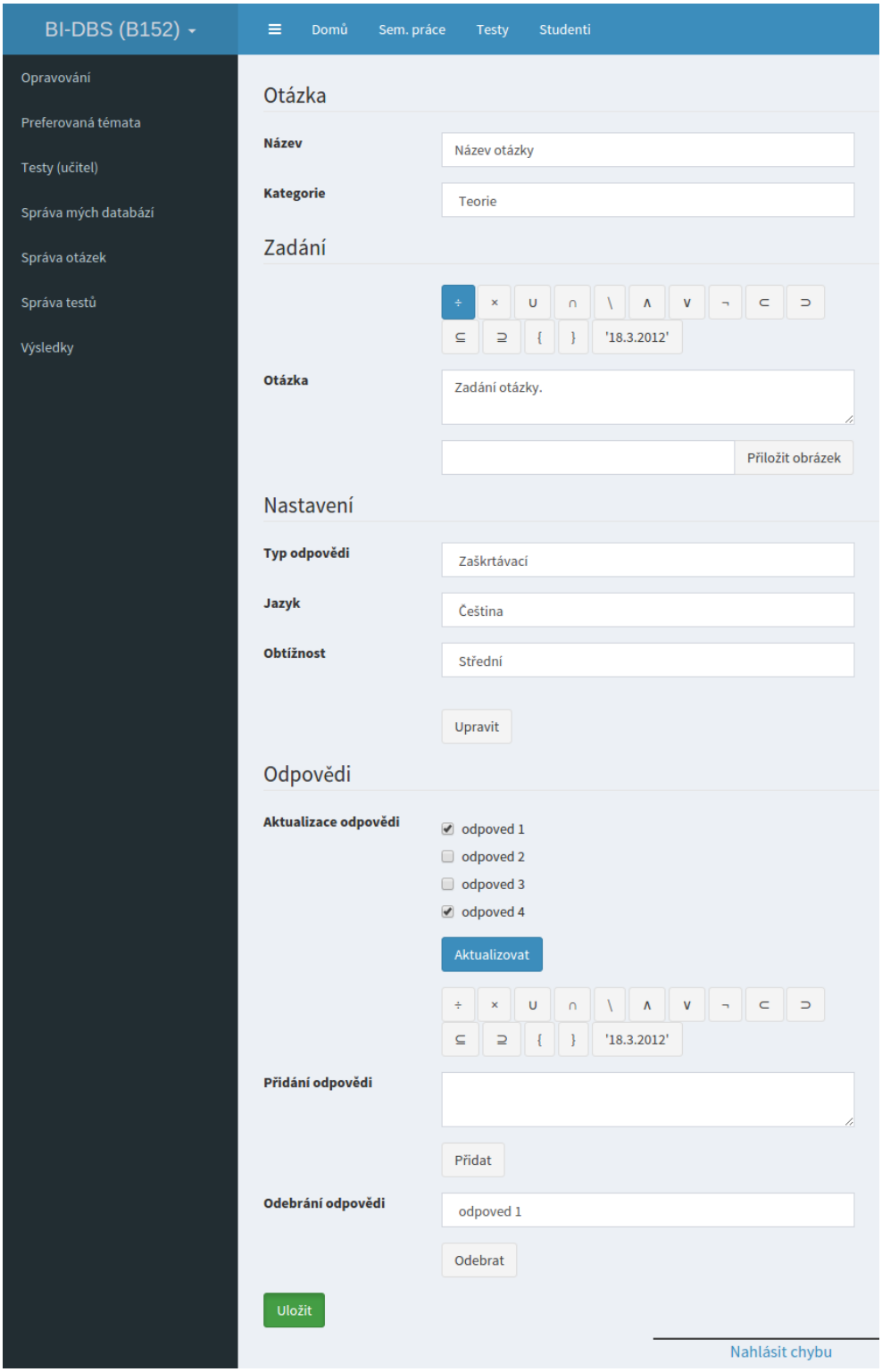
Tvorba této odpovědi je téměř totožná s předchozí, rozdíl je, že zde vyučující může označit více odpovědí jako správné.

1.2.3 Textová odpověď a kreslící nástroj

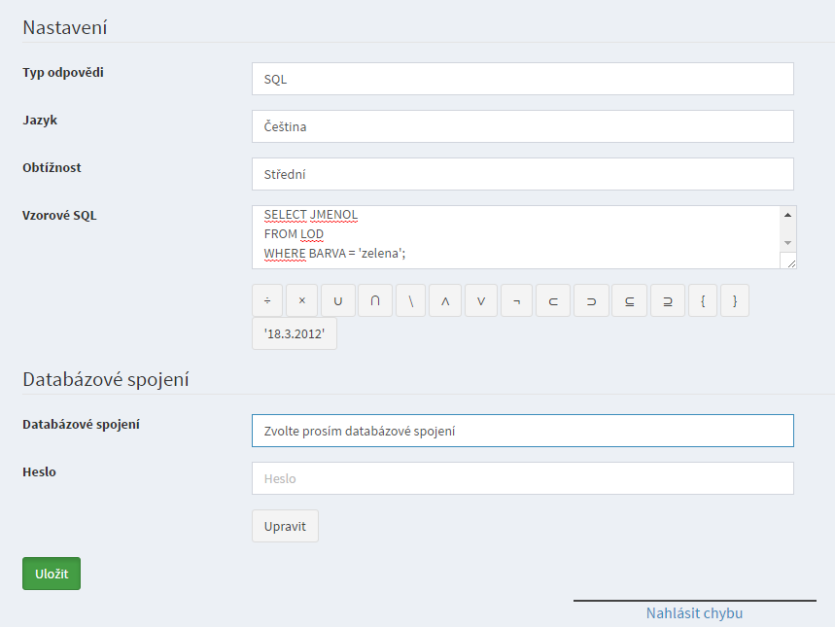
K otázkám s textovou odpovědí, nebo odpovědí ve formě kreslení diagramu se nevytváří referenční odpověď. Systém nepodporuje vytvoření referenční odpovědi k těmto typům otázek. Zejména u kreslícího nástroje je to velká škoda, tento nástroj totiž ukládá vytvořený model do formátu XML a zároveň poskytuje možnost porovnaní dvou modelů uložených v XML.

1.2.4 Odpověď v jazyce SQL

K otázce s odpovědí v jazyce SQL se musí zadat databázové připojení, které bylo dříve vytvořeno. Nad touto databází se bude provádět odpověď jak re-



Obrázek 1.4: Tvorba odpovědi



Obrázek 1.5: Tvorba odpovědi SQL

ferenční, tak studentova, porovnáním vysledků studentovy odpovědi oproti referenční můžeme určit, zda je správná. Formulář na tvorbu odpovědi v jazyce SQL je uveden na obrázku 1.5.

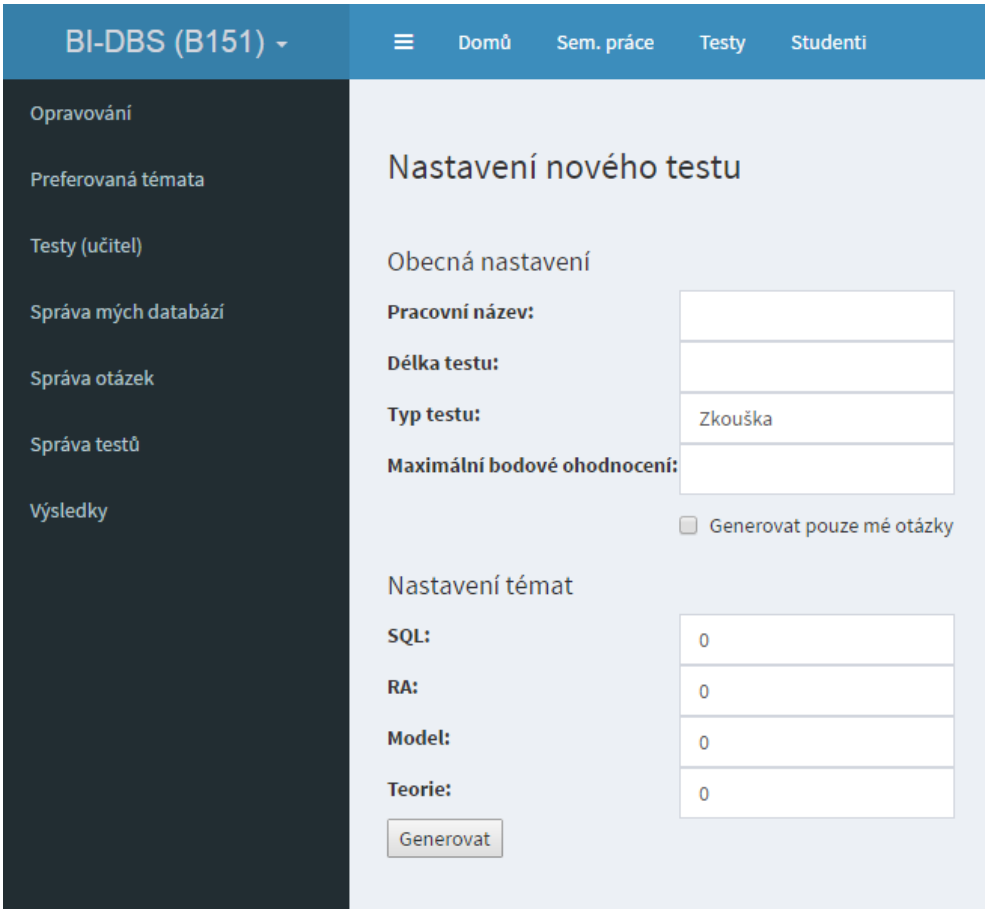
1.2.5 Odpověď v relační algebře

Odpověd v relační algebře je velice podobná odpovědi v jazyce SQL, také se musí k otázce připojit databázové připojení, navíc se ještě výraz v relační algebře převede pomocí RATu (aplikace vytvořená na FIT pro BI-DBS, překládá relační algebru do jazyka SQL) do jazyka SQL a následně se opraví jako SQL odpověď.

1.2.6 Ukládání odpovědí

Důležitým prvkem je také způsob, jak se ukládají data do databáze. Schéma vidíme na obrázku 1.3.

Po vytvoření odpovědi se nám vytvoří záznam v tabulce *answer*, kde se uloží do *percentage_score* správnost odpovědi, při tvorbě odpovědi to je vždy 100 %, pouze u odpovědí s vybíráním to může být 0 % (jiných hodnot může nabýt až při opravování otázek), dále odkaz na *question_task*, tedy otázku, ke které odpověď patří, odkaz na *image*, obrázek (tato vazba se nikdy nevytvoří) a boolean *active* (opět se nevyužívá).



Obrázek 1.6: Tvorba šablony

Do tabulky *answer_texts* se uloží zadaná referenční odpověď a jazyk odpovědi (přebírá se z jazyku otázky).

