FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Frontend manuální korektury otázek v portálu dbs.fit.cvut.cz
Student:	Tomáš Chalupa
Vedoucí:	Ing. Jiří Hunka
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem této práce je návrh a realizace frontendové části portálu dbs.fit.cvut.cz(dále jen portál) zabývající se opravou vyplněných odpovědí z testů studentů databázových předmětů. Dále má tato práce navázat na práci Bc. Andriiho Plyskache, který připravil nový backend testů.

Postupujte v těchto krocích
1. Analyzujte současný stav oprav odpovědí z testů u vyučujících v portálu.
2. Na základě analýzy navrhněte lepší a efektivnější způsoby opravy otázek tak, abyste minimalizovali nárůst složitosti pro vyučující a zároveň zachovali nebo zlepšili současné možnosti opravy.
3. Na základě analýzy implementujte minimálně funkční prototyp.
4. Realizujte vhodné sady testů a ověřte použitelnost Vašeho řešení na cílových uživatelích.
5. Integrujte Vaše řešení po odladění prototypu do portálu nebo přichystejte vaše řešení pro budoucí snadnou integraci.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 22. ledna 2022 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Frontend manuální korektury otázek v portálu dbs.fit.cvut.cz

Tomáš Chalupa

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jiří Hunka

11. května 2022

Poděkování

Na tomto místě bych rád poděkoval vedoucímu této bakalářské práce panu Ing. Jiřímu Hunkovi za cenné rady a připomínky, které mi po celou dobu tvorby práce poskytoval. Dále bych chtěl poděkovat Ing. Oldřichu Malcovi za poskytnutí dat pro testování. Nemohu zapomenout ani na cvičící předmětu BI–DBS, kteří se mnou během tvorby této práce sdíleli své cenné uživatelské zkušenosti s DBS portálem. Jmenovitě tímto děkuji Ing. Janu Blizničenkovi, Ing. Cyrilu Černému, Ing. Janu Matouškovi, Ing. Šimonu Schierreichovi a Ing. Davidu Šenkýřovi. Děkuji také své rodině za podporu nejen během tvorby této práce, ale i během celého studia na FIT.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principu při 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 předpisu. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisu, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programu, 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 zpusobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelum). 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 zpusobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 11. května 2022 . . . . . .

Odkaz na tuto práci

Chalupa, Tomáš. *Frontend manuální korektury otázek v portálu dbs.fit.cvut.cz.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Abstrakt

Tato práce se zabývá návrhem a realizací nového frontendu manuálních oprav testů v portálu dbs.fit.cvut.cz. V práci je nejprve provedena analýza aktuálního stavu manuálních oprav. V rámci analýzy se práce dále zabývá získáváním požadavků na nový frontend formou rozhovorů s cvičícími předmětu BI–DBS. Na základě analýzy je v práci popsán návrh nového frontendu, kde jsou hlavní prvky znázorněny pomocí wireframes. Dále práce popisuje klíčové implementační postupy použité při realizaci řešení. Poslední kapitola se pak věnuje testování výsledku práce s koncovými uživateli.

Klíčová slova DBS portál, frontend, web design, JavaScript, Vue.js

Abstract

This thesis deals with the design and implementation of a new frontend for test answers evaluation in the dbs.fit.cvut.cz portal. The thesis first analyses the current state of test evaluation. Within the analysis, the thesis further deals with requirements elicitation for the new frontend through interviews with teachers of the BI–DBS course. Based on the analysis, the thesis describes the design of the new frontend where the main elements are represented using wireframes. Furthermore, the thesis describes the key implementation procedures used in the realization of the solution. The last chapter focuses on testing the final product with end users.

Keywords DBS portal, frontend, web design, JavaScript, Vue.js

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Existuje celá řada webových aplikací určených na podporu výuky ve školách. Jejich posláním je zefektivnit práci jak vyučujících, tak studentů a v řadě škol jsou dnes nedílnou součástí vzdělávacího procesu. Tyto aplikace mohou sloužit například ke klasifikaci studentů, distribuci studijních materiálů, zadávání a odevzdávání úkolů, psaní testů a podobně. Některé předměty či kurzy pak mohou mít další specifičtější požadavky na to, co by měl systém na podporu jejich výuky umět.

Také na Fakultě informačních technologií ČVUT jsou tyto aplikace využívány a můžeme se zde setkat právě i s řadou systémů, které jsou určeny pro podporu výuky konkrétních předmětů či skupin předmětů. Výhodou fakulty, která vyučuje informatiku, je to, že si může tyto systémy sama vyvíjet a do vývoje zapojit studenty. Ti tak mají příležitost vyzkoušet si v rámci studia práci na systémech, které budou mít v praxi reálné využití, a budou je využívat přímo oni nebo jejich kolegové.

Jedním ze systémů, který na FIT ČVUT vzniká právě tímto způsobem, je DBS portál. Nachází se na adrese dbs.fit.cvut.cz a využívá se k podpoře výuky databázových předmětů, především pak předmětu Databázové systémy (BI–DBS). S portálem přijdou během studia do kontaktu všichni studenti, protože BI–DBS je povinný předmět. Portál funguje již několik let a jeho vývoj stále probíhá. Slouží především pro odevzdávání a opravu semestrálních prací a pro psaní zápočtových a zkouškových testů (tzv. testový modul).

Testový modul umožňuje vyučujícím vytvářet testy a ty pak studenti prostřednictvím portálu vyplňují. Oprava testů probíhá z velké části automaticky, ale v některých případech musí odpovědi studentů opravovat vyučující.

Na vývoji portálu se podílejí studenti v rámci předmětů Softwarový týmový projekt 1 (BI–SP1) a Softwarový týmový projekt 2 (BI–SP2) a také v rámci bakalářských či diplomových prací. Vývojáři DBS portálu v současné době vkládají největší úsilí právě do vývoje testového modulu. Pracuje se na jeho nové verzi a součástí tohoto projektu je také tato práce.

Na počátku tvorby nového testového modulu byla bakalářská práce Andriiho Plyskache, která se zaobírá návrhem backendu. Z ní se pak vychází během dalšího vývoje nového testového modulu a do jisté míry na ní tato práce také navazuje.

Cílem této práce je navrhnout a realizovat frontend manuálních oprav v nové verzi testového modulu DBS portálu. Nové uživatelské rozhraní bude využíváno vyučujícími k opravě odpovědí studentů na otázky, které portál nedokázal opravit automaticky. V aktuální verzi již portál možnost manuálních oprav plně poskytuje, ale vzhledem k probíhajícímu vývoji nové verze celého testového modulu je nutné předělat také tuto část.

Před samotným návrhem a realizací nového frontendu bude provedena analýza. V rámci ní se musíme nejprve důkladně seznámit s testovým modulem a popsat důležité aspekty modulu, které s opravou testů souvisí. Dále je třeba také zanalyzovat, co doposud používaná verze manuálních oprav nabízí a jaké jsou její případné nedostatky.

Na základě analýzy bude vytvořen návrh. Oprava otázek by měla být v nové verzi efektivnější a pro vyučující jednodušší. Návrh pak bude sloužit jako podklad při samotné implementaci. Realizované řešení se následně otestuje na koncových uživatelích, tedy vyučujících předmětu BI–DBS.

Výsledkem této práce bude nový frontend pro manuální opravy testů, který bude integrován do DBS portálu a po vytvoření backendu a dalších částí nového testového modulu jej bude možné začít používat při výuce.

Kapitola 1

Analýza

Předtím, než začneme s navrhováním a vytvářením nového frontendu pro manuální opravy studentských odpovědí v DBS portálu, musíme se nejprve s celou problematikou detailněji seznámit.

Výstup z této práce má, po vytvoření souvisejících součástí, nahradit frontendovou část, která se u manuálních oprav používá nyní. Je proto důležité podívat se na její aktuální stav a zjistit, na co jsou uživatelé zvyklí a jakým způsobem by se případně dalo uživatelské rozhraní vylepšit.

V této kapitole se nejprve seznámíme s DBS portálem jako takovým. Více se pak zaměříme na část, která je určena pro psaní zápočtových a zkouškových testů (testový modul). Dále tato kapitola popisuje, jakým způsobem v portálu probíhá opravování studentských odpovědí, což je velmi důležitá část testového modulu.

Tato práce má navázat na bakalářskou práci Andriiho Plyskache [1], která mění backend testového modulu DBS portálu. Proto je důležité zjistit, jaké změny se v testovém modulu plánují a s těmito změnami dále v této práci počítat.

Výstupem této analýzy musí být především jasný popis toho, jaké má mít nový frontend vlastnosti. V této kapitole se proto budeme věnovat také získávání a specifikací požadavků. Bude nás zajímat především pohled samotných vyučujících předmětu BI–DBS, kteří mají s použitím současné verze portálu praktické zkušenosti.

1.1 DBS portál

DBS portál je webová aplikace sloužící pro podporu výuky předmětu BI–DBS (Databázové systémy) na FIT ČVUT. Počátky vývoje portálu se datují do roku 2013, kdy s nápadem na jeho realizaci přišel Jiří Hunka [2]. Úkolem portálu je co nejvíce zautomatizovat výuku předmětu a tím zefektivnit

práci vyučujících i studentů. Portál je využíván například pro odevzdávání a automatickou kontrolu semestrálních prací.

Další významnou částí portálu je modul pro psaní testů. Dříve psali studenti testy na papír, což nebylo efektivní a tak se začalo pracovat na první verzi rozšíření DBS portálu o testovou část. V roce 2016 vznikla bakalářská práce *Systém pro podporu testování studentů v BI-DBS* od Petra Pejši [3], která se vývojem tohoto modulu zabývala. Více se této části portálu věnuje kapitola 1.2.

V roce 2016 se portál začal poprvé používat při výuce a o rok později už ho používali všichni cvičící [2]. Postupem času byl portál rozšiřován a vylepšován a tento proces trvá do současnosti.

1.2 Testový modul DBS portálu

Tato kapitola se zabývá podrobnějším popisem toho, jak tento modul v současnosti funguje a jaké nabízí možnosti, zejména se zaměřením na vlastnosti, které jsou klíčové pro tuto práci.

Modul má část určenou jak pro vyučující tak také pro studenty. Vyučující mohou vytvářet zadání a otázky, které pak vkládají do testových šablon, ze kterých poté vznikají konkrétní testy. K těmto testům lze přiřazovat studenty, kteří mají test psát. Přiřazení studenti na otázky v testu odpovídají a následně test odevzdají.

1.2.1 Zadání a otázky

Zadání v portálu vytváří garant nebo vyučující. Slouží k tomu, aby se přiřadilo k otázkám. Jedno zadání může být přiřazeno k více různým otázkám a jedna otázka může obsahovat více zadání. Zadání se využívá například pro nějaký delší výchozí text, nebo nějaký obecnější kontext, který je potřeba studentovi předat, aby poté mohl odpovídat na konkrétní otázky.

1.2.1.1 Typy zadání

V portálu je aktuálně možné vytvářet zadání těchto typů: *text*, *obrázek*, *diagram*, *normalizace relačního schématu*. Před vytvářením zadání je nutné zvolit jeden ze zmíněných typů. U každého zadání se zadává jeho název a jazyk.

U zadání typu *text* se při jeho vytváření píše text do textového pole. Zadání typu *obrázek* umožňuje nahrát jeden obrázek z disku. Zadání typu *diagram* vyžaduje při jeho vytváření nakreslit diagram pomocí komponenty pro kreslení konceptuálního schématu. Tato komponenta (tzv. *Kreslítko*) je součástí portálu a využívá se na více místech. Při vytváření zadání typu *Normalizace* se vyučujícímu zobrazí formulář, do kterého může zadat atributy relace a funkční závislosti.

1.2.1.2 Vytváření otázky

Při vytváření otázky se nejprve musí vyplnit základní údaje k otázce: *název*, *úkol*, *obtížnost*, *kategorie*, *úroveň*, *jazyk* a *typ odpovědi*. *Úkolem* je myšleno samotné znění otázky, které se studentovi zobrazí při vyplňování testu společně se zadáním. Položka *obtížnost* umožňuje rozdělit otázky na lehké, středně těžké a těžké. Dále je nutné otázku zařadit do nějaké kategorie, podle toho, k jakému tématu se otázka vztahuje. V současné době jsou na výběr čtyři kategorie: *SQL*, *RA (relační algebra)*, *model* a *teorie*.