Když je vytvořena otázka typu SQL/RA, budou využity další dvě tabulky a to *question_query* a *question_db*. Do tabulky *question_query* se zduplikuje odpověď, kterou již máme uloženou v *answer_texts* (tato tabulka řeší implementační problém a je tedy samozřejmě redundatní). V tabulce *question_db* je uchovávána informace o databázovém připojení k dané otázce.

1.3 Tvorba šablony testu

Aby bylo možné zadat stejný test více lidem, je potřeba vytvořit šablonu obsahující informace o testu, jako je třeba délka testu, nebo jednotlivé otázky. Vytvoření šablony je na obrázku 1.6.

Ve volbě Typ testu je na výběr z těchto možností:

- Demo test
- Test v semestru
- Zkouškový test

Takto vyplněná šablona není validní a nelze z ní vytvořit test. K šabloně ještě musíme přiřadit otázky, které budou obsaženy v testu.

1.4 Přiřazení otázek k šabloně testu

1.4.1 Generování otázek

Systém nabízí možnost generování náhodných otázek k testu na základě zadaných údajů v dalším formuláři. Do formuláře zadáme počet otázek, které chceme vygenerovat z dané kategorie. Generování těchto otázek je náhodné, rozdělení bodů je pokud možno rovnoměrné. Volba generování je uvedena na obrázku 1.6.

Systém nenabízí možnost manuálního sestavení testu. Tato možnost se hodí zejména v případě, kdy vyučující vytvoří novou sadu otázek a z té potřebuje sestavit test. Pokud toto vyučující chce nyní, musí si vygenerovat test a pak ručně nahradit vygenerované otázky těmi, které chce mít v testu. To je nepříjemné při představě, že si vyučující v první řadě vůbec nemusel nic generovat a rovnou mohl sestavit test podle sebe.

1.4.2 Sestavení testu

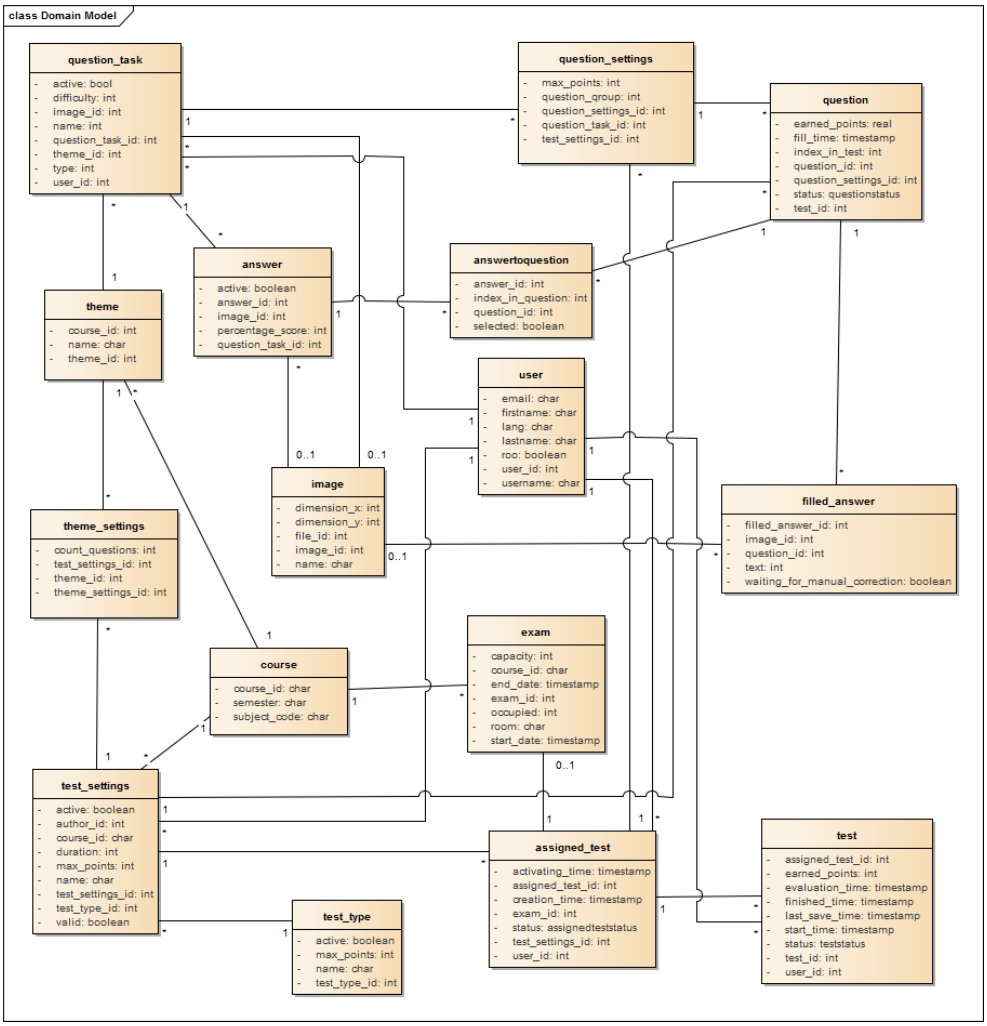
V seznamu vygenerovaných otázek vyučující otázky zkontroluje, případně může otázky nahradit jinými, nebo změnit bodové ohodnocení otázky, které bylo vygenerované automaticky, ale pro danou otázku je nevyhovující.

Tyto změny otázek jsou uživatelsky velice nepříjemné a složité. Otázky v seznamu nejsou řazeny a po změně, nebo odebrání se mohou zamíchat. To může být matoucí a může se zdát, že se otázka nenahradila nebo se nahradila jinou otázkou, než bylo zadáno.

1.4.3 Uložení šablon

Na obrázku 1.7 je znázorněné další schéma databáze, zabývající se tvorbou, sestavením a spuštěním testu (schéma opět opomíjí části, které zde nejsou potřebné). Při vytvoření testu se uloží záznam do *test_settings* s povinnou vazbou na *test_type*.

Při generování otázek k testu se uloží jednotlivá otázka do tabulky *question_settings*, která obsahuje informace o počtu bodů k přiřazené otázce, odkaz na přiřazenou otázku a odkaz na test, ke kterému otázka patří. Tímto způsobem se přiřadí otázka k testu s informací o maximálním počtu přidělitelných bodů za otázku.



Obrázek 1.7: Schéma staré databáze (test)

1.5 Vytvoření testu

Z vytvořené šablony vytvoří vyučující test. Nemusí se tedy už starat o přidání a ohodnocení otázek, to už bylo provedeno při tvorbě šablony. Vytvořený test ještě nelze spustit, musí se k němu přiřadit studenti, kteří budou psát tento test.

Do databáze se uloží záznam o vytvořeném testu do tabulky *assigned_test* s odkazy na šablonu testu (*question_settings*) a uživatele, který vytvořil tento test (*user*).

1.5.1 Přiřazení studentů

K vytvořenému testu přiřadíme studenty. V tabulce máme seznam všech studentů přihlášených do kurzu, vybereme ty, kteří budou psát test. Můžeme k testu přiřadit přímo celou paralelku (cvičení, proseminář nebo přednáška), tyto paralelky se stahují z KOSu. Pokud chce psát test student z jiné paralelky, musí ho vyučující přidat zvlášť.

Pro každého studenta přiřazeného do testu se vytvoří záznam v tabulce *test*, k záznamu je odkaz na vytvořený test (*assigned_test*). Viz obrázek 1.7.

1.6 Spuštění testu

Vyučující může test spustit jak jednotlivě, tak všem studentům najednou. Po spuštění lze jednotlivým studentům test zastavit a ukončit, to samé lze udělat všem studentům najednou.

1.7 Vypracování testu

Nyní, když je test spuštěný, se studenti mohou pokusit o jeho správné vyřešení. Čas testu se spustí, když do něj student vstoupí. Student se nyní může pokusit vypracovat test. Z tvorby otázky víme, že na otázku se dá odpovídat různými způsoby.

Odpovědi jsou typu:

1. zaškrtávací s jednou správnou odpovědí
2. zaškrtávací s více správnými odpověďmi
3. textová odpověď
4. kreslící nástroj
5. odpověď v jazyce SQL
6. odpověď v relační algebře (RA)

U všech typů má student možnost svou odpověď uložit, pouze u odpovědi v jazyce SQL a RA si může navíc student zkontrolovat, jestli je jeho odpověď syntakticky správně, takzvaně zvaliduje odpověď. To mu sice neřekne, jestli je jeho odpověď správná, ale přinejmenším ví, že se jeho SQL/RA úspěšně provede.

1.8 Uložení testu

Pokusím se vysvětlit, jak systém ukládá studentovi odpovědi do databáze. Na obrázku 1.7 je schéma. Když student odpovídá na otázku typu *zaškrtávací s více správnými odpověďmi*, nebo *zaškrtávací s jednou správnou odpovědí*, uloží se odpovědi do tabulky *answertoquestion* a pro každou možnou odpověď se tam vytvoří záznam s parametrem, který značí, zda student tuto odpověď označil jako správnou. Pokud je odpověď jiného typu, vytvoří se záznam v tabulce *filled_answer*.

1.9 Odevzdání a automatické ohodnocení testu

Když má student již vyplněné všechny otázky, nebo už nechce ve vyplňování dále pokračovat, ukončí test. Po ukončení testu proběhne nad testem automatická oprava. Automatická oprava opraví otázky, které lze automaticky vyhodnotit a ty, které nelze, pošle vyučujícímu do manuální opravy.

Pokud student neukončí svůj test do časového limitu nastaveného při tvorbě šablony testu, tak systém ukončí test za něj a automaticky opraví otázky, které lze vyhodnotit, ostatní pošle vyučujícímu do manuální opravy.

Pro každý typ otázky se provede jiná automatická oprava.

1.9.1 Zaškrtávací s jednou správnou odpovědí

Automatická oprava zpravidla opraví otázky, kde se vybírá odpověď z nabídky, otázky tohoto typu opraví vždy, za správné odpovědi přidělí 100 % bodů, za nesprávné 0 %.

1.9.2 Zaškrtávací s více správnými odpověďmi

Stejně jako u typu s jednou správnou odpovědí se otázky tohoto typu opraví vždy, za správné odpovědi se přidělí 100 % bodů, za nesprávné 0 % (za nesprávné je považován jakýkoliv výběr, který není správný).

1.9.3 Textová odpověď

Otázky vyžadující textovou odpověď se opravují velice špatně, šance, že student odpověděl na otázku stejnou větou, či slovem jako vyučující v referenční odpovědi je poměrně malá. Nicméně se provede porovnání studentovy odpovědi na shodu s referenční odpovědí. Když se odpovědi shodují, ohodnotí se otázka 100 % bodů. Jinak se vyčká na manuální opravu vyučujícím.

1.9.4 Kreslící nástroj

Otázky tohoto typu se automaticky neopravují.

1.9.5 Odpověď v jazyce SQL

Otázky na dotazovací jazyk SQL se opraví pouze tehdy, když se odpověd shoduje se vzorovou odpovědí, nebo je shodný výstup dotazu z databáze. Dotazy se vykonávají nad databází, která byla přiřazena k této otázce. Tyto otázky se ohodnotí 100% body, ostatní čekají na manuální opravu.

1.9.6 Odpověď v relační algebře

Otázky na relační algebru (RA) jsou téměř totožné s otázkami v jazyce SQL pouze s tím rozdílem, že otázky na RA se nejprve pomocí RATu převedou na výraz v jazyce SQL. Následně je proces opravení shodný s otázkou v jazyce SQL.

1.10 Manuální oprava

Jestliže se neopravily automaticky všechny otázky, musí vyučující zbylé opravit ručně. Systém nabízí možnost, jak manuální opravu urychlit a zjednodušit (bohužel je to zdroj velkého množství chyb s systému). Pro každou nezodpovězenou otázku, seřadí odpovědi od studentů a duplikující se odpovědi nabídne k opravě pouze jednou, čímž vyučujícím ušetří čas a navíc je tato oprava spravedlivá pro všechny studenty, kteří otázku zodpověděli stejně.

Samotná oprava je jednoduchá, systém zobrazí zadání otázky a odpověď, která byla označena jako správná, buď při tvorbě otázky, nebo při manuální opravě (to je poměrně matoucí, měla by se zobrazit odpověď, která byla vytvořena společně s otázkou). Dále zobrazí studentovu odpověď.

Vyučující danou odpověď zkontroluje a ohodnotí jí dle správnosti, přičemž nemůže překročit maximální bodové ohodnocení, které bylo nastaveno při tvorbě testu. Ohodnotí-li vyučující otázku jako správnou a udělí za ní plný počet bodů, přidá se taková odpověď jako referenční. To znamená, že až se příště tato otázka objeví v testu, množina jejích referenčních odpovědí bude větší. Tudíž pro příště se vyučujícímu do manuální opravy dostane méně otázek.

Kapitola 2

Návrh nové aplikace pro podporu výuky BI-DBS

Z analýzy starého systému [1] jsme zjistili, že způsob ukládání dat je nesprávně navrhnut, zbytečně rozsáhlý a obsahuje duplikující informace. Postupujme jako při průchodu systémem a navrhujme nové uspořádání aplikace na základě definovaných požadavků. V průběhu si všimneme chyb v návrhu starého systému.

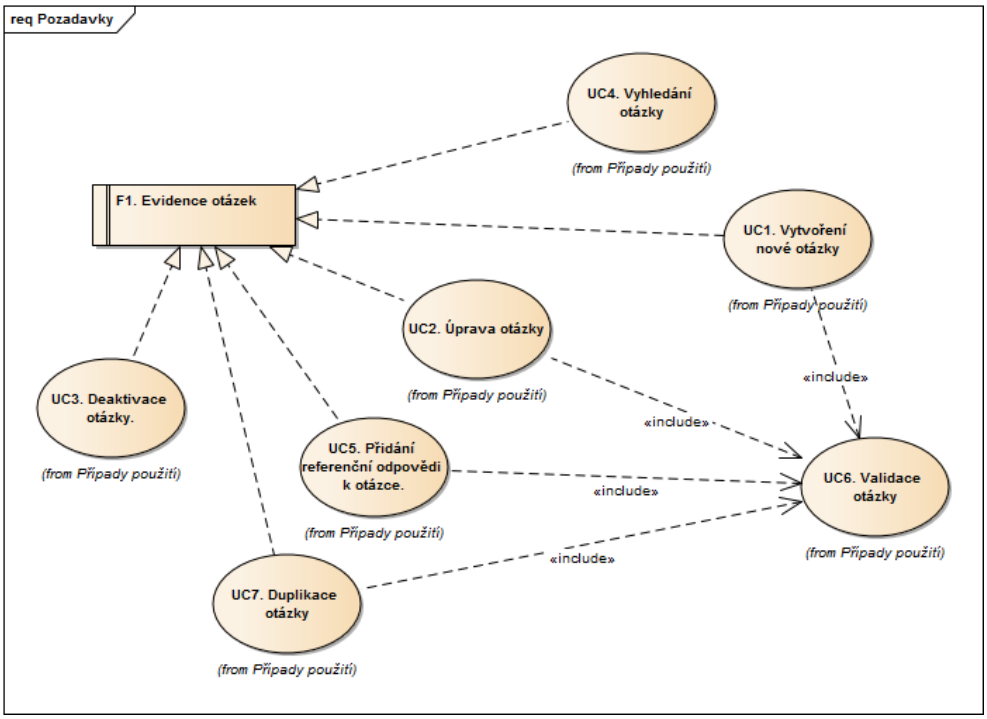
2.1 Otázka

Při bližším prozkoumání zjistíme, že pro uložení otázky je klíčové uložení zadání otázky, obrázku k otázce, obtížnosti, kategorie, jazyku a typu otázky. Všechny tyto údaje jsou esenciální pro jakoukoliv práci s otázkou, není proto důvod ukládat tyto informace separátně, jak je tomu ve starém systému.

Pouze pro kategorii, obtížnost, jazyk a typ otázky by mělo význam vytvořit tabulku všech možných kategorií, obtížností, jazyků a typů k otázce, pro jejich jednoduchou správu. Jelikož není očekávané, že by bylo potřeba přidávat či odebírat typy, není proto nutné tvořit samostatné tabulky. Samozřejmě s rozšiřitelností musíme počítat, to ale můžeme vyřešit implementačně (vytvoříme si tabulku staticky v kódu).

Práce s otázkami je definovaná *use case diagramem* - viz obrázek 2.1. Tyto požadavky se vztahují pro práci s otázkami z pohledu vyučujících. Z diagramu vidíme, že když se pokusíme jakkoliv upravit otázku, vyžaduje to validaci dané otázky.

Validace otázky je prověření otázky, zda splňuje všechny podmínky, aby mohla být přiřazena k testu. Jinými slovy testujeme otázku, zda má korektně vyplněné zadání a jsou k ní přiřazeny referenční odpovědi. Přidání referenční odpovědi k otázce je klíčová věc. Bez referenční odpovědi nemůže být otázka validní.



Obrázek 2.1: Požadavky na otázku

2.2 Odpověď

Odpověď je závislá na typu otázky. Každý typ otázky vyžaduje jiný druh odpovědi či dokonce více odpovědí. Nicméně pokusíme se to co nejvíce sjednotit pro jednoduchost a přehlednost. Najdeme společné části u různých druhů odpovědí a ty budeme vytvářet jednotně.

Zaměřme se na tvorbu odpovědi a vidíme, že každá odpověď vyžaduje nějaký text (pouze kreslící nástroj je výjimkou, ten budeme muset řešit zvlášť, ať chceme, nebo ne). Stačí nám pro ostatní tedy pouze vzorová odpověď? Ne, pro odpověď v jazyce SQL a relační algebře potřebujeme ještě zadat databázové spojení k databázi, nad kterou se nám budou vykonávat dotazy.

2.2.1 Textová odpověď

Nejjednodušším typem odpovědi je textová odpověď, je to pouze text, který dále nepotřebuje další akce. Bude to tedy naše standardní odpověď, ostatní odpovědi budou pouze rozšířením této. Nyní si projděme jednotlivé typy odpovědí a zaměřme se na rozdíly od textové odpovědi, lépe si tak všimneme rozdílů mezi odpověďmi.

2.2.2 Zaškrtávací s jednou správnou odpovědí

Tento typ odpovědi se na první pohled příliš nelíší od textové, jediný rozdíl zde je v tom, že vytvoříme více textových odpovědí. Z těchto odpovědí vybereme jednu správnou a tu označíme jako správnou. Žádné další akce nejsou nutné.

2.2.3 Zaškrtávací s více správnými odpověďmi

Tvorba této odpovědi je téměř totožná s předchozí, rozdíl je v tom, že se zde může označit více odpovědí jako správné.

2.2.4 Kreslící nástroj

Kreslící nástroj je kapitolou samou o sobě. Proto zde místo nějakého druhu textového pole budeme mít možnost nakreslit diagram, stejným nástrojem jakým následně budou kreslit i studenti v testu. Tento nástroj byl vytvořen Jiřím Slavotínkem přímo pro naše účely. Diagram vytvořený tímto nástrojem převede nakreslený diagram do formátu XML (můžeme ho uložit jako text). Což jak uvidíme dále, je pro naše účely ideální. Nakonec s ním tedy můžeme pracovat jako s textem.

2.2.5 Odpověď v jazyce SQL

K otázce s odpovědí v jazyce SQL je potřeba kromě textového pole zadat také databázové připojení. Nad touto databází se bude provádět odpověď a je pro nás klíčové, aby nám odpověď (tedy SQL dotaz v tomto typu odpovědi) vracela nějaké relevantní výsledky z databáze. Tyto výsledky budou následně sloužit pro ověření správnosti odpovědí studentů v testu.

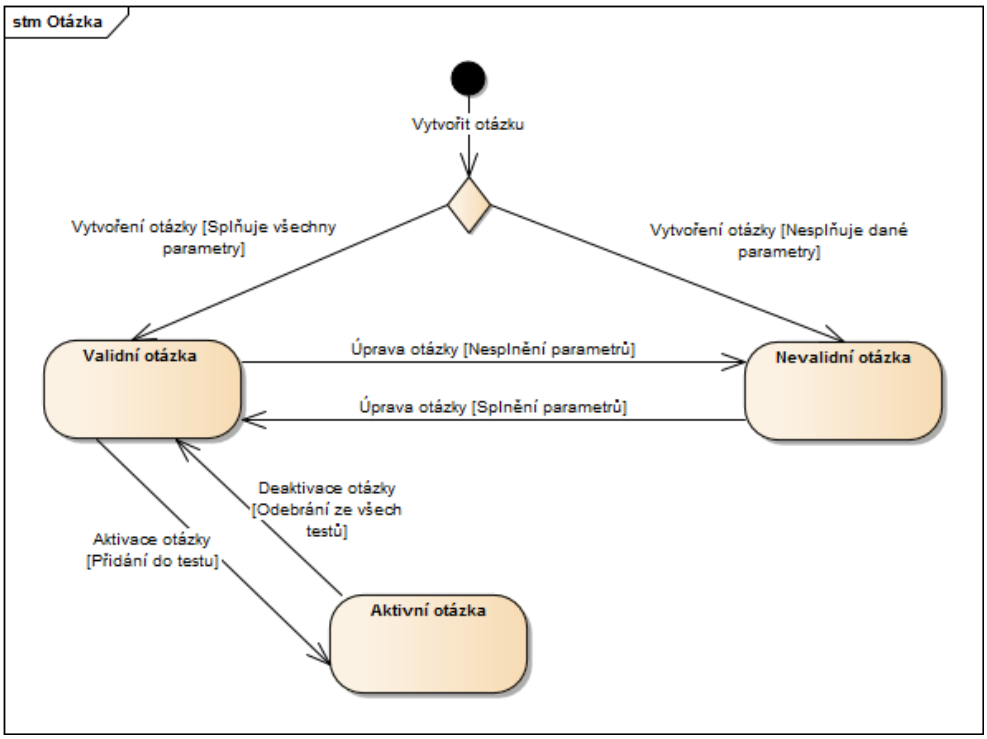
2.2.6 Odpověď v relační algebře

Odpověd v relační algebře je velice podobná odpovědi v jazyce SQL, také se musí k otázce připojit databázové připojení, navíc se ještě výraz v relační algebře převede pomocí RATu do jazyka SQL a následně s ní pracuji jako s SQL odpovědí.