Pomocí volby úrovně otázky je možné otázky rozdělit do různých skupin podle toho, zda se hodí na demo test, test v semestru nebo na zkoušku. Poslední údaj, který se při vytváření otázky vyplňuje je *typ odpovědi*. Typy odpovědí a rozdíly mezi nimi jsou pro tuto práci důležité a jsou rozebrány v kapitole 1.2.1.3.

Aby byla otázka validní (šla zařadit do testů) je nutné k ní ještě přiřadit zadání a vyplnit alespoň jednu referenční odpověď. Formulář pro přidání referenční odpovědi se liší podle typu odpovědi. Referenční odpověď se zobrazuje vyučujícím při manuálním hodnocení, ale slouží také pro automatické opravování, které je popsáno v kapitole 1.2.2.

1.2.1.3 Typy odpovědí

Při vytváření otázky je nutné zvolit typ odpovědi. Aktuálně možné typy odpovědí jsou: *text*, *checkbox*, *radio list*, *diagram*, *SQL*, *RA*, *transformace* a *normalizace relačního schématu*.

Odpověď typu *text* po studentovi vyžaduje, aby napsal krátkou textovou odpověď na zadanou otázku. U odpovědí typu *radio list* i *checkbox* dostane student na výběr ze seznamu možností (možnosti lze definovat ve správě odpovědí u každé otázky). Z tohoto seznamu musí zaškrtnout všechny správné odpovědi. U typu *radio list* je správná možnost vždy jen jedna, u typu *checkbox* jich může být více.

Odpověď typu *diagram* je určena pro otázky, které testují znalosti studentů týkající se návrhu konceptuálního schématu. Odpověď tohoto typu vytvářejí pomocí nástroje na kreslení diagramů (*Kreslítko*).

U odpovědí typu *SQL* má být odpovědí SQL dotaz (aktuálně pouze dotazy typu SELECT). Podobným typem odpovědi je *RA*, která zase vyžaduje odpověď zapsanou v relační algebře. U obou typů si může student před odevzdáním testu zkontrolovat, zda lze zadaný dotaz přeložit.

U otázek, kde je typem odpovědi *transformace* se od studenta očekává, že převede zadaný konceptuální model na relační a výsledek vyjádří pomocí zjednodušeného relačního zápisu.

Vytváření otázek, kde je typem odpovědi *normalizace relačního schématu* se od ostatních otázek trochu liší. Vyučující zde může vybrat pouze zadání, které je stejného typu (tedy *normalizace relačního schématu*). Forma těchto

otázek je jasně daná a dá se rozdělit na podúlohy. Množinu podúloh, které má vytvářená otázka obsahovat vybírá vyučující při vytváření otázky. Na základě tohoto výběru a přiřazeného zadání se automaticky otázky vygenerují a k otázce se již nepřiřazují žádné referenční odpovědi. Toto řešení vzniklo v rámci bakalářské práce *Automatické generování a oprava otázek na normalizaci databáze pro předmět BI-DBS* [4], jejímž autorem je Marek Erben.

1.2.2 Automatické opravy

Jak již bylo zmíněno portál má za úkol co nejvíce zautomatizovat výuku předmětu BI–DBS. Tento požadavek se vztahuje i na modul pro testování studentů. Je proto kladen velký důraz na to, aby vyučující měli s opravováním testů co nejméně práce a zároveň aby byly opravy studentských odpovědí vždy korektní. V rámci portálu tedy vznikl a stále se vyvíjí systém automatické opravy, který se snaží co nejvíce odpovědí opravit automaticky, bez zásahu vyučujících. Základ pro automatickou opravu testů vytvořil Petr Pejša ve své bakalářské práci [3].

Automatická oprava je spuštěna vždy po odevzdání testu. Prochází každou otázku a pokouší se opravit studentovy odpovědi. Pokud je odpověď na otázku nevyplněná, je automaticky ohodnocena 0 body. Dále se odpověď porovná s referenční odpovědí. Pokud odpovídá, udělí se plný počet bodů, pokud ne, dojde k porovnání s dříve opravenými studentskými odpověďmi, které byly ohodnoceny plným počtem bodů. Jestliže je nalezena shoda s některou z těchto odpovědí, je otázka také ohodnocena plným počtem bodů. Pokud systém nedokáže odpověď opravit automaticky, je zařazena do tzv. *manuálních oprav*, kde ji musí ručně opravit vyučující.

V textu byl zatím popsán pouze obecný postup, jak automatická oprava probíhá. Konkrétní způsoby automatického vyhodnocování a porovnávání s jinými odpověďmi se liší v závislosti na typu odpovědi.

1.2.2.1 Odpovědi typu radio list a checkbox

Odpověď typu *radio list* lze vždy opravit automaticky, protože je přesně dáno, která jedna odpověď ze seznamu je správná. Typ *checkbox* lze opravit automaticky pouze v případě, že student zaškrtnul všechny správné odpovědi a žádnou špatnou nebo naopak, všechny špatné a žádnou správnou. V tomto případě je mu udělen plný počet bodů (resp. nula bodů), jinak je nutné odpověď opravit manuálně.

1.2.2.2 Odpovědi typu text

U odpovědi typu *text* lze automatickou opravu provést pouze v případě, že studentova odpověď přesně odpovídá referenční odpovědi, nebo nějaké jiné (studentské) odpovědi ohodnocené za plný počet bodů. Textové odpovědi je pak přidělen plný počet bodů.

1.2.2.3 Odpovědi typu diagram

Odpověď typu *diagram* označí automatická oprava za správnou v případě, že je diagram v opravované odpovědi s tím v referenční odpovědi totožný. Tuto informaci lze získat díky nástroji na kreslení diagramů, který umí dva diagramy porovnat [3].

1.2.2.4 Odpovědi typu SQL a RA

Pro možnost automatické opravy u těchto dvou typů odpovědí musí být pro příslušné otázky vytvořena databáze. Automatická oprava ve více fázích. Nejprve je řetězec se studentovou odpovědí (platí pro dotaz v SQL i v RA) porovnán s odpovědí referenční. Pokud se řetězce rovnají, je odpověď ohodnocena plným počtem bodů a automatická oprava končí. Jinak je řetězec stejným způsobem porovnám se dříve ohodnocenými odpověďmi od studentů. Pokud ani zde není nalezena shoda, automatická oprava pokračuje dál.

U typu *SQL* se poté zkontroluje, zda dotaz obsahuje klíčové slovo *commit*. Pokud ano, je automatická oprava ukončena a odpověď musí jít do manuálních oprav. Odpověď v *RA* je v tomto kroku převedena pomocí překladače do SQL. Pokud se dotaz nepodaří přeložit, jde odpověď do manuálních oprav.

Jestliže ani po všech předchozích krocích automatická oprava neskončila, je SQL dotaz (původní v případě odpovědi typu *SQL* nebo výsledek překladu z minulého kroku u odpovědi typu *RA*) proveden nad příslušnou databází. Poté dojde k porovnání vrácených výsledků s výsledky, které vrací dotaz v referenční odpovědi. V případě, že jsou oba výsledky stejné, je za odpověď udělen plný počet bodů. V opačném případě musí být odpověď ohodnocena manuálně.

1.2.2.5 Odpovědi typu transformace

Odpověď typu *transformace* prošla na portálu značným vývojem. Tento typ odpovědí dříve na portálu vůbec nebyl a odpovědi na otázky spojené s transformací se zadávaly pomocí typu *text*.

První verzí, která tyto odpovědi do portálu integrovala byla implementována v bakalářské práci Filipa Machaly *Automatická oprava zjednodušeného relačního zápisu* [5]. V této práci je popsáno, že odpověď studenta je normalizována do formátu JSON [1] a ten je pak porovnán s takto normalizovanou referenční odpovědí. Výsledkem porovnání pak je opravená odpověď ve formátu JSON, kde jsou označeny chyby.

Dále se podporou zjednodušeného relačního zápisu a tedy i otázkám typu *transformace* věnoval Martin Šach v bakalářské práci *Podpora tvorby a automatická oprava zjednodušeného relačního zápisu v portálu dbs.fit.cvut.cz* [7]. Tato práce popisuje aplikaci, ve které lze vstup ve zjednodušeném relačním

[1] JavaScript Object Notation – Jednoduchý textový formát, sloužící k výměně dat, nezávislý na programovacím jazyce. [6]

zápisu převést na diagram ve stejném formátu, jako je diagram v zadání. Do budoucna je v plánu při automatické opravě odpověď studenta na diagram převést a výsledek pak porovnat s diagramem v zadání. Tímto způsobem by bylo možné opravy tohoto typu zcela automatizovat. Toto řešení ale není v současné době implementováno a tak je v této práci potřeba přistupovat k opravě odpovědí tohoto typu běžným způsobem.

1.2.2.6 Odpovědi typu normalizace relačního schématu

Problematikou automatického opravování odpovědí typu *normalizace* se zabývala již zmíněná bakalářská práce Marka Erbena [4]. Autor v této práci představuje knihovnu, pomocí které je možné všechny odpovědi tohoto typu hodnotit automaticky. Dále je zde pak uvedena váha jednotlivých podúloh, které v rámci otázek na normalizaci existují. Počet bodů, které student za svou odpověď obdrží je pak vypočítán pomocí vzorce, který je v práci také uveden. Hodnocení odpovědí tohoto typu je proto již plně automatizováno.

1.2.3 Manuální opravy

Pokud se odpověď na otázku nepodaří opravit během automatických oprav, směřuje odpověď do oprav manuálních. Zde odpověď musí zkontrolovat a body udělit některý z vyučujících. Ten má také možnost napsat k odpovědi komentář, který se potom zobrazí studentovi, nebo více studentům, kteří odpověď napsali.

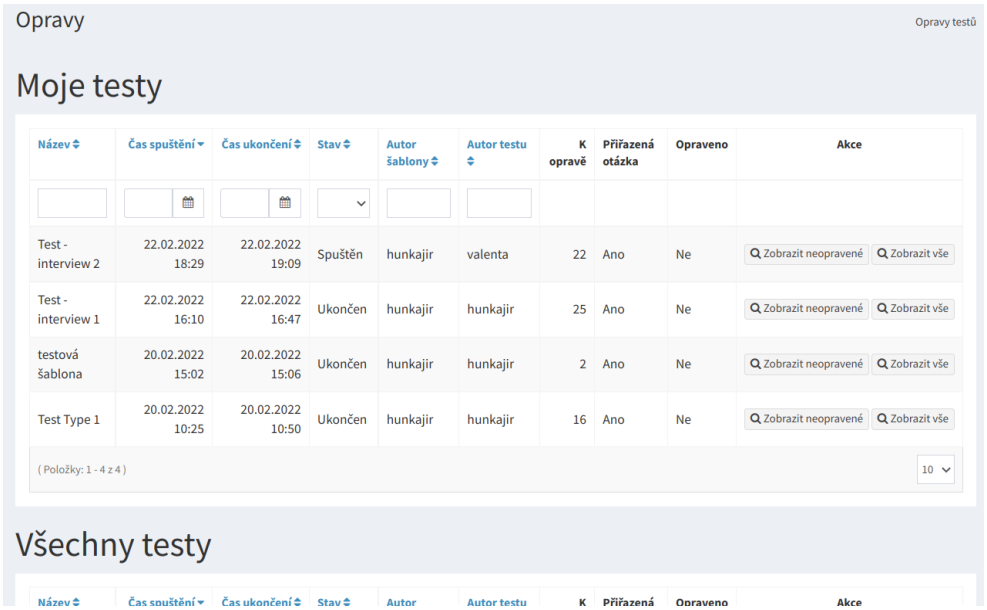
Portál se i zde snaží opravu co nejvíce zjednodušit a zautomatizovat a tak když vyučující ohodnotí nějakou odpověď, a to jakýmkoliv počtem bodů, jsou totožné odpovědi od ostatních studentů ohodnoceny stejným počtem bodů a je k nim přiložen také stejný komentář. Pokud by se totožná odpověď objevila i někdy jindy v testu vytvořeném ze stejné testové šablony, je (i částečný) počet bodů přidělen automaticky a takové odpovědi se nemusí znovu opravovat manuálně.

Portál umožňuje vyučujícím procházet si automaticky opravené odpovědi a v případě, že by nesouhlasili s výsledkem automatického hodnocení, tak ho mohou i změnit. Tato část je v testovém modulu oddělena od části s manuálními opravami, ale s odpověďmi se v ní pracuje úplně stejně.