2.3 Validace otázky

Abychom byli schopni s otázkami relevantně pracovat, potřebujeme si vytvořit řád, jak s nimi pracovat. Potřebujeme odlišit otázky, které jsou korektně vyplněné a můžeme je tedy použít v testu. Znamená to tedy, že otázka, které třeba chybí referenční odpověď, nebo dotaz, nevrací žádné výsledky (SQL a RA). Nebudou použity v testu.

Po vytvoření otázky, nebo po její úpravě musíme tuto otázku zvalidovat. Není-li tomu tak, označíme otázku jako nevalidní. Pro zvalidování nevalidní



Obrázek 2.2: Stavy otázky

otázky je potřeba vyplnit všechny atributy otázky tak, aby byla validní. V kapitole 3.2.2, kde se věnuji implementaci, ukáži kód validační funkce a vysvětlím, jak vyplnit otázku, aby byla validní - viz obrázek 2.2.

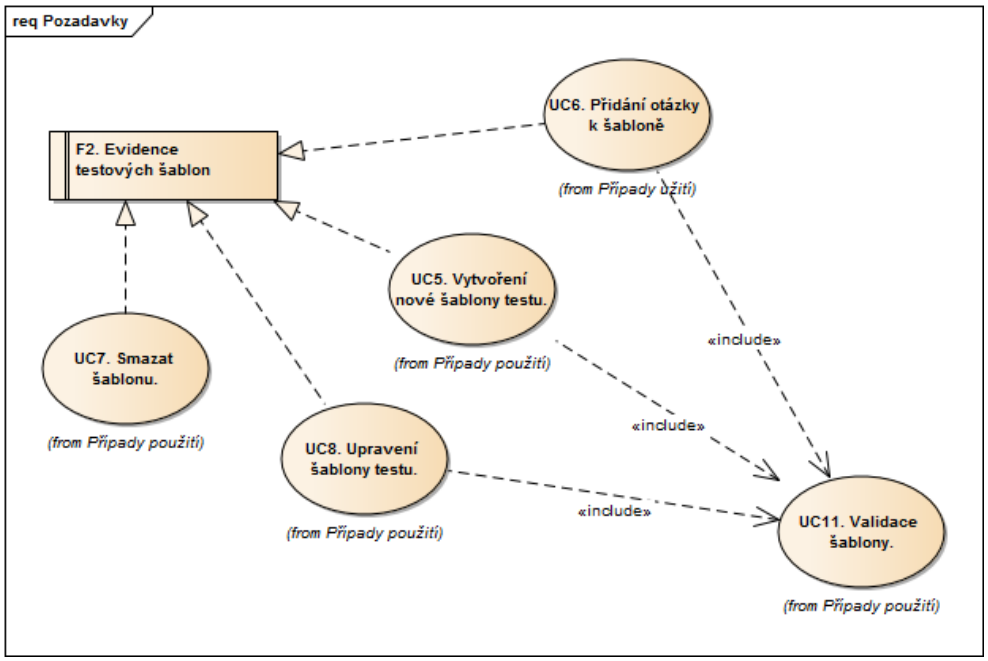
Nevalidní otázka nemůže být použita v testu. Validní smí být použita v testu, jakmile jí použiji v testu bude z ní aktivní otázka. Aktivní otázka je už použita v testu a smí být tedy použita znovu.

2.4 Šablona testu

Až si vytvoříme dostatek otázek, ze kterých si budeme chtít sestavit test, vytvoříme šablonu testu, která nám bude tyto otázky sdružovat.

Myšlenka šablony je velice jednoduchá, vytvoříme šablonu a z ní dále budeme kopírovat její informace, kolikrát budeme chtít. My tvoříme šablonu pro test, proto k ní musí být přiřazeny otázky, na které budou studenti odpovídat. Pro otázky potřebujeme dále znát maximální bodové ohodnocení. Chceme dále vědět, jak dlouho bude daný test trvat, aby mohl být zastaven po vypršení času.

Požadavky na šablonu máme definovány grafem - viz obrázek 2.3.



Obrázek 2.3: Požadavky na šablonu

2.4.1 Přiřazení otázek

Pro přiřazení otázek platí pravidlo, že jedna otázka nesmí být ve stejné šabloně víckrát, za jednu otázku nemůže být přiděleno více bodů, než je uvedeno v šabloně, do šablony můžeme vkládat pouze validní otázky. Jinak už je to pouze na nás, jaké otázky do testu dáme a kolik jim přidělíme bodů.

2.4.2 Aktivace otázek

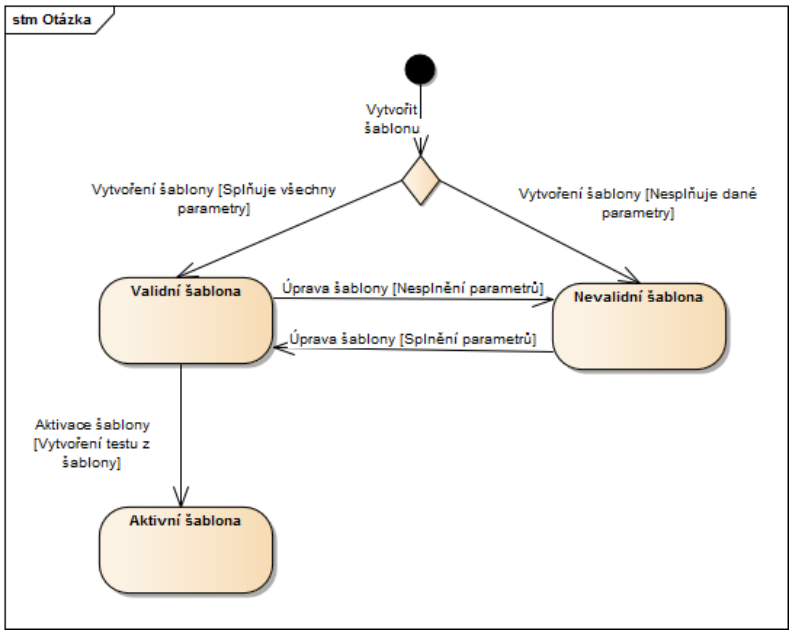
Jakmile otázku přiřadíme k šabloně, už nechceme, aby nám otázku někdo náhodou nesmazal, nebo neupravil. Převedeme si otázku do dalšího stavu - viz obrázek 2.2.

2.4.3 Deaktivace otázek

Otázka nemůže být deaktivována ručně. Jediný způsob deaktivace otázky je její odstranění ze všech šablon, kde byla použita. Dále uvidíme, že otázka nepůjde odstranit z šablony, pokud už byl ze šablony vytvořen test.

2.4.4 Validace šablony

Šablona musí být zvalidována před uložením. Validací docílíme toho, že si ošetříme, zda je za jednotlivé otázky udělen správný počet bodů, kontrolujeme,



Obrázek 2.4: Stavy šablony

zda jsou otázky v šabloně validní. Pokud tomu tak není, označíme šablonu jako nevalidní, což bude mít za následek, že z ní nepůjde vytvořit test pro studenty. Rozdíl mezi validní a nevalidní šablonou je tedy v tom, že z validní mohu spustit test, z nevalidní nikoliv - viz obrázek 2.4.

2.5 Test

Test vytvoříme ze šablony. Přiřadíme k němu studenty a test sputíme. Když je test spuštěný, studenti ho mohou řešit, proto budeme muset test uzavřít po uplynutí času vyhrazeného na tento test.

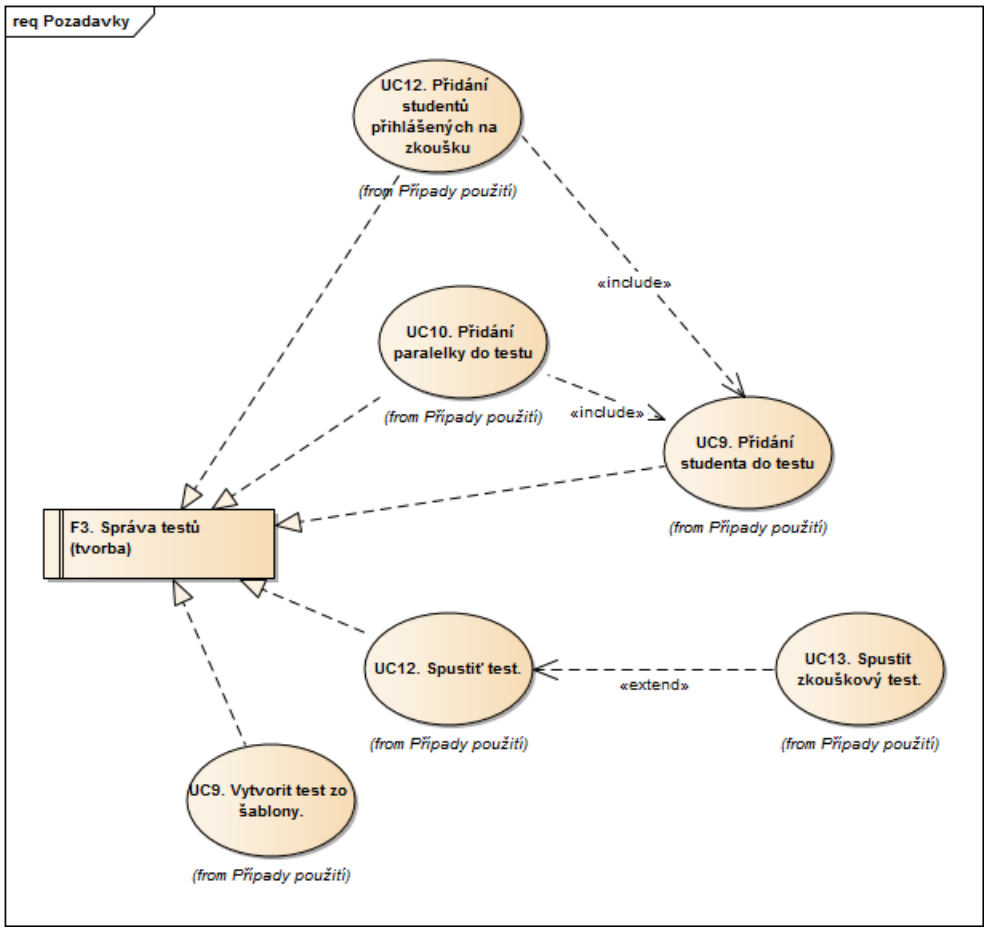
Naše požadavky jsou definovány grafem viz. obrázek 2.5. Stavy testu jsou zobrazeny v grafu 2.6.