Část portálu zabývající se manuálními opravami testů je pro tuto práci velmi důležitá, protože se práce věnuje právě návrhu a tvorbě jejího nového uživatelského rozhraní. Současný stav stránek, které s manuální opravou souvisejí, je proto detailněji popsán v následující kapitole.

1.3 Současný stav stránek pro manuální opravu

V této kapitole si popíšeme to, jak v současné době vypadá z uživatelského hlediska část testového modulu, která se týká manuálních oprav. Část, ve které



Obrázek 1.1: Stránka se zobrazením testů k manuální opravě

mohou vyučující procházet a měnit odpovědi, které byly ohodnoceny automaticky obsahuje oproti těmto stránkám jen velmi málo rozdílů. Ty budou popsány v podkapitole 1.3.3.

Nachází-li se uživatel v testovém modulu DBS portálu, do části manuálních oprav se dostane kliknutím na položku „Manuální opravy“ v levém menu. Samotný proces manuální opravy otázek se dá rozdělit do dvou částí. První část je výběr testu a navigace ke konkrétní otázce, kterou chce uživatel opravovat. Druhou částí je pak samotná oprava odpovědí na vybranou otázku.

1.3.1 Zobrazení testů a otázek k manuální opravě

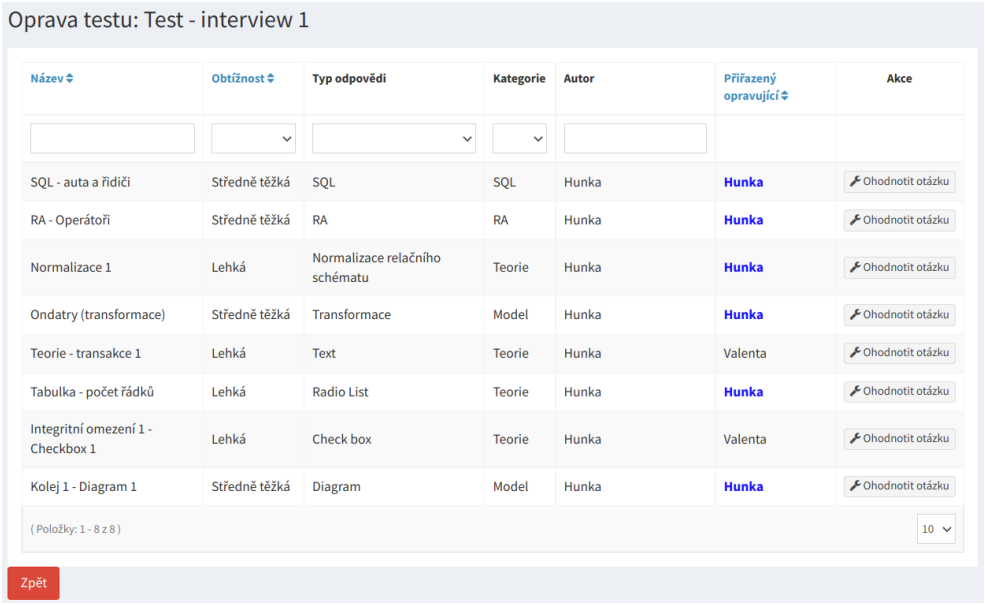
Nejprve se uživatel dostane na stránku se zobrazením testů (obr. 1.1). Celý postup od otevření manuálních oprav až po zobrazení otázek k vybranému testu popisuje následující scénář. Stránky, které mu jsou v jeho průběhu zobrazeny jsou pak detailněji popsány v textu dále.

UC1: zobrazení testů a otázek k manuální opravě

Aktéři: vyučující, garant.
Počáteční podmínka: aktér se nachází v testovém modulu DBS portálu.

Scénář:

1. Aktér se rozhodne zobrazit si odpovědi studentů na otázky z testů.



Obrázek 1.2: Stránka se zobrazením otázek k opravě

2. Aktér klikne v levém menu stránky na položku „Manuální opravy“.
3. Systém zobrazí stránku, která obsahuje dvě tabulky – „Moje testy“ a „Všechny testy“.
4. Aktér si z jedné z tabulek vybere test, ke kterému chce zobrazit otázky.
5. Aktér si ve sloupci „akce“ na příslušném řádku testu si kliknutím na jedno z tlačítek zvolí, zda chce zobrazit všechny otázky nebo jen ty neopravené.
6. Systém zobrazí tabulku s otázkami k testu.

Tabulka „Moje testy“ (popsána v bodě č. 3 scénáře UC1) obsahuje ve svých řádcích testy, které mají šablonu, kterou vytvořil právě přihlášený uživatel. Tabulka „Všechny testy“ (tentýž bod) obsahuje všechny testy, které se psaly (včetně těch kompletně opravených automaticky). Obě tabulky mají stejné sloupce: název, čas spuštění, čas ukončení, stav, autor šablony, autor testu, k opravě (počet odpovědí k manuální opravě), přiřazená otázka, opraveno (dvě možnosti: ano – všechny otázky jsou opraveny, ne – je potřeba manuální oprava), akce (tlačítka: „Zobrazit neopravené“ a „Zobrazit vše“).

Po zvolení jedné z akcí v předchozích tabulkách systém zobrazí tabulku s otázkami k vybranému testu (bod č. 6 scénáře UC1). V případě volby „Zobrazit neopravené“ systém zobrazí tabulku, kde jsou v řádcích pouze neopravené

otázky k danému testu. Tabulka s otázkami je na obrázku 1.2. Možnost „Zobrazit vše“ zobrazí všechny otázky, včetně těch opravených automaticky. Obě tabulky jsou jinak stejné (mají stejné sloupce). Tabulky mají tyto sloupce: název, obtížnost, typ odpovědi, kategorie, autor, přiřazený opravující, akce (tlačítko „Ohodnotit otázku“).

Nezávisle na tom, kdo je přihlášený, vidí uživatel všechny otázky. U otázek, ke kterým je právě přihlášený uživatel přiřazený jako opravující, je jeho jméno ve sloupci „Přiřazený opravující“ zvýrazněno tučným modrým písmem. Kliknutím na tlačítko „Ohodnotit otázku“ ve sloupci akce se uživatel dostane na stránku, kde hodnotí jednotlivé studentské odpovědi k jedné otázce, které vyžadují manuální opravu.

1.3.2 Opravování studentských odpovědí

Každá otázka v rámci testu má svou vlastní stránku. Stránka je nadepsána názvem otázky a dále zde jsou zobrazeny další údaje o otázce a samozřejmě také studentské odpovědi na tuto otázku, které je potřeba manuálně opravit. Ve výchozím zobrazení je stránka rozdělena na levou a pravou stranu (obrázek 1.3. Na levé straně jsou zadání k otázce a otázka. Na pravé straně se pak nachází pod sebou seřazené odpovědi studentů. Jednotlivé údaje o otázce i odpovědi jsou uvnitř vizuálně oddělených prvků. Tyto prvky budou v této práci dále označovány pojmem *box*. Boxy je obvykle možné zabalit/minimalizovat, čímž se jejich obsah skryje. Detailnějšímu popisu prvků na této stránce se bude věnovat kapitola 1.3.2.1.

Následující text popisuje scénáře týkající se stránky určené pro opravu otázky.

UC2: hodnocení otázky

Aktéři: vyučující, garant.
Počáteční podmínka: aktér se nachází v testovém modulu DBS portálu a prošel UC1.

Hlavní scénář:

1. Aktér se rozhodne, že bude opravovat odpovědi studentů u nějaké otázky ze zvoleného testu.
2. Aktér si z tabulky vybere otázku a klikne u ní na tlačítko „Ohodnotit otázku“.
3. Systém zobrazí stránku, kde se nachází údaje k vybrané otázce a odpovědi od studentů (každá má svůj vlastní box).
4. Aktér postupně prochází odpovědi od studentů, u každé z nich nastavuje počet bodů a může napsat komentář. Hodnocení u každé odpovědi potvrdí kliknutím na tlačítko „Ohodnotit“.

5. Systém hodnocení odpovědi uloží a minimalizuje box s právě ohodnocenou odpovědí.
6. Jestliže se aktér rozhodne hodnocení změnit, může obsah boxu znovu zobrazit a počet bodů i komentář upravit.
7. Pokud aktér ohodnotí všechny odpovědi na vybranou otázku, systém na to upozorní.
8. Jakmile chce aktér ze stránky s otázkou odejít, klikne na tlačítko „Zpět“.
9. Systém zobrazí tabulku odpovědí k danému testu.

Alternativní scénář 1: uživatel se rozhodne upravit zadání u otázky. Scénář začíná po bodu č. 3 hlavního scénáře.

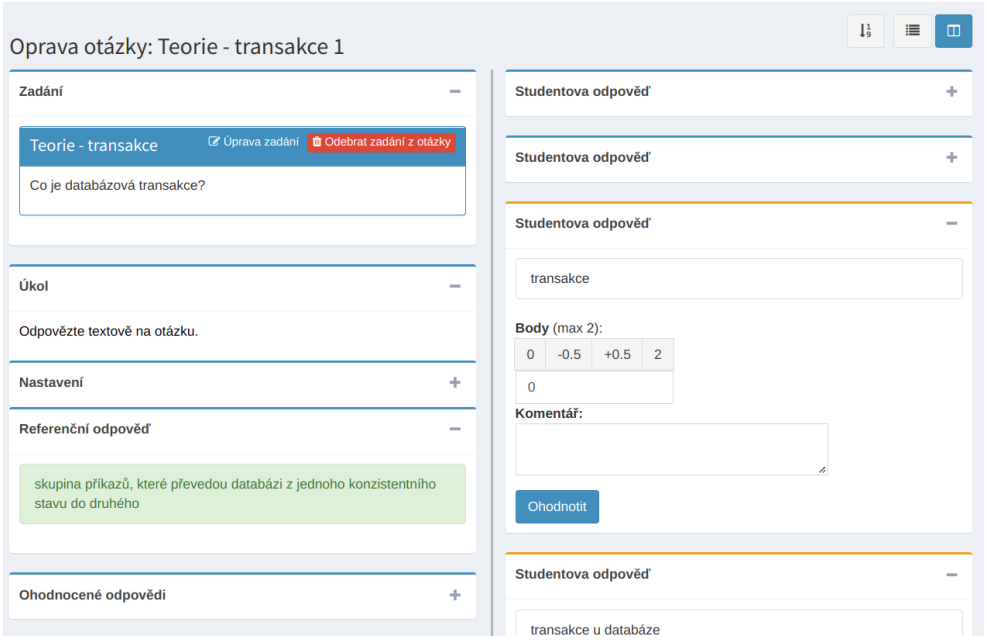
1. Aktér klikne u zadání na tlačítko „Úprava zadání“ v boxu zadání.
2. Systém přesměruje aktéra na formulář, umožňující změnit zadání.
3. Aktér zadání upraví a klikne na „Uložit“.
4. Systém změnu zadání uloží a přesměruje uživatele na stránku se seznamem všech zadání v testové části portálu.

Alternativní scénář 2: uživatel se rozhodne odstranit zadání z otázky. Scénář začíná po bodu č. 3 hlavního scénáře.

1. Aktér klikne u zadání na tlačítko „Odebrat zadání z otázky“ v boxu u zadání.
2. Systém odstraní zadání z otázky a zobrazí aktérovi stránku s otázkou ale už bez odstraněného zadání.
3. Scénář pokračuje v bodě č. 4 hlavního scénáře, popřípadě v bodě č. 8 hlavního scénáře.

Alternativní scénář 3: uživatel se rozhodne zobrazit si již ohodnocené studentské odpovědi k otázce (pokud jsou již nějaké uloženy v databázi). Scénář začíná po bodu č. 3 hlavního scénáře.

1. Pokud v systému jsou nějaké ohodnocené studentské odpovědi, systém společně s údaji o otázce zobrazí zabalený box „Ohodnocené odpovědi“.
2. Aktér box rozbalí.
3. Systém zobrazí tabulku ve které každý řádek představuje jednu studentskou odpověď.



Obrázek 1.3: Stránka s opravou otázky

1.3.2.1 Popis prvků na stránce pro opravu otázky

Jak již bylo zmíněno, stránka je rozdělena ve výchozím zobrazení na levý a pravý sloupec (obrázek 1.3). Toto zobrazení je možné pomocí tlačítka v pravém horním rohu změnit a tyto sloupce se tak zobrazí pod sebou (do tohoto zobrazení se stránka přepne sama, pokud si ji uživatel prohlíží na užším displeji). V této kapitole si detailněji popíšeme jednotlivé prvky na této stránce. Při popisu budeme vycházet z výchozího rozvržení stránky (levý a pravý sloupec).

Levá strana obsahuje tři boxy. Nejvýše umístěný box obsahuje zadání přiřazená k otázce. U zadání je možnost ho upravit nebo přímo odebrat z otázky. Kliknutí na možnost „Upravit zadání“ uživatele přesměruje na stránku /test/assignments/assignment-edit/?assignmentId=x, kde x je ID zadání.

Tato stránka se nachází v jiné části testového modulu, než je část pro manuální opravy, a obsahuje formulář, přes který je možné zadání upravit. Pokud uživatel na stránce s formulářem klikne na tlačítko „Uložit“ nebo na tlačítko „Zpět“, je přesměrován na stránku se seznamem vytvořených zadaní a nedojde tak k přesměrování zpět na stránku s manuální opravou konkrétní otázky.

Po kliknutí na možnost „Odebrat zadání z otázky“ je zadání odebráno z otázky bez jakéhokoliv dalšího potvrzení. Zadání je v tomto případě z otázky odstraněno definitivně a všude (nejen pro ten konkrétní test). Funkce úpravy nebo mazání zadání u otázky slouží jako taková zkratka pro opravujícího uživatele pokud by v zadání narazil na nějakou chybu.

Druhý box v levém sloupci obsahuje tři části: *úkol*, *nastavení* a *referenční odpověď*. Část *úkol* obsahuje znění úkolu, který byl zadán při vytváření otázky. V *nastavení* jsou vypsány informace o dané otázce v rámci testu: *typ*, *obtížnost*, *kategorie*, *úroveň*, *maximum bodů*. Při vytváření otázky byla zadána nějaká referenční (správná) odpověď. Ta je také v této části stránky zobrazena.

Třetí box obsahuje ohodnocené odpovědi a na stránce se zobrazuje pouze pokud už byly nějaké studentské odpovědi na danou otázku v kterémkoliv testu ohodnoceny. Ve výchozím stavu je tento box zabalený. Obsahuje tabulku, kde každý řádek představuje jednu studentskou odpověď. Sloupce tabulky jsou: odpověď studenta, za kolik bodů byla tato odpověď ohodnocena a kdo tuto odpověď opravil.

Na pravé straně stránky jsou boxy s jednotlivými odpověďmi studentů, které je potřeba opravit manuálně. Jedná se o odpovědi studentů, kteří na tuto otázku odpověděli v rámci zvoleného testu. Obsah těchto boxů se liší v závislosti na typu otázky (více ke konkrétním typům v kapitole 1.3.2.2).

U odpovědí nejsou uvedeny žádné informace o jejím autorovi. Opravující hodnotí odpověď pomocí formulářových prvků, které jsou v boxu pod každou otázkou. Zde může udělit za odpověď počet bodů. Portál hlídá možné rozsahy bodů a dovolí uživateli zadat pouze počet bodů v intervalu mezi nulou a maximálním počtem bodů za otázku.

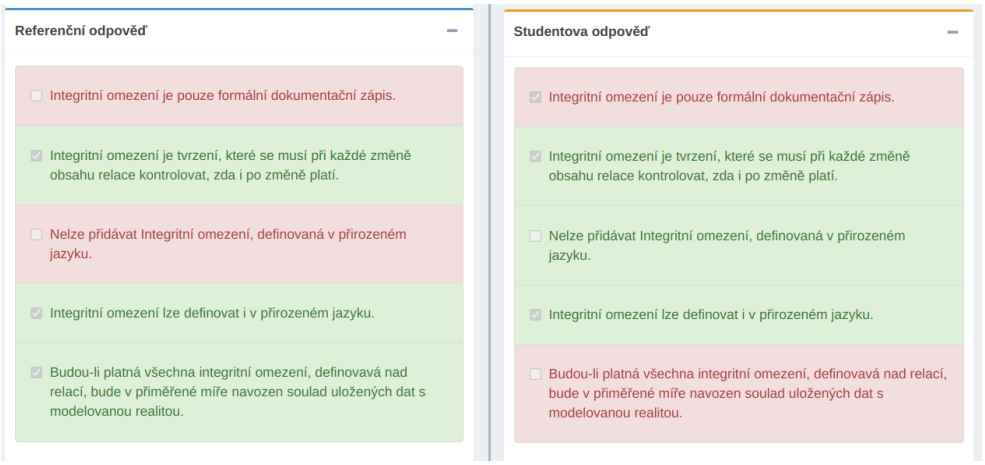
Body se u odpovědí zadávají do formulářového pole buď pomocí klávesnice a nebo klikáním na tlačítka, která jsou umístěna nad tímto polem. Tato tlačítka jsou čtyři. První napíše do pole nula bodů, čtvrté naopak maximální počet bodů, které lze za otázku získat. Mezi těmito tlačítky se nachází dvě další, která umožňují přičítat nebo naopak odečítat ze zadaného hodnocení 0.5 bodu.

Dále má opravující možnost k odpovědi napsat komentář. Poté co je odpověď ohodnocena (kliknutím na tlačítko „Ohodnotit“), je celý box s odpovědí minimalizován. Na stránce ale zůstane a uživatel má možnost hodnocení zpětně upravit.

Portál se snaží vyučujícím opravování co nejvíce usnadnit a v případě stejné odpovědi více studentů (záleží na typu otázky) může opravit více odpovědí zároveň. Po odeslání hodnocení každé odpovědi se zobrazí upozornění s informací, kolik odpovědí bylo naráz opraveno. V tomto případě se všem jednotlivým odpovědím připočte stejný počet bodů a také se k nim přidá stejný komentář. Po opravě všech odpovědí se na stránce zobrazí zelené upozornění s textem „Všechny odpovědi byly ohodnoceny“, který po chvíli zmizí. Boxy s odpověďmi ale může uživatel stále rozbalovat a hodnocení měnit.

V dolní části stránky je na levé straně tlačítko „Zpět“, kterým se lze vrátit na stránku s tabulkou otázek z daného testu (popsanou v kapitole 1.3.1).

V pravém horním rohu stránky se nachází tři tlačítka. Jedno z nich slouží pro seřazení odpovědí. Další dvě tlačítka jsou určena pro již zmíněné přepínání způsobu rozvržení prvků na stránce – dva sloupce vedle sebe nebo pod sebou.



Obrázek 1.4: Referenční a studentská odpověď typu checkbox

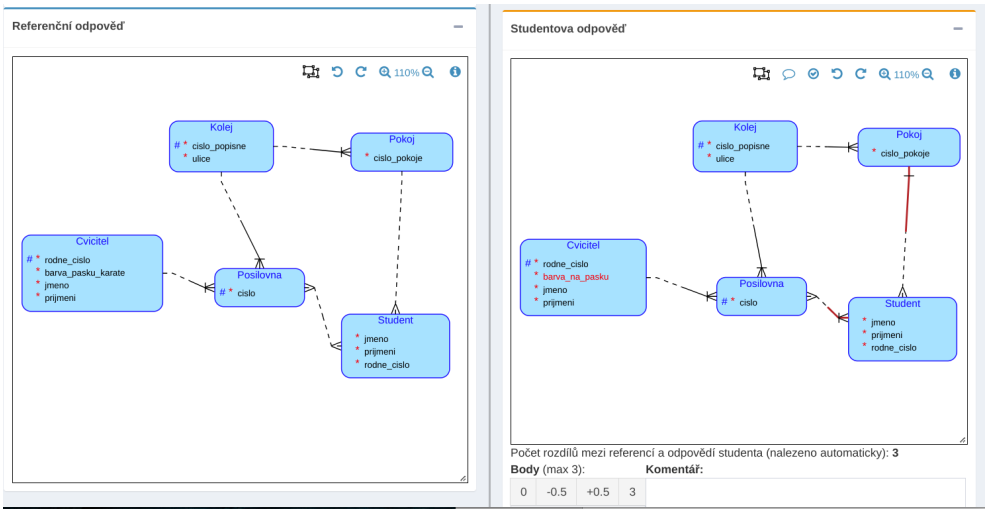
1.3.2.2 Zobrazení odpovědí dle typu

Referenční odpovědi a odpovědi studentů se při jejich zobrazení na stránce s opravou otázky liší podle toho, jakého typu odpověď je. Následující text detailněji popisuje jak jsou zobrazeny referenční a studentské odpovědi. Pod každou studentskou odpovědí (u všech typů) je ještě formulář pro vyučující, kde mohou za odpověď udělit body a okomentovat ji.

U odpovědí typu *text* je celkem jasné, že se odpovědi zobrazují čistě jako pouhý text. Jiné typy jsou ale zobrazeny komplexněji. I přesto, že některé typy odpovědí se opravují plně automaticky, je nutné s nimi při návrhu také počítat. Tyto odpovědi jsou totiž zobrazeny v části portálu, kde vyučující mohou procházet a přepisovat automatické opravy.

Referenční i studentské odpovědi typu *checkbox* a *radio list* se zobrazují ve stejném formátu (obrázek 1.4). Jedná se o seznam možností, kde na pozadí textu za správnými odpověďmi je zelené pozadí, na pozadí textu se špatnými odpověďmi je pozadí červené. U typu *radio list* u studentských odpovědí je tímto způsobem zvýrazněna pouze odpověď vybraná studentem. Studentův výběr je ve studentské odpovědi vyznačen pomocí zaškrtnutého příslušného formulářového prvku (checkbox pro typ odpovědi *checkbox*, radio pro typ odpovědi *radio list*).

V odpovědích typu *diagram* (obr. 1.5) je diagram zobrazen pomocí *Kreslítka*. Diagram v něm ale nejde editovat. V diagramu jsou červeně zvýrazněny rozdíly mezi studentskou odpovědí a referenční odpovědí, které byly nalezeny automaticky. *Kreslítko* u studentských odpovědí je možné přepnout do režimu oprav a pak klikáním na entity nebo atributy lze vyznačovat chyby manuálně. V tomto režimu je také možné psát k jednotlivým prvkům v diagramu komentáře. Ty se píší do javascriptového prvku – vyskakovacího okna *prompt*. Počet automaticky nalezených rozdílů je pak napsán pod diagramem.



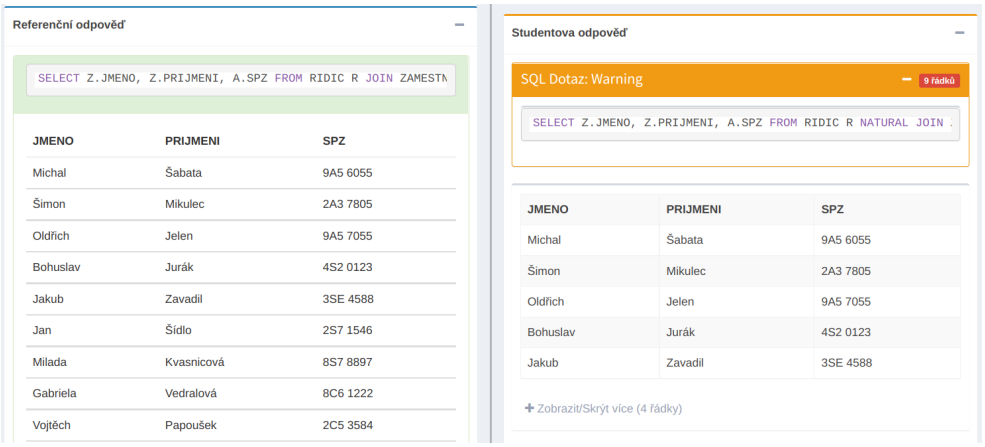
Obrázek 1.5: Referenční a studentská odpověď typu diagram

Systém se snaží zobrazené diagramy uspořádat tak, aby si byly co nejvíce podobné. Opravující pak při porovnávání studenských odpovědí s referenční nemusí diagramy přeskupovat a mapovat k sobě odpovídající entity. Diagramy se automaticky uspořádají nezávisle na tom, jak byly do systému zadány. Toto řešení je popsáno v diplomové práci [8], jejímž autorem je Tomáš Fedor.