Jakmile vytvoříme test ze šablony, už nebudeme chtít, aby šla šablona upravit, nebo smazat. Zároveň tak otázky, které jsou v testu, nechceme, aby šly upravit, nebo smazat. Toho docílíme aktivací šablony a otázek (ty už se aktivovaly přidáním do šablony).

2.5.1 Aktivace šablony

Šablona se aktivuje vytvořením testu z dané šablony.

Aktivovaná šablona tedy už nepůjde upravit či smazat, to je žádoucí, protože kdybychom smazali šablonu, ze které je vytvořen test, vznikla by nekonzistence dat. Zároveň nám z toho plyne, že otázky přiřazené do této šablony,



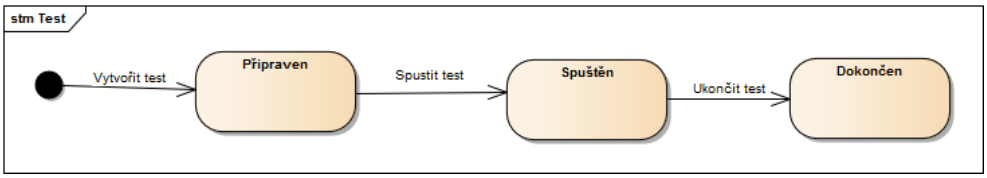
Obrázek 2.5: Požadavky na test

už nepůjdou upravit či smazat. Toho už jsme docílili pravidlem pro deaktivaci otázek.

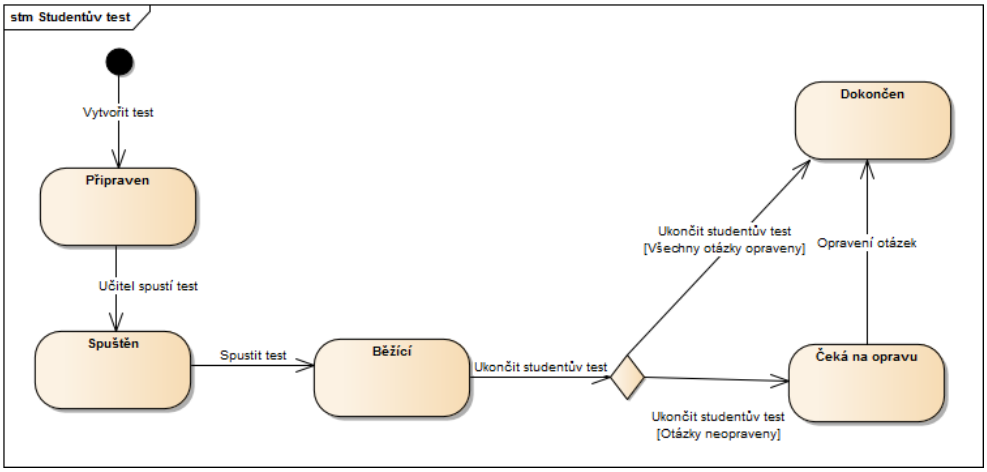
2.5.2 Přiřazení studentů do šablony

Když je test ve stavu *připraven*, můžeme do něj přiřazovat studenty. Studenty budeme chtít přiřadit po celých paralelkách (cvičení, přednáška), nebo po všech studentech přihlášených na zkoušku. Informace o paralelkách a studentech přihlášených na zkoušku budeme stahovat z KOSu.

Pro přidávání studentů do testu platí jediné pravidlo. V testu může být student přiřazen pouze jednou. To znamená, že se nesmí stát, aby byl student v testu přiřazen víckrát, to by mělo za následek, že student by mohl test vyplnit víckrát.



Obrázek 2.6: Stavy testu



Obrázek 2.7: Stavy studentova testu

2.5.3 Spuštění testu

Spuštěním testu umožníme studentům test vypracovat. Po spuštění testu už nebude možné přidat či odebrat studenta. Kvůli integritě dat nebude ani možné smazat tento spuštěný test, test se pouze převede do stavu *Dokončen*.

2.5.4 Uzavření testu

Test uzavřeme po vypršení přiděleného času, uzavřením testu znepřístupníme studentovi, který do té doby nespustil svůj test (stav *Spuštěn* viz obrázek 2.7), vstup do testu. Studentův test ve stavu *Běžící* uzavírat nebudeme, protože jak uvidíme dále, student si spustí svůj čas při vstupu do testu.

2.6 Studentův test

Při přiřazení studenta do testu se nám vytvoří pro studenta jeho test s otázkami, které byly přiřazeny do šablony, ze které byl vytvořen test. Studentův test se nám vytvoří ve stavu *Připraven*, po spuštění testu se stav změní na *Spuštěn* - viz obrázek 2.7.

2.6.1 Spuštění studentova testu

Student je oprávněn ke vstupu do testu pouze tehdy, pokud je test ve stavu *Spuštěn*. Ve stavu *Připraven* student pouze vidí, že byl přiřazen do testu, ale nemůže do testu vstoupit. Když student vstoupí do testu, změní se stav na *Běžící* a se vytvoří *timestamp* (zaznamená se aktuální čas), od této *timestamp* se bude měřit studentovi čas na vypracování testu.

2.6.2 Ukončení studentova testu

Pokud student vyplní všechny otázky, nebo jen chce test odevzdat, ukončí test. Ukončením testu nad testem proběhne automatická oprava, která opraví otázky, které jdou automaticky opravit. Ostatní pošle k manuální opravě. Pokud student neukončí test v časovém limitu, bude test ukončen automaticky.

2.7 Automatická oprava

Stavový diagram studentovy odpovědi vidíme na obrázku 2.8.

Automatická oprava testu zkontroluje postupně každou otázku z testu. Pokud není otázka vyplněná, ohodnotí studentovu odpověď 0 body a označí ji jako *Automaticky opravená*, jinak následují další testy.

Otázku zkusí porovnat s referenční odpovědí zadanou při tvorbě otázky, pokud se odpovědi shodují, ohodnotí jí maximem bodů a označí ji jako *Automaticky opravená*.

Dále otázku zkusí porovnat s jakoukoliv odpovědí uloženou při opravě jiného testu (více dále), pokud se odpovědi shodují, ohodnotí jí počtem bodů přidělených k otázce a označí ji jako *Automaticky opravená*.

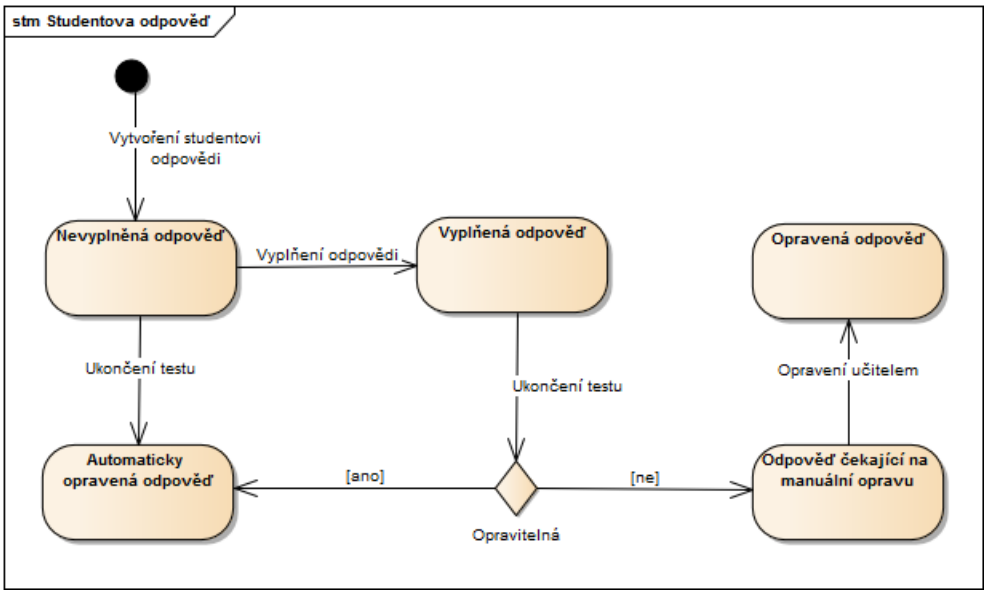
To je vše, co se dá zkontrolovat u typů otázky *zaškrtávací s jednou správnou odpovědí*, *zaškrtávací s více správnými odpověďmi* a *textová odpověď*, takže pokud se dané otázky automaticky neopravily, budou označeny jako *Čekající na manuální opravu*.

2.7.1 Odpověď v jazyce SQL

U otázek typu *SQL* provedeme studentův dotaz nad databází, která byla zadána ve tvorbě otázky. Pokud je výstup z databáze u studentovy odpovědi totožný s výstupem z databáze referenčních odpovědí, ohodnotí jí maximem bodů a označí ji jako *Automaticky opravená*. Je-li tomu jinak, otázka se označí jako *Čekající na manuální opravu*.

2.7.2 Odpověď v relační algebře

Otázka typu *RA* je nejprve pomocí RATu převedena na SQL dotaz, který následně vykonáme nad databází. Postup je potom stejný jako u otázky typu *SQL*.



Obrázek 2.8: Stavy studentovy odpovědi

2.7.3 Kreslící nástroj

Funkci na porovnání dvou diagramů (studentova a referenčního) vytvořil Jiří Slavotínek společně s nástrojem na kreslení diagramů. Tato funkce dokáže rozpoznat, zda jsou dva diagramy totožné. Pokud tomu tak je, bude otázka ohodnocena plným počtem bodů a bude označena jako *Automaticky opravená*, jinak jako *Čekající na manuální opravu*.

2.8 Manuální oprava

Odpovědi studentů, které nevyhodnotil automat, musí ručně vyhodnotit vyučující.

Při manuální opravě otázky se bude ukládat každá jedinečná studentova odpověď. K této odpovědi se uloží počet bodů, které za ně přidělil vyučující v manuální opravě. Najdou se stejné odpovědi a ohodnotí se stejně.

Uložení bodů přidělených za odpověď bude sloužit v budoucnu jako vzorová odpověď. Znamená to tedy, že pokud někdo odpoví na tuto otázku stejně, ohodnotí se jeho odpověď automaticky z dat, která se uložila při manuální opravě.

2.8.1 Správná odpověď

Pokud vyučující v manuální opravě ohodnotí otázku maximem bodů, bude tato odpověď sloužit jako referenční pro všechny budoucí testy.

2.8.2 Nesprávná odpověď

Pokud ale vyučující studentovu odpověď neohodnotí maximem bodů, bude tato odpověď sloužit jako referenční pouze pro testy ze šablony, ve které je přiřazena tato otázka a zároveň bude sloužit jen vyučujícímu, který tuto odpověď takto ohodnotil.

Znamená to tedy, že dvě stejné nesprávné odpovědi mohou dva vyučující ohodnotit rozdílně. Každý vyučující dává důraz na jiné věci, proto si musí své studenty hodnotit sám.

Kapitola 3

Implementace nové aplikace pro podporu výuky BI-DBS

Před samotnou implementací jsme se museli rozhodnout, jaké technologie budeme používat. Na novém systému jsem nepracoval sám, já jsem pracoval pouze na části věnující se testování. Část věnující se semestrálním pracem vypracoval Filip Glazar. Na společných částech systému má velkou zásluhu právě Filip, protože už má s podobnými projekty zkušenosti, a zhostil se vytvoření základní struktury aplikace.

Společně jsme zvolili technologie, které oba využíváme v systému. Toto jednotné používání stejných technologií má velký význam pro jednotný vzhled a chování celé aplikace.

3.1 Zvolené technologie

Technologie jsme volili na základě naší zkušenosti práce s nimi a jejich použitelností pro naše potřeby. Dalším kritériem byl výběr pokud možno stejných technologií, jaké byly použity na starém systému, z důvodu zachování stejného vzhledu a možnosti použít části starého systému bez nutnosti velkých zásahů (bohužel se nám podařilo ze starého systému použít jen komponentu na přihlašování pomocí Shibboleth).

3.1.1 PHP

Programovací jazyk PHP [2] byl jasnou volbou pro základní stavbu aplikace. S PHP jsme měli oba s Filipem dostatečné zkušenosti.

3.1.2 Nette

Samotný jazyk PHP nám nestačil, proto jsme zvolili český framework Nette [3], tento framework byl použit i na starém systému. Jeho výběr byl tedy nasnadě.

Jako další varinta byl výběr frameworku Symfony [4], tento framework jsme, ale nepoužili, protože jsme s ním neměli takové zkušenosti jako s Nette a oba nabízejí skoro stejnou podporu. Nette jsme tedy vybrali především proto, že se jedná o český framework s výbornou dokumentací, jednoduchým používáním a byl také použit ve staré aplikaci.

3.1.3 Nette/Database

Nette/Database je nadstavbou nad PHP PDO.

„*Třída Nette\Database\Connection tvoří obálku nad PDO a reprezentuje připojení k databázi. Základní funkcionalitu poskytuje Nette\Database\Context. Nette\Database\Table poskytuje pokročilou vrstvu pro práci s tabulkami.*“[5]

Nette/Database poskytuje vrstvu pro pokročilou práci s tabulkami, umožňuje snadno skládat dotazy, snadno získávat data a pokládat efektivní dotazy.

3.1.4 Nette/Forms

Nette/Forms výrazně usnadňuje vytváření a zpracování formulářů.

„*Nette Framework klade velký důraz na bezpečnost aplikací, a proto úzkostlivě dbá i na dobré zabezpečení formulářů. Dělá to zcela transparentně a nevyžaduje manuálně nic nastavovat. Ochrání vaše aplikace před útokem Cross Site Scripting (XSS) i Cross-Site Request Forgery (CSRF), odfiltruje ze vstupů kontrolní znaky, ověří validitu UTF-8 kódování nebo jestli nejsou položky vybrané v select boxech podvržené atd.*“

„*Použitím Nette\Forms se vyhneme celé řadě rutinních úkolů, jako je třeba psaní dvojí validace (na straně serveru a klienta), minimalizujeme pravděpodobnost vzniku chyb a bezpečnostních děr.*“[6]

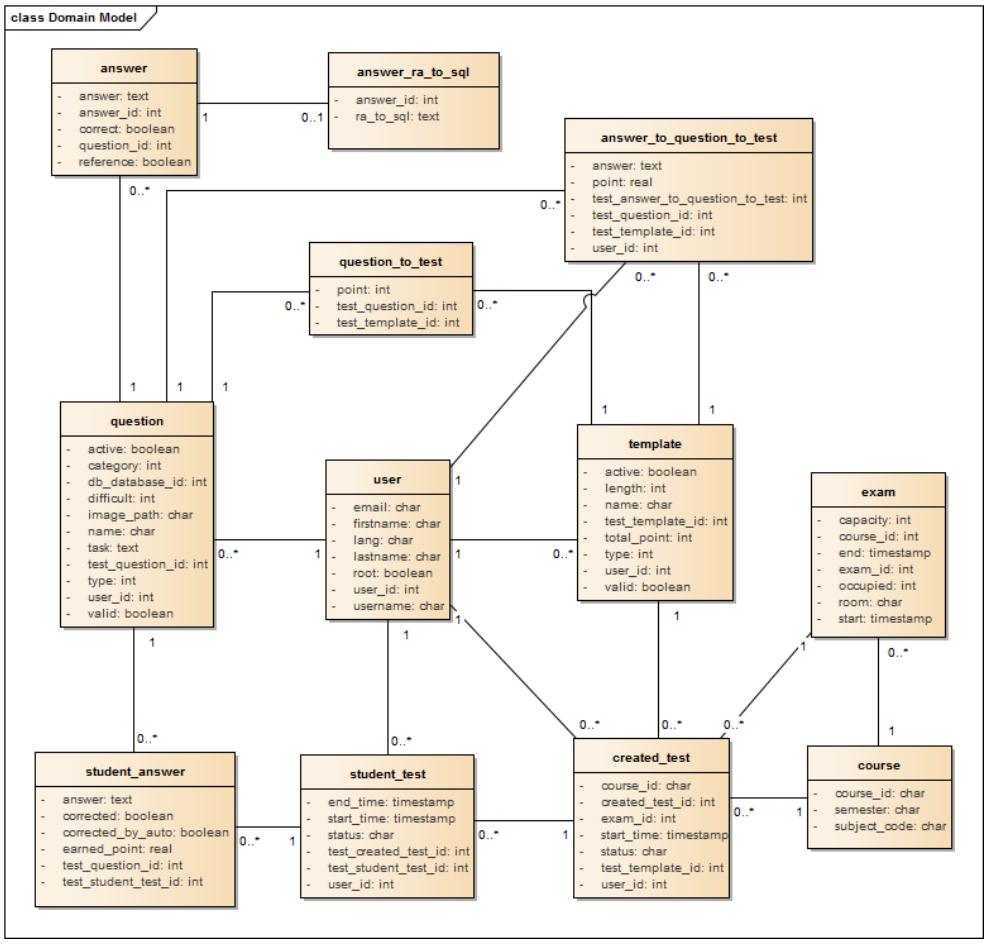
Nette/Forms nám poskytuje validační funkce přímo ve formuláři. Využívám tedy hojně validační pravidla na kontrolu vyplnění textového pole a kontrolu hodnoty, že je číslo a je v daném rozsahu.

3.1.5 Nette/Security

Nette/Security [7] zajišťuje přihlašování a odhlašování uživatelů, ověření uživatelských oprávnění a zabezpečení proti zranitelnosti.

3.1.6 PostgreSQL

Pro ukládání dat jsme zvolili databázový systém PostgreSQL [8]. Nette má pro PostgreSQL výbornou podporu, součástí je knihovna Nette/Database, která usnadňuje databázové dotazy. PostgreSQL byla taktéž použita na starém systému. Aktuální databázovou strukturu vidíme na obrázku 3.1.



Obrázek 3.1: Schéma databáze

3.1.7 AdminLTE 2

Šablonu na vzhled jsme použili AdminLTE 2 [9]. Šablona podporuje responsivní web, je postavena na CSS a Bootstrap 3. Využívá Bootstrap3 komponenty a přestylizuje je do jednotného designu. Tuto šablonu jsme vybrali především proto, že byla použita na starém systému. V rámci zachování vzhledu to byla jediná reálná možnost.

3.1.8 Grido

Pro tvorbu tabulek (testů, otázek, aj.) jsme vybrali Grido DataGrid pro Nette [10]. Je to komponenta, která vypisuje data do tabulek a usnadňuje práci s jednotlivými řádky tabulek. Grido nabízí možnost vytvářet filtry pro každý sloupec tabulky. Grido je kompatibilní s knihovnou Nette/Database. To pro nás znamenalo, že jsme data z databáze vybrané pomocí Nette/Database mohli

předat Gridu a to se postaralo o jejich výpis. Nicméně Grido nebylo šťastnou volbou. V průběhu implementace jsme zjistili, že neuspokojí všechny naše požadavky na práci s tabulkami. Museli jsme doimplementovat část funkcionality pro práci s akčními tlačítky.

3.1.9 NicEdit

WYSIWYG editor NicEdit [11] jsem použil pro tvorbu zadání otázky. NicEdit je textový editor umožňující v textovém poli vytvořit nadpisy, seznamy, nebo formátovat písmo. NicEdit toho umí ještě více, ale to my už nevyužijeme. NicEdit je integrován pomocí javascriptu.

3.2 Implementace otázek

3.2.1 Tvorba otázky

Tvorba otázky je téměř totožná s tvorbou na starém systému [1]. Získáme od uživatele stejná data, jako tomu bylo na starém systému. Rozdíl je, že ve starém systému se ve formuláři, který tvořil otázky, tvořily také odpovědi. Nyní se odpovědi tvoří dodatečně, po vytvoření otázky. Je to z důvodu přehlednosti a také, pro každý typ otázky tvoříme jiný typ odpovědi. To znamená, že je pro nás jednodušší vytvořit si pro každý typ odpovědi jiný formulář, který bude přesně vystihovat daný typ. Než mít jeden velký responzivní formulář.

3.2.2 Validace otázky

Téměř po každé akci s otázkou (viz obrázek 2.1) se provede validace otázky. Validováním otázky je myšleno, že se provede kontrola rozhodující, zda otázka splňuje požadavky a smí být použita v testu.

Implementaci validační funkce můžete vidět níže.

```
/**
 * @param $question_id
 * @return bool
 */
protected function isValidQuestion($question_id)
{
    $question = $this->questionModel->getQuestionById($question_id)
            ->toArray();
    if (!$question['name']) return false;
    if (!$question['task']) return false;
    if (!$question['type']) return false;
    $answers = $this->answerModel
            ->getReferenceAnswersByQuestion($question_id);
    $good = false;
    $onlyOneGood = true;
    foreach ($answers as $answer)
```

```
            if ($answer['correct']) {
                if ($good)
                    $onlyOneGood = false;
                $good = true;
            }
        if (!$good)
            return false;
        if ($question['type'] == 1 && !$onlyOneGood) //Radio Box
            return false;
        return true;
    }
```

Funkce kontroluje, zda otázka obsahuje název, zadání a typ (je to minimum pro přiřazení do testu). Dále kontroluje, zda je k otázce přiřazena alespoň jedna správná referenční odpověď (u typu 1 (*Zaškrtávací s jednou správnou odpovědí*) musí být pouze jedna správná).