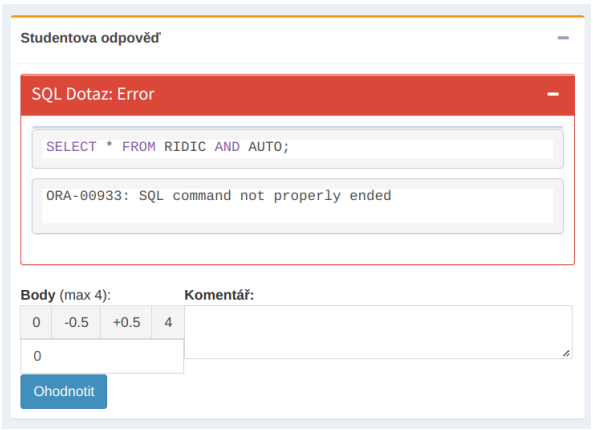
Referenční odpověď typu *SQL* obsahuje samotný SQL dotaz a pod ním tabulku, kde je v řádcích výsledek tohoto dotazu. Studentská odpověď je zobrazena velmi podobně jako referenční. Navíc ještě obsahuje výstup z překladače: success, warning, error (včetně zprávy) a počet vrácených řádků. Pod dotazem je rovněž tabulka s výsledkem tohoto dotazu, ale zobrazuje se pouze prvních pět řádků. Všechny řádky je možné zobrazit kliknutím na položku „Zobrazit/Skrýt více“ pod tabulkou. Na obrázku 1.6 můžeme vidět referenční a studentskou odpověď typu *SQL*, kde výsledek dotazu ve studentské odpovědi neodpovídá té referenční. Obrázek 1.7 pak ukazuje studentskou odpověď, kde se dotaz ani nepodařilo přeložit.

Odpověď typu *RA* se zobrazuje velice podobně jako *SQL*. Referenční odpověď obsahuje dotaz v relační algebře a pod ním jeho výsledky. Ve studentově odpovědi je dotaz, pod ním je výsledek překladu tohoto dotazu do SQL a samotný přeložený SQL dotaz. Dále je zde výsledek, který tento dotaz vrátil s omezením na pět řádků (stejně jako u SQL).

Referenční odpověď u typu *transformace* je text ve formátu zjednodušeného relačního zápisu. V kapitole 1.2.2 bylo u odpovědí tohoto typu zmíněno, že výstupem automatické opravy je JSON, ve kterém jsou zaznamenány rozdíly oproti referenční odpovědi. Předtím je ale nutné studentskou odpověď přetransformovat do standardního tvaru (změní se pořadí prvků atd.).



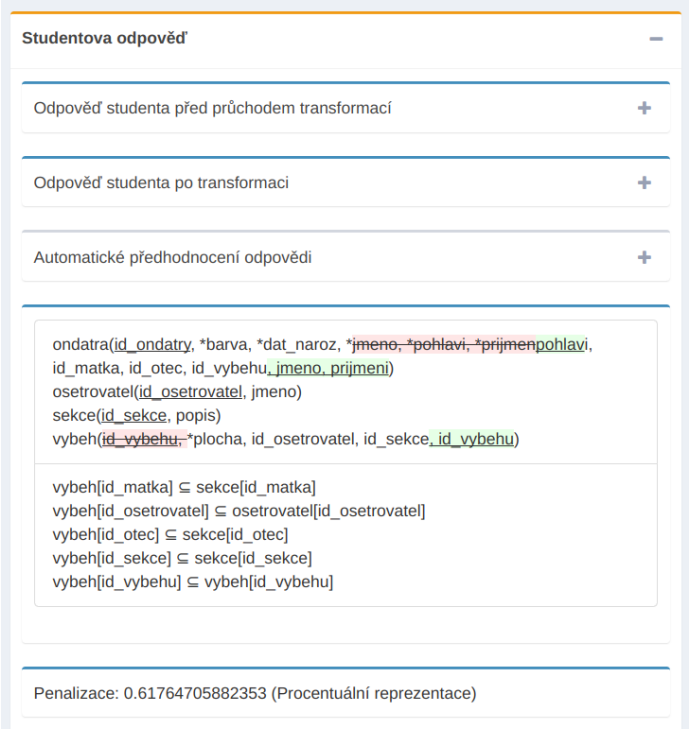
Obrázek 1.6: Referenční a studentská odpověď typu SQL



Obrázek 1.7: Studentská odpověď typu SQL s chybovým výsledkem překladu

Studentské odpovědi jsou tak rozděleny do pěti částí (boxů), ve kterých je vidět výstup této automatické opravy. První část je odpověď přesně v tom tvaru, jak ji student napsal, ve druhé části je přetransformovaná odpověď. Třetí část obsahuje výstup z automatické opravy, tedy předhodnocenou odpověď. Zde jsou vypsány všechny chyby, které automatická oprava nalezla a procentuální penalizace, kterou za ně udělila. Ve čtvrté části jsou pak znázorněny rozdíly mezi přetransformovanou odpovědí studenta a referenční odpovědí a pátá část obsahuje celkovou penalizaci, kterou automatické hodnocení udělilo. Studentská odpověď typu *transformace* je na obrázku 1.8.

U otázek s odpovědí typu *normalizace relačního schématu* se nenachází přímo referenční odpověď, protože žádná ani nejde při vytváření otázky zadat. Místo toho je v levém sloupci mezi boxy „Úkol“ a „Nastavení“ (kapitola 1.3.2.1) box s vygenerovaným ukázkovým řešením. Jak bylo zmíněno



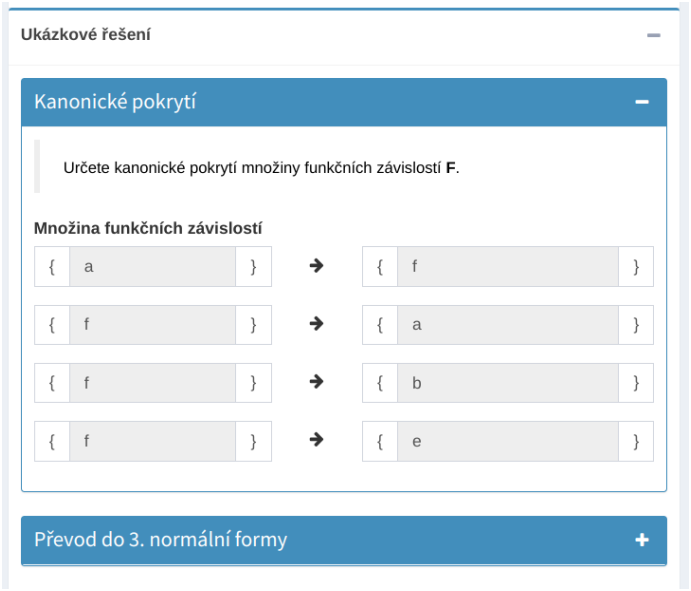
Obrázek 1.8: Studentská odpověď typu transformace

při popisu tohoto typu odpovědi (kapitola 1.2.1.3), otázka se dá rozdělit na podúlohy. Každá podúloha má svůj formulář, který je vidět jak v ukázkovém řešení, tak v odpovědích studentů. Každá odpověď má u každé podúlohy také uvedeno znění úkolu, který se k dané podúloze váže. Ukázkové řešení podúlohy *Kanonické pokrytí* si můžeme prohlédnout na obrázku 1.9

1.3.3 Stránky pro prohlížení automatických oprav

Stránky pro manuální opravy se příliš neliší od stránek, kde mohou vyučující procházet automaticky opravené odpovědi. Proto většina popisu současného stavu stránek pro manuální opravy platí také pro prohlížení automatických oprav. I zde vyučující nejprve vybírá test z něhož chce vidět otázky a následně vybírá samotné otázky. Otázky se mu zobrazují stejným způsobem, jako bylo popsáno dříve v textu. Hodnocení u automaticky opravených odpovědí může vyučující změnit nebo k nim přidat komentář. Tato kapitola popíše jen ty drobné rozdíly, mezi stránkami pro manuální opravy a stránkami pro automatické opravy.

Na tabulku testů, u kterých si chceme prohlédnout automatické opravy se dostaneme kliknutím na položku „Automatické opravy“ v levém menu testového modulu portálu. I zde jsou dvě tabulky: „Moje testy“ a „Všechny testy“.



Obrázek 1.9: Ukázkové řešení odpovědi typu normalizace

Tabulky v řádcích obsahují testy, které se psaly a oproti tabulkám, které jsou popsány u manuálních oprav v kapitole 1.3.1 se liší v tom, že neobsahují sloupce *k opravě*, *přiřazená otázka* a *opraveno*. Navíc obsahují sloupec *automaticky opraveno*, ve kterém je uveden počet automaticky opravených odpovědí v daném testu. Ve sloupci *akce* je pak na každém řádku pouze jedna akce (tlačítko) „Zobrazit“.

Po kliknutí na „Zobrazit“ se zobrazí stránka s automaticky opravenými otázkami z testu. Zde se logicky nenachází sloupec *přiřazený opravující*, ale jsou tu místo toho tři sloupce, ze kterých vyplývá, kolik odpovědí bylo automaticky ohodnoceno. Jedná se o sloupec, ve kterém je počet všech odpovědí, dále je zde uveden počet automaticky opravených odpovědí a ve třetím sloupci je uvedeno, kolik odpovědí bylo automaticky hodnoceno 0 body.

1.4 Nový backend

Jedním z hlavních cílů této práce je navázat na bakalářskou práci Andriiho Plyskache [1], který se zabýval refaktoringem backendu celé testové části DBS portálu. Autor v této práci uvádí, že původní řešení této části portálu začalo být kvůli dalšímu rozšíření a novým požadavkům nedostatečné a tak bylo rozhodnuto, že se vytvoří zcela nový testový modul. Ten by měl nejen dostatečně naplnit aktuální požadavky na systém testování studentů, ale zároveň by měl být také snadno rozšiřitelný do budoucna.

Návrh počítá s rozdělením aplikace na menší samostatné části, které spolu

budou poté komunikovat. Některé části portálu se v nové verzi měnit takřka nebudou, u jiných ale dojde ke změnám. Tyto změny se pak více či méně dotknou také té části testového modulu, které se věnuje tato práce.

Jak bylo popsáno v předchozích kapitolách, celá část týkající se manuálních oprav jde rozdělit na část, kdy si uživatel nejprve vybere test a v něm pak otázku, kterou chce opravovat a na část samotné opravy otázky. Při opravování otázky je nutné zobrazit zadání, samotnou otázku a odpovědi. Do nového návrhu frontendu manuálních oprav je tak nutné zapracovat změny, které v těchto oblastech návrh nového backendu přináší.

Dle návrhu v práci [1] přibude nový typ zadání: *databáze*. Tento typ má sloužit k uchování informací o databázovém připojení. Bude využíván pro vyhodnocení otázek s odpovědí typu *RA* nebo *SQL*. Studentům tento typ zadání nebude nijak zobrazován. Dále přibude v testovém modulu možnost přidávat a spravovat připojení. Tato připojení se budou přiřazovat právě k zadání typu databáze a budou sdíleny pro všechny vyučující.

U otázek dojde ke zrušení kategorií a úrovní. Místo toho jim budou přiřazovány tzv. štítky. Ty budou sloužit pro rozdělování otázek do kategorií dle libovolného klíče. Vytvářet a spravovat je bude moci pouze garant. Jedna otázka může mít více štítků a štítek může označovat více otázek. Jejich využití bude především u automaticky generovaných (dynamických) testů. Ze stejného důvodu bude možné otázky také řadit do různých skupin.

Nový návrh také řeší problém, který je u šablon v aktuální verzi. Tam totiž vždy, když je ze šablony vytvořen nějaký test, šablonu už poté není možné modifikovat. Návrh počítá s tím, že testy budou na šabloně nezávislé a nastavení šablony bude tak možné změnit i poté, co podle ní byl vytvořen test. Vytvořené testy bude možné přidávat do skupin a přibude tím podpora více variant jednoho testu, které pak mohou být spuštěny v rámci jednoho termínu.

Studentské a referenční odpovědi budou v databázi sjednoceny do jedné tabulky. V bakalářské práci Andriiho Plyskache je dále uvedeno, že „*vyučující bude schopen ze studentské odpovědi udělat referenční v průběhu manuální opravy*“ [1]. Tuto možnost bude tedy nutné také zapracovat do návrhu frontendu manuálních oprav v této práci.

1.5 Požadavky

Před začátkem vývoje jakéhokoliv většího softwarového projektu je nutné správně specifikovat, jak přesně má výsledný systém vypadat, co by mělo být implementováno a jaké by měl mít systém vlastnosti. Správná analýza těchto požadavků je při vývoji softwaru klíčová, protože zefektivní pozdější práci a zabrání řadě nedorozumění mezi vývojáři, zadavateli a koncovými uživateli. Je třeba podchytit, co přesně zákazník od nového produktu očekává.

Musí se počítat s tím, že ani samotný zákazník ze začátku nedokaže všechny své požadavky správně specifikovat.

Protože se bude celý frontend manuálních oprav testové části portálu vyvíjet znovu, bylo by vhodné, vedle zachování stavajících funkcionalit, nové řešení rozšířit a vylepšit. Některé tyto možnosti úprav jasně vyplývají už z předchozí analýzy současného stavu. Jelikož ale nemáme žádnou zkušenost s touto částí portálu během výuky, kdy mnoho studentů generuje mnoho rozličných odpovědí na otázky v testech, je naše současná představa o tom, jak současné řešení rozšířit značně omezená.

Z tohoto důvodu je potřeba se ujistit, že chápeme správně, jak současné řešení jeho stálí uživatelé používají a zjistit jejich pohled na případná rozšíření a především jaká rozšíření by oni nejvíce uvítali. To zjistíme pomocí výzkumu přímo mezi uživateli. V této kapitole si nejprve projdeme jaké strategie výzkumů vůbec jsou a jaké máme při získávání požadavků možnosti. Nakonec si nějakou možnost zvolíme, průzkum provedeme a zpracujeme jeho výsledky.

1.5.1 Strategie výzkumů

Při provádění výzkumu rozlišujeme dvě hlavní strategie, pomocí kterých můžeme tento výzkum provádět – kvantitativní a kvalitativní. Každý z těchto přístupů má své klady i zápory. Někdy může být vhodnější zkoumat kvantitativně a někdy zase kvalitativně. Velice záleží na tom, co zkoumáme a co je naším cílem.

1.5.1.1 Kvantitativní výzkum

Kvantitativní strategie výzkumu přistupuje ke zkoumaným prvkům tak, že se je snaží nějakým způsobem měřit nebo třídit. Zaměřuje se na to, aby se získané informace daly jednoduše uspořádat a porovnávat pomocí různých formálních metod.

Postup je u takového výzkumu předem naplánován. Počítá se s tím, že se získá větší množství dat, která se pak ve velice zobecněné formě zpracují. Výhodou této strategie je, že data dokážeme získat a zpracovat poměrně rychle, ale závěry tohoto výzkumu mohou být docela abstraktní. [9]

1.5.1.2 Kvalitativní výzkum

Kvalitativní strategie je opakem té kvantitativní. Nepočítá se zde tak s tím, že se data získají nějakým standardizovaným postupem. Tento přístup se více se zaměřuje na konkrétní skutečnosti, které zkoumá individuálně a více do hloubky. Data je možné vyhodnocovat ještě během jejich sběru a výstupům poté přizpůsobovat další výzkum.

Tento přístup se hodí v situacích, kdy potřebujeme zkoumané skutečnosti prozkoumat detailně a nehodí se nám příliš velká míra zobecnění. Zkou-

mání pomocí kvalitativní strategie je ale oproti kvantitativní časově náročnější a klade větší nároky na člověka, který výzkum provádí. [9]

1.5.2 Techniky získávání požadavků

Existuje celá řada různých technik jakými lze získat požadavky na systém. V podstatě každou techniku lze využít jak při kvalitativním, tak kvantitativním výzkumu, ale některé techniky mohou být pro některou ze strategií výhodnější. Při získávání požadavků se pracuje s takzvanými „stakeholders“, což jsou strany, které jsou v projektu nějakým způsobem zainteresovány. Mohou to být například vývojáři, zadavatelé projektu a nebo koncoví uživatelé. [10]

Pokud je cílem projektu rozšíření nebo předělání nějakého existujícího systému, tak by se za jednu z těchto technik by se dala určitě označit také analýza existujícího systému a současného řešení [10]. To je případ i této práce a tato analýza byla provedena v kapitole 1.3. V této podkapitole si popíšeme některé další techniky, které je možné k získání požadavků využít.

1.5.2.1 Rozhovor

Rozhovor je možné využít jak při kvantitativním i kvalitativním přístupu. Při kvantitativním výzkumu se obvykle tvoří otázky, které jsou přesně naformulovány a ve většině případů jsou tyto otázky uzavřené (respondent má na výběr jen jistou množinu odpovědí). V těchto rozhovorech se vždy postupuje přesně podle nějakého předem určeného harmonogramu.

Při využití kvalitativní strategie můžeme rozlišit tyto typy rozhovorů: volný, polostrukturovaný a strukturovaný. U volného rozhovoru nejsou předem připraveny otázky. Ty vznikají až během rozhovoru, který je veden neformálním způsobem. Polostrukturovaný rozhovor má již předem připraveny otázky, nebo alespoň okruhy, které je potřeba při rozhovoru probrat. Je ale možné tuto přípravu během konání rozhovoru přizpůsobovat nebo otázky řešit v jiném pořadí.

Strukturovaný rozhovor má otázky i jejich pořadí předem dané. Otázky jsou otevřené, ale je nutné hlídat, aby respondent příliš neodbíhal od tématu. Rozhovor v této formě ztrácí jistou přirozenost a různorodost respondentů je potlačena. Výhodou strukturovaných rozhovorů je ale to, že se výstupy z nich snadněji zpracovávají a vyhodnocují. [9]

Analytik se musí před samotným rozhovorem dobře připravit. Pokud se nejedná o volný rozhovor, musí si ujasnit všechny okruhy a případně konkrétní otázky, které potřebuje s respondentem probrat. Rozhovorů je vhodné provést více s více respondenty, protože požadavky jednotlivých uživatelů se mohou lišit, nebo si dokonce odporovat.

Pro snazší analýzu výsledků je možné si rozhovor nahrávat. Je důležité aby respondent věděl, proč se rozhovor koná a jaké jsou jeho cíle. Během rozhovoru musí analytik aktivně naslouchat a reagovat na to co respondent

říká, pokládat doplňující otázky a předkládat respondentovi vlastní nápady ke zhodnocení. [10]

1.5.2.2 Workshop

V případě získávání požadavků pomocí workshopů se nejprve pečlivě vybere skupina účastníků výzkumu tak, aby v ní byli zastoupeni různé typy lidí, kteří jsou do projektu zainteresováni (uživatelé, vývojáři, testeři...). Tato skupina se pak sejde a společnou prací vytváří nové požadavky. Tímto způsobem je možné zjistit co přesně jednotlivé skupiny od systému očekávají tak, aby tomu ostatní rozuměli a zároveň se tím vyjasní případné požadavky, které si navzájem odporují. Tato technika je poměrně náročná na organizaci. Musí se také dobře odhadnout počet a složení zúčastněných. [10]

1.5.2.3 Pozorování

Pokud se při zjišťování požadavků omezíme pouze na nějakou formu rozhovoru s uživatelem (či více uživateli) nemusíme zdaleka odhalit vše, co by měl nový systém umět. Uživatel si na řadu požadavků nemusí vůbec vzpomenout, nebo je nemusí umět srozumitelně popsat. Pokud má vytvářený produkt zajišťovat nějakou agendu, která již v organizaci probíhá (např. pomocí starší verze systému), může být lepší nechat uživatele se systémem pracovat a při této práci ho pozorovat.

Těchto uživatelů může být více a je velice důležité rozlišit, kdy se zjištěný požadavek týká jednotlivců a kdy celých skupin. Tato pozorování mohou i nemusí být interaktivní. To znamená, že je možné je provádět tak, že uživateli při plnění úkolů pokládáme otázky, ale můžeme ho i pouze pozorovat. Tato pozorování je možné pro další vyhodnoceni také nahrávat. [10]

1.5.2.4 Dotazník

Pokud chceme získat velké množství jednoznačně strukturovaných dat od většího počtu uživatelů, můžeme využít dotazník. Uživatelům se pomocí něj položí různé otázky. Ty musí být správně formulovány tak, aby byly srozumitelné a aby účastníka výzkumu nenaváděli k volbě nějaké odpovědi. Tato data je poté poměrně jednoduché zpracovat a výstup je pak možné využít také k analýze pomocí jiné techniky (např. workshop). [10]

1.5.3 Volba strategie a techniky

Koncovými uživateli našeho systému jsou vyučující předmětu BI–DBS. Ti již mají také praktické zkušenosti s manuální opravou testů v současném testovém modulu. Specifikace požadavků a rozšíření na nový frontend se tak neobejde bez informací právě od nich.

Počet účastníků výzkumu je tedy shora omezen počtem cvičících. Vzhledem k této skutečnosti a k povaze celého projektu tak bude provedena analýza za využití kvalitativní strategie. Na konzultaci s vedoucím práce bylo rozhodnuto, že bude pro získání požadavků využita technika rozhovoru. Bude se jednat o polostrukturovaný rozhovor.

Tato forma byla zvolena z důvodu, že je nutné výsledky rozhovorů, které budou vedeny s více cvičícími, snadno zpracovat a shrnout. Zároveň je žádoucí poskytnout cvičícím při rozhovoru jistou volnost, aby nebyli strukturou rozhovoru omezeni až příliš. Cílem je totiž získat i ty požadavky, o kterých zatím vůbec nevíme a v plně strukturovaném rozhovoru bychom se k nim nemuseli dostat.

Celý proces toho jak probíhala příprava na rozhovory, průběh samotných rozhovorů a především jejich výsledek je popsán v následujícím textu.

1.5.4 Příprava rozhovorů

Z podmnožiny cvičících předmětu BI–DBS, kterou určil vedoucí práce bylo náhodně vybráno pět potencionálních respondentů, kteří byli pomocí e-mailu osloveni se žádostí o rozhovor. Byl jim sdělen cíl rozhovoru a cíl celé této práce. Rozhovor byl připravován tak, aby jej bylo možné uskutečnit jak ve formě osobní konzultace, tak pomocí videohovoru se sdílenou obrazovkou.

Cílem rozhovorů bylo zjistit pohled vyučujících na aktuální stav manuálních oprav, co vyučujícím při opravě testů vadí, co jim tam chybí a tak podobně. Předtím, než mohl začít první rozhovor, musela proběhnout příprava otázek. Dále bylo záměrem při rozhovoru respondentům zobrazovat současné řešení frontendu manuálních oprav v DBS portálu. Z tohoto důvodu bylo nutné připravit před zahájením rozhovorů vzorová data tak, aby v systému existoval test, ve kterém jsou otázky se všemi typy odpovědí a na každou otázku odpovědělo několik studentů různými způsoby (správně, částečně správně, špatně).

1.5.5 Průběh rozhovorů

Z pěti oslovených cvičících s rozhovorem souhlasili všichni. Konkrétně se jednalo o tyto cvičící: Ing. Šimon Schierreich, Ing. David Šenkýř, Ing. Cyril Černý, Ing. Jan Matoušek a Ing. Jan Blizničenko. Se třemi se rozhovor uskutečnil online pomocí platformy *Microsoft Teams*, další dva respondeti rozhovor poskytli osobně ve své kanceláři na fakultě. Na začátku rozhovoru všichni cvičící souhlasili s tím, že rozhovor bude nahráván. Pořízené nahrávky jsou určeny pouze pro potřeby snažšího zpracování a se cvičícimi bylo domluveno, že nebudou nikde publikovány. Nahrávky byly pořízeny pomocí programu *OBS Studio*. Během rozhovorů byl použit software *Vagrant*, ve kterém byla aktuální verze portálu lokálně spuštěna.

Doba trvání rozhovorů se pohybovala přibližně mezi 20 až 35 minutami. Na začátku rozhovoru byl vždy responentovi zopakován účel rozhovoru a před-

staveny cíle práce. Každý rozhovor měl trochu jiný průběh, ale postupně se prošly všechny připravené okruhy a byly pokládány (v různých obměnách) předem připravené otázky.