3.2.3 Uložení otázky

Otázky jsou uloženy v databázi v tabulce *question*. Do databáze uložím data zadaná do formuláře a reference na tabulky *user* představující autora otázky a *db_database*, což je databáze přiřazená k otázce (vyplňuje se jen u odpovědí typu SQL/RA).

Do tabulky *answer* se ukládá odpověď, každá odpověď musí mít referenci na *question* - viz obrázek 3.1.

V průběhu vývoje přibyla ještě tabulka *answer_ra_to_sql*, která se využívá jen u otázek na relační algebru. Tabulka byla přidána z důvodu optimalizace při autoopravě. Je v ní uložen překlad referenční odpovědi v relační algebře do jazyka SQL, díky tomu už ji nemusíme překládat při každé opravě.

3.2.4 Zobrazení otázek

Otázky vykreslujeme do Grida, pro každou otázku máme 5 možných akcí (View, Edit, Duplicate, Manage answers, Delete) - viz obrázek 3.2.

Name	Difficult	Answer type	Category	Language	Author	Actions
						Search Reset
Otázka	Medium	Text	SQL	cs	Valenta	View Edit Duplicate Manage answers Delete

Obrázek 3.2: Grido tabulka otázek

3.2.5 View

Akce *View* zobrazí otázku v modálním okně. V modálním okně máme vypsáno: zadání otázky, nastavení otázky (typ, kategorie, obtížnost a jazyk), obrázek (pokud je) a odpovědi přiřazené k této otázce.

3.2.6 Edit

Akce *Edit* nás přesměruje na tvorbu otázky, formulář na tvorbu otázky se vyplní z dat uložených u otázky. Do editace se dá vstoupit pouze tehdy, pokud otázka nebyla použita v testu. Pokud byla otázka použita v testu, uživatel bude varován, že úpravou této otázky může způsobit nejasnosti u testů, v nichž byla otázka použita (pokud vyučující pozmění zadání).

3.2.7 Duplicate

Akce *Duplicate* vytvoří kopii otázky. Otázku přejmenuje tak, že na konec původního jména přidá číslo počínaje od 1 takové, aby celkový název byl jedinečný např. 'Starý název (4)'. Ostatní položky pouze zkopíruje. Pro odpovědi na duplikovanou otázku vytvoří vlastní duplikáty.

3.2.8 Manage answers

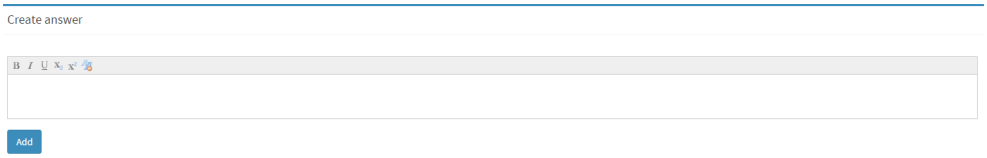
Tato akce se zpočátku může zdát jako zbytečná, protože u nově vytvořené otázky je zpravidla jedna referenční odpověď. Proč tedy potřebujeme nějaký systém na management odpovědí? Odpovědi na otázku nám totiž přibývají s každým napsaným testem a pokud někdy v průběhu opravování uznám jako správnou odpověď takovou, která správná není, musím mít možnost tuto odpověď nějakým způsobem změnit.

3.2.9 Delete

Akce *Delete* smaže otázku a k ní přiřazené odpovědi. V akci *Delete* nejde otázku smazat, pokud je použita v testu. Smazání otázky by mohlo způsobit kritické chyby. Např. kdybychom smazali otázku, která by byla přiřazena v testu a tento test spustili, studenti by neměli šanci na vyplnění otázky.

3.2.10 Tvorba odpovědi

Tvorba odpovědi je, jak už jsem uvedl, závislá na typu otázky. Nicméně rozdíly nejsou zas až tak zásadní. Vždy dostáváme jako odpověď nějaký text (výjimkou je *Diagram*). Tvorba odpovědi může vypadat následovně - viz obrázek 3.3.



Obrázek 3.3: Tvorba odpovědi

3.2.10.1 Diagram

Tvorba referenčního diagramu se provádí v kreslícím nástroji, který vytvořil Jiří Slavotínek. V nástroji se vytváří entity, kterým se přiřazují atributy. U atributů se zvolí, zda je primární, povinný nebo volitelný. Jednotlivé entity se propojují relací. Relace si zvolím, zda je povinná či volitelná, a zda je identifikující, dále se volí násobnost na obou stranách relace.

3.2.10.2 Označení správných odpovědí

Tento bod se týká pouze otázek typu *Zaškrtávací s jednou správnou odpovědí* a *Zaškrtávací s více správnými odpověďmi*. Odpověď se nám jako výchozí označí jako nesprávná a my tedy musíme vybrat správné odpovědi a ty označit jako správné.

3.2.10.3 Smazaní odpovědi

Pokud odpověď, kterou jsme vytvořili, není z nějakého důvodu správná, musíme jí smazat a vytvořit správnou referenční odpověď.

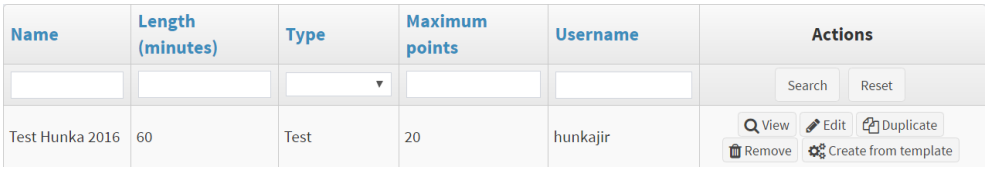
3.3 Implementace šablon testů

3.3.1 Tvorba šablony testu

Na vytvoření šablony testu použijeme formulář, který se příliš neliší od formuláře na starém systému. Potřebujeme od uživatele získat ta stejná data. Uživatel tedy zvolí název šablony, čas pro daný test, zda je šablona určena pro test v semestru, zkoušku, nebo jde o demo test a celkový počet bodů za test.

3.3.2 Přiřazení otázek k šabloně

Do šablony musíme přiřadit nějaké otázky. Můžeme použít ruční přiřazení, kde postupně vybírám otázky a k nim určuji bodové ohodnocení. Další varianta je automatické generování otázek.



Obrázek 3.4: Grido tabulka šablon

3.3.2.1 Manuální přiřazení

Při manuálním přiřazení si vyučující vybere ze všech validních otázek, otázky jsou zobrazeny v Grido tabulce, takže má k dispozici filtry pro každý sloupec (stejná tabulka je použita v zobrazení otázky - viz obrázek 3.2). Rozdíl je v akcích, které můžeme dělat s otázkou, zde to je pouze přidání otázky (*Add*) do testu (po kliknutí na přidání se zvolí počet bodů za otázku), nebo druhá možnost odebrat otázku z testu (*Remove*). Možnost odebrání se vztahuje pouze na otázky, které jsou přiřazené v testu.

3.3.2.2 Automatické generování

Před automatickým generováním zvolím, kolik otázek chci vygenerovat do testu z dané kategorie. Další možnost, jak ovlivnit generování, je omezit okruh otázek, ze kterých se bude vybírat. Prvním omezením může být možnost, generovat jen ty otázky, u kterých je aktuálně přihlášený uživatel uveden jako autor. Druhé omezení je povinné, je to výběr jazyka otázky. Ten bude stejný u všech otázek.

3.3.3 Uložení šablony testů

Šablony testů se ukládají do tabulky *template*, máme uložen název šablony, údaj o délce testu, typ testu (test, zkouška), celkový počet bodů a referenci na tabulku *user*, což je autor šablony.

V tabulce *question_to_test* jsou uloženy otázky, přiřazené k testu. U každé přiřazené otázky je uložen počet bodů a reference na tabulky *question* a *template*.

3.3.4 Zobrazení šablony testů

Všechny vytvořené šablony zobrazíme pomocí Grida - viz obrázek 3.4. Se šablonou můžeme dělat následující akce:

3.3.4.1 View

Akce *View* zobrazí šablonu v modálním okně. V modálním okně se zobrazí Grido tabulka otázek přiřazených do šablony. Používám stejnou Grido tabulku

jako u tvorby otázek (viz obrázek 3.2), s omezenými možnostmi (mohu otázky pouze zobrazit).

3.3.4.2 Edit

Akce *Edit* nás přesměruje na tvorbu šablony, formulář na tvorbu šablony vyplní z dat uložených v šabloně. Do editace se dá vstoupit pouze tehdy, pokud ze šablony nebyl vytvořen test. Jakmile byl ze šablony vytvořen test, nebude možné ji změnit. Kdybychom změnili šablonu, ze které byl test vytvořen, způsobilo by to nekonzistence v databázi.

3.3.4.3 Duplicate

Akce *Duplicate* vytvoří kopii šablony. Šablonu přejmenuje tak, že na konec původního jména přidá číslo počínaje od 1 takové, aby celkový název byl jedinečný např. 'Starý název (4)'. Ostatní položky pouze zkopíruje. Pro otázky v duplikované šabloně se vytvoří vlastní duplikáty.

3.3.4.4 Delete

Akce *Delete* smaže šablonu a k ní přiřazené otázky. Když je šablona použita v testu, nejde žádným způsobem smazat. Smazání šablony by mohlo způsobit kritické chyby a nekonzistence.

3.3.4.5 Create from template

Akce *Create from template* vytvoří test ze šablony. Šablona musí být validní. Vytvoří nový záznam v tabulce *created_test*, v tomto záznamu bude reference na *template* tedy šablonu, ze které je test vytvořen, reference na *user* tedy uživatele, který test vytvořil a nastaví se do stavu *připraven* - viz obrázek 2.6.

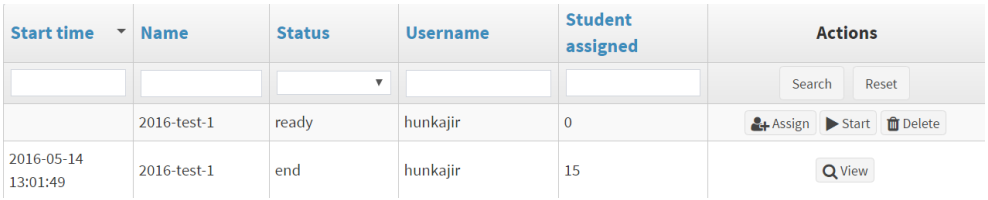
3.4 Implementace testů

3.4.1 Tvorba testu

Test vytvoříme z validní šablony (viz *Create from template*). Test se vytvoří s prázdnými poli *start_time* a *exam_id*. Obě se případně doplní v dodatečně.

3.4.2 Přiřazení studentů k testu

Z požadavků uvedených na obrázku 2.5 víme, že přiřazovat studenty chceme jak jednotlivě, tak po skupinách (paralelky, přihlášení na zkoušku). Přiřazením jednoho studenta vytvoříme záznam v tabulce *student_test*, takto vytvořený test je v počátečním stavu *Připraven* - viz obrázek 2.7.