Probírané otázky lze rozdělit do těchto okruhů: celková spokojenost s aktuálním stavem a role respondenta v portálu, procházení testů a otázek k opravě, stránka pro hodnocení otázek, hodnocení otázek dle typu odpovědí a automatické opravy. V případě potřeby byly respondentům položeny také doplňující otázky.

Občas se během rozhovoru stalo, že respondent sám navrhl nějakou úpravu nebo vylepšení aktuálního stavu. Zjištění nových požadavků bylo jedním z hlavních cílů rozhovorů a proto byl s respondentem tento návrh vždy detailněji rozebrán.

1.5.6 Výsledek rozhovorů

Všichni cvičící, se kterými byl rozhovor proveden uvedli, že jsou s aktuálním stavem spíše spokojeni. Nejvíce bylo vyzdvihováno především to, jak se portál snaží opravu zautomatizovat a celkově co nejvíce usnadnit. I přesto ale rozhovory poukázaly na několik nedostatků, které aktuální verze má.

Jak již bylo zmíněno, cvičící během rozhovorů sami přicházeli s návrhy a tématy a tak byly odhaleny některé nové důležité aspekty opravování testů, které jsou pro tuto práci dále důležité. V následujících podkapitolách se postupně na jednotlivé aspekty detailněji podíváme. Bude zde uvedeno to, co si cvičící myslí o aktuálním řešení, jak ho používají a co z toho celkově vyplývá pro návrh nového frontendu.

1.5.6.1 Role cvičících v testovém modulu

Všichni cvičící uvedli, že mají nějakou svou sadu otázek, kterou vytvořili a mají na starosti její opravu. Dříve jednotliví cvičící vytvářeli také testy, ale v posledních semestrech se tato činnost více zcentralizovala. Bylo zjištěno, že někteří cvičící nemají zkušenost se všemi typy odpovědí.

1.5.6.2 Přístup k otázkám

Respondenti byli dotázáni, jakým způsobem se dostávají na stránku, na které probíhá hodnocení studentských odpovědí. Čtyři z pěti cvičících uvedli, že k jednotlivým otázkám přistupují kliknutím na položku „Manuální opravy“ v levém menu portálu. Pouze jeden cvičící přistupuje k otázkám pomocí notifikací, které jsou zobrazeny v horní liště portálu.

V tabulkách „Moje testy“ a „Všechny testy“ cvičící nejvíce zajímá sloupec *akce* a také informace, kolik otázek, jim zbývá v daném testu k opravě. Po kliknutí na jednu z akcí se uživateli zobrazí stránka se všemi otázkami k danému testu. Zde cvičící nejčastěji hledají informaci, které otázky opravují oni (sloupec *přiřazený opravující*). Dalším důležitým údajem je také *typ odpovědi*.

Celkově nebylo během žádného rozhovoru zjištěno, že by cvičící měli nějaký problém s přístupem k hodnocení jednotlivých otázek, nebo že by jim zde něco chybělo. V této části tak není potřeba provádět žádné zásadní změny.

1.5.6.3 Typ zobrazení

Dále byla s cvičícími probírána stránka, kde dochází k samotnému hodnocení odpovědí. Co se týče zobrazení, tak všichni cvičící uvedli, že používají výhradně zobrazení, kdy je stránka rozdělena na dva sloupce, kde vlevo je zadání a další údaje o otázce a vpravo jsou studentské odpovědi. Možnost zobrazit si nejprve zadání a pod ním všechny odpovědi označili cvičící za nepraktickou, protože během opravy většinou potřebují údaje o otázce vidět.

Žádný z dotazovaných cvičících neprovádí manuální opravu na mobilním zařízení. Všichni zároveň uvedli, že ani v případě, že by bylo rozhraní stránky na menším displeji přívětivější, by na mobilním zařízení spíše neopravovali. Při vývoji nové verze uživatelského rozhraní tak zobrazení na mobilním zařízení nebude prioritou.

Rozvržení, kdy je stránka rozdělena do dvou sloupců hodnotili cvičící pozitivně. Dva cvičící uvedli, že využívají možnost si pomocí dělící čáry zmenšit levou stranu. V nové verzi by se tak dalo inspirovat tímto typem zobrazení.

1.5.6.4 Zadání a údaje o otázce

Nyní si rozebereme pohled cvičících na zobrazování zadání a údajů o otázce, kterou právě opravují. Ve výchozím zobrazení se tedy jedná o levý sloupec. Mezi tyto údaje patří úkol, nastavení, referenční odpověď a ohodnocené odpovědi. Jeden respondent uvedl, že se na levou stranu při opravě většinou moc nedívá. Ostatní cvičící naopak sdělili, že je pro ně důležité při opravě vidět především zadání a referenční odpověď. Zároveň ocenili možnost si tyto boxy minimalizovat, což umožňuje již aktuální verze manuálních oprav.

Možnost editovat zadání přímo ze stránky pro hodnocení manuálních oprav cvičící nevyužívají téměř nikdy. Z rozhovorů vyplynulo, že chyby v této fázi testu již v zadání spíše nenacházejí a možnost úpravy nepřekáží, ale není nutná. Možnost odstranit zadání z tohoto místa pak označili za úplně zbytečnou.

Box, který obsahuje nastavení otázky, označili za zbytečný všichni cvičící. Všechny pro opravu důležité údaje (typ odpovědi, maximum bodů) jsou zřetelné i z jiných částí stránky a celkově tento box při opravě není potřeba.

Na box, který zobrazuje dříve ohodnocené odpovědi studentů s přidělenými body napříč různými testy, cvičící měli rozdílné pohledy. Jeden cvičící tento box nevyužívá vůbec, další uvedl, že mu to při opravě příliš nepomáhá. Ostatní tři cvičící o zobrazení dříve ohodnocených odpovědí řekli, že je občas užitečné a celkově jsou spokojeni s jeho aktuálním stavem. Cvičící však možnosti nahlížet do nich využívají spíše v menším množství případů.

1.5.6.5 Hodnocení odpovědí

Způsob hodnocení odpovědí se liší dle typu odpovědi. Jednotlivým odpovědím se bude věnovat podkapitola 1.5.6.7. Všechny odpovědi, nezávisle na typu, mají ale při hodnocení společný způsob, jakým se zadávají komentáře a body. Také tyto dva prvky byly předmětem rozhovorů.

Bylo zjištěno, že většina dotazovaných se snaží psát komentáře k odpovědím poměrně často. Tři cvičící uvedli, že při opravě používají nějaký program (poznámkový blok), kam si zapisují komentáře k typickým chybám a kolik bodů za danou chybu strhli. Komentáře poté k jednotlivým odpovědím pouze kopírují. V návrhu by proto bylo dobré nalézt způsob, jak by cvičícím tuto funkci mohl poskytnout přímo portál.

Aktuální podobu formuláře na zadávání bodů hodnotili všichni cvičící pozitivně. Dle situace využívají všechny způsoby, jakými lze body zadat (pomocí klávesnice i klikáním na tlačítka). Celkově se tak na tomto formuláři nemusí nic výrazného měnit.

1.5.6.6 Zpětné procházení ohodnocených odpovědí

Čtyři z pěti dotazovaných uvedli, že se v průběhu opravování otázky potřebují poměrně často vracet k již ohodnoceným odpovědím. Stává se totiž, že během opravy narazí na studentskou odpověď, ve které se nachází podobná chyba, která se vyskytla někde předtím. Cvičící si chtějí připomenout jak danou odpověď ohodnotili, případně hodnocení zpětně změnit.

Tomuto zpětnému procházení odpovědí ale portál příliš nepomáhá. Ohodnocená odpověď se minimalizuje a aby ji cvičící viděl, musí ji znovu rozbalit. Když je pak opravených odpovědí mnoho, toto zpětné dohledávání se značně komplikuje, protože cvičící musí rozbalit více odpovědí, než naleznou tu, kterou hledají. Někteří cvičící uvedli, že z tohoto důvodu si otevřou zdrojový kód stránky a vyhledávají dříve ohodnocenou odpověď v něm. Bude tedy nutné vymyslet řešení, které umožní opravené odpovědi zpětně procházet.

1.5.6.7 Typy odpovědí

Během rozhovorů byly procházeny všechny typy odpovědí. Ne všichni cvičící mají zkušenost s opravou všech typů. Pokud ale cvičící daný typ opravuje, byl dotázán, jaký má na opravu daného typu názor a jak by ho vylepšil.

U typů *checkbox* a *text* panovala shoda, že aktuální způsob hodnocení je v pořádku a není třeba zde nic moc měnit. Někteří cvičící opravují odpovědi typu *checkbox* stylem, že mají nějaký vzorec nebo mechanismus, podle kterého udělují body na základě počtu správně nebo špatně zaškrtnutých možností. Může ale nastat případ, kdy student zaškrtne dvě možnosti, které si navzájem odporují a dokáže tak spíše, že se odpověď snažil uhodnout. Z tohoto důvodu tak není možné odpovědi typu *checkbox* plně automatizovat. Nabízí se ale možnost alespoň podle nějakého vzorce body opravujícímu navrhnout.

U odpovědí typu *diagram* cvičící zmiňovali, že jim občas trochu vadí, že je diagram v odpovědi příliš velký a nevejde se celý do příslušného okna. Zvlášť u tohoto typu se pak projevuje problém, kdy je obtížné zobrazit referenční odpověď vedle té studentské tak, aby byly na obrazovce vedle sebe a nemuselo se při porovnávání skrolovat.

Typy *SQL* a *RA* jsou podobné a v rozhovorech se řešily společně. Určitě nejdůležitější zjištění, které rozhovory přinesly bylo, že by cvičící u těchto typů využili možnost si nad stejným databázovým připojením, jako je daná otázka, provést vlastní dotaz. Tato možnost by se hodila v situaci, kdy cvičící potřebuje trochu upravit dotaz ze studentské odpovědi, aby si ověřil, zda při opravě identifikoval všechny chyby.

Bylo zjištěno, že cvičící si během oprav typu *SQL* a *RA* obvykle potřebují zobrazit všechny řádky ve studentské odpovědi a současný způsob zobrazení jim vyhovuje. Jeden cvičící také uvedl, že by ocenil možnost si řetězec se studentovou odpovědí naformátovat (rozepsání na více řádků a odsazení).

Názory na současné řešení manuálních oprav odpovědí typu *transformace* jsou rozporuplné. Ukázalo se, že současný způsob zobrazení (diff mezi studentovou transformovanou odpovědí a referenční odpovědí) není moc intuitivní a člověk si na něj musí zvyknout. Do jisté míry se s ním ale poté dá pracovat. Cvičící uváděli, že si často rozbalují box s původní studentskou odpovědí.

1.5.6.8 Automatické opravy

Se stránkami, které se týkají automatických oprav, cvičící nepřicházejí skoro vůbec do kontaktu. Využití tato stránka má jen výjimečně, ale je užitečná například v případě, že někdo omylem označí špatnou odpověď jako správnou a pak ji portál používá během automatických oprav.

1.5.7 Shrnutí

1.5.7.1 Funkční požadavky

F1: Zobrazení testů k opravě

Systém zobrazí tabulku s testy, s informací kolik odpovědí je potřeba opravit a zda je nějaká otázka přiřazena k opravě právě přihlášenému uživateli. Tento požadavek plní již současný systém. V nové verzi je potřeba zachovat alespoň současnou formu.

Priorita: vysoká
Složitost: nízká

F2: Zobrazení přehledu otázek z testu k opravě

Systém po výběru testu zobrazí otázky, které byly v rámci tohoto testu zadány. Zároveň u těchto otázek bude uvedena informace, kolik odpovědí v nich zbývá opravit a kdo je přiřazen jako opravující.

Priorita: vysoká
Složitost: nízká

F3: Zobrazení otázky a odpovědí k opravě

Systém po výběru otázky k opravě zobrazí zadání, úkol a referenční odpověď. Dále zobrazí také opravené i neopravené odpovědi na danou otázku v rámci vybraného testu. Systém nabídne uživateli způsob zobrazení při kterém bude moci opravovat otázky a zároveň stále nahlížet do zadání, úkolu a referenční odpovědi. Také mu umožní tyto položky snadno porovnávat s opravovanými odpověďmi studentů.

Priorita: vysoká
Složitost: střední

F4: Usnadnění hodnocení při výskytu stejných chyb

Systém bude uživateli při hodnocení nabízet komentáře, které napsal dříve. Zároveň mu bude na základě těchto komentářů předkládat návrhy na bodové hodnocení.