Obrázek 3.5: Grido tabulka testů

3.4.2.1 Přiřazení paralelky

Paralelky máme uloženy v databázi v tabulce *parallel*. V tabulce *usertoparallel* jsou uloženi studenti přihlášení do dané paralelky. Postup přiřazení studentů do testu je stejný jako u jednotlivce.

3.4.2.2 Přiřazení zkouškového termínu

Zkouškové termíny máme uloženy v databázi v tabulce *exam*. V tabulce *usertoexam* jsou uloženi studenti přihlášení na daný zkouškový termín. Zde navíc potřebujeme před použitím tabulku *exam* aktualizovat, protože studenti se mohli na daný termín přihlásit den předem. Postup přiřazení studentů je stejný jako u jednotlivce.

3.4.3 Zobrazení testu

Všechny vytvořené testy zobrazíme pomocí Grido tabulky - viz obrázek 3.5. S testem můžeme dělat následující akce:

3.4.3.1 Assign

Akce *Assign* nás přesměruje na stránku s přiřazováním studentů. Pokud se jedná o zkouškový test, přesměruje nás na Grido tabulku zkouškových termínů, pokud se jedná o normální test, zobrazí se nám tabulky studentů a paralelek. Jak vidíme na obrázku 3.5, akce *Assign* není dostupná u běžících, nebo ukončených testů.

3.4.3.2 Delete

Akce *Delete* smaže test. Opět si můžeme všimnout, že jakmile test již není ve stavu *Připraven*, ale je minimálně ve stavu *Spuštěn*, už nejde smazat. Je to opět z důvodů zachování konzistence.

3.4.3.3 View

Akce *View* nám zobrazí studenty přiřazené do testu a jejich vypracované odpovědi. Po zobrazení studentovy odpovědi máme možnost jí obodovat (takto

obodovaná odpověď nebude distribuována do ostatních testů).

3.4.3.4 Start

Akce *Start* spustí test. Změní se stav na *Spuštěn* a zaznamená *start_time*. Zároveň všem přiřazeným studentům změní jejich stav testu na *Spuštěn*.

3.4.4 Vypracování testu

Po spuštění testu vyučujícím, máme možnost jako studenti si test otevřít a začít pracovat. Otevřením testu se změní stav na *Běžící* a zaznamená se nám čas spuštění. V testu můžeme vyplňovat otázky a ukládat. Po vypracování všech otázek, test uložíme. Uložením testu spustíme automatickou opravu, která se pokusí opravit otázky (některé z principu nejdou). Potom změní stav testu na *Dokončen*, nebo *Čeká na manuální opravu* - viz obrázek 2.7.

3.5 Implementace oprav testů

Opravy testů rozdělíme na automatickou opravu, která se provede po ukončení testu, a manuální opravu. Manuální oprava vyučujícím se provádí u otázek, které nebyly vyhodnoceny automaticky.

3.5.1 Automatická oprava

Automatická oprava nejprve porovná odpovědi na přesnou shodu s referenční odpovědí (tabulka *answer*). Pokud se shoda najde, ohodnotí otázku maximálním počtem bodů.

Ve druhé fázi se pokusí porovnat studentovu odpověď s odpověďmi uloženými v databázi jako nesprávné (tabulka *answer_to_question_to_test*), tyto odpovědi jsou obodované z předchozích oprav, proto za ně přidělíme dříve přidělený počet bodů.

3.5.2 Manuální oprava

Manuálně se opraví otázky, které nebyly vyhodnoceny automaticky. V manuální opravě se snažím šetřit cenný čas vyučujících, proto vyučující každou jedinečnou odpověď opraví právě jednou, tato jeho oprava se uloží a pokud se objeví v budoucnu stejná odpověď, tak už se opraví automaticky. Pokud vyučující v opravě uznal odpověď jako 100% správnou, uloží se odpověď mezi správné odpovědi (tabulka *answer*).

Pokud vyučující neohodnotí otázku 100 % uloží, se jeho ohodnocení do tabulky *answer_to_question_to_test*. V této tabulce je uložena odpověď, její ohodnocení, reference na vyučujícího (*user*), který ji ohodnotil (každý vyučující si hodnotí sám). Nakonec se ještě uloží reference na šablonu (*template*), ze které byl vytvořen test, ve kterém je tato otázka. Je to z toho důvodu,

že otázka může být ohodnocena v každém testu jiným počtem bodů. Proto je hodnocení vyučujícího závislé na maximálním počtu bodů přidělených za otázku.

3.6 Testování

V průběhu vývoje je nová aplikace průběžně testována. Díky mému vedoucímu práce, který kvůli rozsahu projektu zajistil tým studentů studujících předmět SP1, aby pomohli s tvorbou aplikace. O nasazení aplikace a její běh na serveru se průběžně stará student Oldřich Malec, který se zároveň staral o řízení práce studentů. Zčásti jsme využili studenty i k testování aplikace. Testování je realizováno pomocí Selenium testů a Nette Testeru.

3.6.1 Selenium testy

Využili jsme testy Selenium IDE [12]. Využili jsme jejich rozšíření pro Firefox. Je to nástroj, který nahrává, edituje a spouští testy. Takto nahrané testy jsou uloženy ve formátu XHTML jako soubory na disk. Byly vytvořeny testy na většinu elementárních kroků běhu aplikace (vyplnění formuláře, odeslání formuláře).

3.6.2 Nette Tester

Nette Tester [13] je použit pro kontrolu vnitřní struktury. Je využit jako Unit test, kontrolující jednotlivé třídy v aplikaci. Vytvoří tedy testovací třídu a posílá do jednotlivých metod nějaké parametry a testuje, zda metoda vrací správný výsledek.

3.7 Dokumentace

Dokumentace byla psána průběžně ve formě dokumentačních komentářů (v komentáři je uvedeno, co metoda vrací atd.). Pro tvorbu dokumentace z těchto komentářů jsme použili ApiGen [14]. ApiGen byl zvolen z důvodu jeho použití na samotném Frameworku Nette, proto je použit i zde.

Závěr

Cílem této práce bylo analyzovat starý systém pro podporu výuky studentů v BI-DBS, navrhnout nové vnitřní uspořádání aplikace s důrazem na snadnou údržbu a rozšiřitelnost. Navržené řešení naimplementovat, řádně zdokumentovat a otestovat funkčnost nové aplikace. Tyto cíle byly splněny.

Analýza probíhala už v rámci mého dřívějšího studia předmětů BI-SP1 a BI-SP2. Díky této dlouhodobé účasti na projektu jsem získal velké povědomí o problémech sužujících starý systém. Věděl jsem tedy, že většina problémů je způsobena komplikovaností ukládání dat do databáze a obecně práce s daty. To bylo způsobeno střídáním studentů vyvíjejících tento systém. V rámci předmětů se na systému vystřídaly 4 týmy studentů. Každý tým, po převzetí projektu, začal s vlastní úpravou návrhu, kterou ale nikdy nedotáhl do konce.

Moje navržené řešení vychází ze staré aplikace, ale výrazně odlehčuje návrh, který je takto více rozšiřitelný. Návrh je nyní jednodušší a díky tomu je i lépe udržovatelný. Díky důkladné analýze jsem se v návrhu vyvaroval chyb, které byly ve starém systému.

Nový systém je aktuálně (od 15.5.2016) nasazen na adrese dbs2.fit.cvut.cz a v blízké době se na něm budou psát zkouškové testy. Společně se studenty Filipem Glazarem, Oldřichem Malecem a studenty z týmů BI-SP1 jsme otestovali průběh zkoušky s reálnými daty. Celá aplikace byla v průběhu vývoje řádně testována pomocí unit testů a testovacích scénářů. Aplikace byla řádně zdokumentována, dokumentace popisuje jednotlivé moduly a třídy aplikace.

Věřím, že aplikace bude vyučujícími využívána k jejich spokojenosti, neboť jim ušetří jejich drahocený čas strávený s opravami testů.

Literatura

[1] *BI-DBS*. [cit. 2016-05-13]. Dostupné z: https://dbs.fit.cvut.cz/?locale=cs

[2] Group, T. P.: *PHP: Documentation*. [cit. 2016-05-13]. Dostupné z: http://php.net/docs.php

[3] Foundation, N.: *Dokumentace | Nette Framework*. [cit. 2016-05-13]. Dostupné z: https://doc.nette.org/cs/2.3

[4] Symfony: *Symfony, High Performance PHP Framework for Web Development*. [cit. 2016-05-13]. Dostupné z: https://symfony.com/

[5] Foundation, N.: *Databáze | Nette Framework*. [cit. 2016-05-13]. Dostupné z: https://doc.nette.org/cs/2.3/database

[6] Foundation, N.: *Formuláře | Nette Framework*. [cit. 2016-05-13]. Dostupné z: https://doc.nette.org/cs/2.3/forms

[7] Foundation, N.: *Přihlašování & oprávnění uživatelů | Nette Framework*. [cit. 2016-05-13]. Dostupné z: https://doc.nette.org/cs/2.3/access-control

[8] Group, T. P. G. D.: *PostgreSQL: Documentation*. [cit. 2016-05-13]. Dostupné z: http://www.postgresql.org/docs/

[9] Studio, A.: *AdminLTE 2 | Documentation*. [cit. 2016-05-13]. Dostupné z: https://almsaeedstudio.com/themes/AdminLTE/documentation/index.html

[10] Bugyík, P.: *Grido*. [cit. 2016-05-13]. Dostupné z: http://o5.github.io/grido-examples/documentation.cs.html

[11] Production, B. K.: *NicEdit - WYSIWYG Content Editor, Inline Rich Text Application*. [cit. 2016-05-13]. Dostupné z: http://nicedit.com/docs.php

[12] Project, S.: *Introduction & Selenium Documentation*. [cit. 2016-05-13]. Dostupné z: http://www.seleniumhq.org/docs/01_introducing_selenium.jsp

[13] Foundation, N.: *Nette Tester – pohodové testování | Nette Framework*. [cit. 2016-05-13]. Dostupné z: https://tester.nette.org/cs/

[14] ApiGen: *ApiGen | Smart and Readable Documentation for your PHP project*. [cit. 2016-05-13]. Dostupné z: http://www.apigen.org/

Příloha **A**

Seznam použitých zkratek

BI-DBS Databázové systémy

BI-SP1 Softwarový týmový projekt 1

BI-SP2 Softwarový týmový projekt 2

PHP Hypertext Preprocessor

SQL Structured Query Language

RA Relační algebra

RAT Relational Algebra Translator

XML Extensible Markup Language

KOS Komponenta studium

WYSIWYG What you see is what you get

XHTML Extensible Hypertext Markup Language

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `latex` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `BP_Petr_Pejša_2016.pdf` ..... text práce ve formátu PDF