Priorita: vysoká
Složitost: vysoká

F5: Zpětné procházení již opravených odpovědí

Systém umožní snadno vyhledávat v obsahu již opravených odpovědí. Uživatel zadá do vyhledávače text a systém mu zobrazí odpovědi, které daný text nějakým způsobem obsahují.

Priorita: střední
Složitost: střední

F6: Možnost úpravy a provedení SQL a RA dotazů

Systém na stránce pro opravu otázky poskytne editor, do kterého půjde psát SQL a RA. Takto napsané dotazy pak bude moci systém provést nad databázovým připojením, které se váže k opravované otázce. Systém dále uživateli co nejvíce usnadní možnost do tohoto editoru zkopírovat dotazy ze studentských odpovědí.

Priorita: vysoká
Složitost: střední

1.5.7.2 Nefunkční požadavky

NF1: Propojení s backendem

Backend pro manuální opravu v současnosti není k dispozici. Při vytváření nového frontendu se musí vycházet z jeho návrhu. Až bude backend realizován, musí jít s nově vzniklým frontendem propojit tak, aby do frontendu nebylo nutné příliš zasahovat.

Priorita: vysoká
Složitost: střední

NF2: Použité technologie

Při vývoji frontendu nových součástí se v DBS portálu používá Vue.js. Z důvodu jednotného přístupu a snadného sdílení a znovupoužití některých komponent bude tento framework bude použit také při vývoji frontendu pro manuální opravy.

Priorita: vysoká
Složitost: střední

Kapitola 2

Návrh

V předchozí kapitole jsme se nejprve seznámili s aktuálním stavem frontendu manuálních oprav testového modulu a poté jsme provedli analýzu požadavků a rozšíření. Na základě této analýzy nyní vytvoříme návrh nového frontendu.

Výstup z této kapitoly nám umožní začít s implementací. Některé části návrhu nového frontendu manuálních oprav budou, kromě textového popisu, pro lepší představu doplněny wireframy. Wireframe (drátěný model) je ilustrace rozhraní stránky, která se zaměřuje na rozmístění prvků na stránce, dostupné funkce a zamýšlené chování. Obvykle však neřeší grafiku, barvy, ani styly. [11]

2.1 Přehled testů a otázek

V této části si popíšeme vzhled dvou stránek, které budou sloužit pro výběr testů a otázek, které chce uživatel opravit.

Vhledem k tomu, že cvičící v rozhovorech nevyjádřili nespokojenost s aktuálním stavem těchto stránek, budou stránky téměř stejné, jako v současné verzi. Cvičící na těchto stránkách nebudou trávit mnoho času. Jejich jediným účelem je, aby se uživatelé snadno zorientovali nejprve v tabulce s testy a posléze také v tabulce s otázkami a rychle se dostali na stránku pro hodnocení studentských odpovědí k vybrané otázce.

2.1.1 Manuální opravy

Po kliknutí na položku „Manuální opravy“ v levém menu nového testového modulu se uživateli zobrazí tabulka s testy. Tabulka bude pouze jedna a cvičící si bude moci najít své testy pomocí filtrů, které budou obsahovat sloupce tabulky. Tabulka bude obsahovat tyto sloupce: *název*, *čas spuštění*, *čas ukončení*, *stav*, *autor šablony*, *autor testu*, *zbývá opravit pro přihlášeného uživatele*, *zbývá opravit celkem* a *akce*. Ve sloupci *akce* budou dvě tlačítka: *zobrazit neopravené* a *zobrazit vše*. Ty se budou chovat stejně, jako v současné verzi portálu.

Po zvolení jedné z akcí se uživateli zobrazí stránka s další tabulkou. Nad tabulkou bude v levém horním rohu umístěn název testu a v pravém tlačítko „Zpět“, které vrátí uživatele na stránku s testy. V tabulce budou vypsány jednotlivé otázky k vybranému testu a bude obsahovat tyto sloupce: *název*, *typ odpovědi*, *autor*, *přiřazený opravující*, *zbývá opravit* a *akce*.

Oproti současné verzi nebude mít tabulka sloupce *obtížnost* a *kategorie*, protože nový backend bude otázky kategorizovat jiným způsobem. Zároveň bylo během rozhovorů zjištěno, že cvičící na této stránce zajímají výhradně sloupce *typ odpovědi* a *přiřazený opravující*. Stejně, jako v současné verzi, i zde bude ve sloupci *přiřazený opravující* zvýrazněno jméno, které odpovídá právě přihlášenému uživateli. To pak pomůže cvičícímu snáze najít otázku, kterou má opravit.

Nově zde přibude sloupec *zbývá opravit*, ve kterém bude uveden počet unikátních odpovědí, které čekají na manuální opravu. Po kliknutí na tlačítko „ohodnotit otázku“ ve sloupci *akce* se pak uživateli zobrazí stránka s hodnocením vybrané otázky. Návrh této stránky popisuje kapitola 2.2.

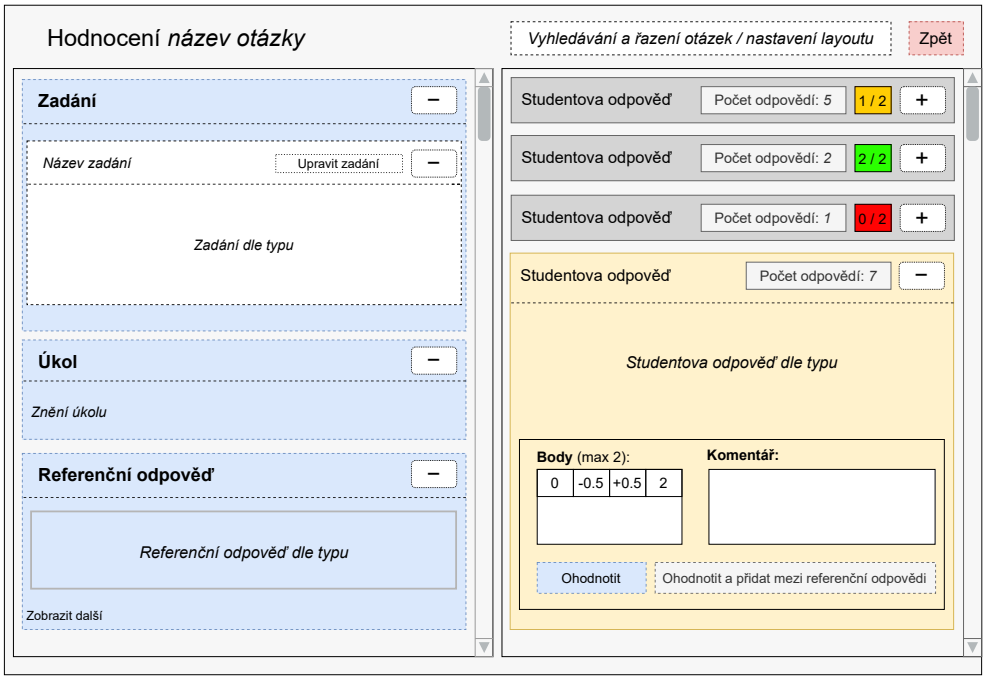
2.1.2 Automatické opravy

Procházení automatických oprav bude řešeno podobně, jako opravy manuální. Rozdíl bude pouze v některých sloupcích tabulek na obou stránkách. Tabulka s testy nebude obsahovat sloupce udávající počet odpovědí, které zbývá opravit. Místo nich bude mít sloupec *automaticky opraveno*, kde bude uveden počet odpovědí, které se v daném testu opravily automaticky. Akce v této tabulce bude jen jedna a kliknutí na ní zavede uživatele k otázkám z příslušného testu.

Tabulka s otázkami nebude, na rozdíl od té v manuálních opravách, obsahovat sloupce *přiřazený opravující* a *zbývá opravit*. Přibudou v ní ale sloupce *všechny odpovědi*, *automaticky opravené odpovědi* a *automaticky hodnoceno za 0 bodů*. Sloupce s těmito informacemi obsahuje také tabulka v současném řešení automatických oprav. Tabulka bude mít jednu akci, která uživatele přenese na stránku s hodnocením vybrané otázky, kde si bude moci automaticky opravené odpovědi procházet a případně měnit jejich hodnocení. Stránka s hodnocením otázky bude stejná, jako v případě manuálních oprav a je popsána v následující kapitole.

2.2 Hodnocení odpovědí

Nové uživatelské rozhraní stránky pro hodnocení studentských odpovědí dozná oproti současnému stavu několika změn. Protože ale během rozhovorů (kapitola 1.5.6) hodnotili cvičící současný stav spíše pozitivně, nebude nutné dělat příliš velké změny a uspořádání prvků na stránce v nové verzi bude současnou verzí částečně inspirováno.



Obrázek 2.1: Výchozí rozvržení stránky pro hodnocení odpovědí

2.2.1 Rozvržení stránky

Stránka s hodnocením odpovědí bude mít dva režimy rozvržení. Ve výchozím zobrazení bude stránka vertikálně rozdělena na poloviny. Toto rozvržení se na první pohled podobá výchozímu rozvržení stránky v současné verzi manuálních oprav (dva sloupce). Předpokládá se, že ho bude využívat podstatná většina vyučujících, protože během rozhovorů bylo zjištěno, že všichni dotazovaní vyučující využívají v současné verzi právě rozdělení stránky na dva sloupce.

Návrh výchozího rozvržení stránky pro hodnocení odpovědí je zobrazen na obrázku 2.1. Obrázek ukazuje pouze obecné rozložení jednotlivých prvků. Konkrétní obsah se bude lišit dle typu odpovědi a jeho návrh bude popsán dále v textu.

Levá strana se bude skládat z těchto boxů: *Zadání*, *Úkol* a *Referenční odpověď*. Oproti starému řešení, návrh neobsahuje box *Nastavení*, jelikož jedním z výsledků analýzy bylo, že cvičící tento box při opravě nepotřebují. Zatím se nebude v levé části počítat ani s boxem s ohodnocenými odpověďmi. Při vývoji bude však kladen důraz na to, aby bylo snadné kdykoliv v budoucnosti do této části nový box přidat. Všechny tyto boxy bude možné zabalit (minimalizovat). Tato možnost je na obrázku 2.1 u všech boxů vyznačena symbolem + či – v jejich pravém horním rohu. Stejná funkce je již v současné verzi, ale v té nové bude umožněno minimalizovat také boxy s jednotlivými zadáními.

Uživatel bude mít možnost si nastavovat pozici dělící čáry, která se nachází uprostřed stránky a tím si zvětšovat jednu polovinu stránky na úkor druhé. Obě poloviny stránky boudou zvlášť skrolovatelné. Na levé straně bude moci uživatel pomocí funkce *drag and drop* každý z boxů uchopit a přesunout na jinou pozici v rámci levé strany. Uživateli tímto bude umožněno si tyto boxy dle vlastních preferencí přeskupit.

Tyto preference se pro každého uživatele uloží a stejné nastavení se pak bude aplikovat i při zobrazení této stránky s jinou otázkou, ale se stejným typem odpovědi. Stejným způsobem se uloží také to, které boxy mají být po příchodu na stránku minimalizované.

Možnost změnit pořadí boxů v části stránky s údaji o otázce a možnost samostatně touto polovinou stránky skrolovat, nabídne vyučujícím větší komfort při opravování odpovědí. Tuto polovinu si totiž budou moci libovolně přizpůsobit dle svých preferencí nezávisle na zbytku stránky. Na pravé straně budou ve výchozím rozvržení zobrazeny odpovědi studentů.

Druhým typem rozvržení prvků na stránce bude zobrazení prvků pod sebou. To bude určeno především pro zobrazení ve zúženém okně prohlížeče a také na mobilním zařízení. Na stránce budou nejprve zobrazeny zadání a další informace o otázce (levá strana ve výchozím rozvržení) a pod touto částí se budou nacházet studentské odpovědi. Zde by se dalo zamyslet nad nějakým řešením, aby i v tomto zobrazení byly během opravování vidět informace o otázce. Během analýzy bylo však zjištěno, že uživatelé na mobilním zařízení testy opravovat nechtějí a vyhovuje jim rozvržení prvků na stránce do dvou polovin. Z těchto důvodů bude lepší se při vývoji zaměřit na jiné a podstatnější aspekty nového frontendu.

V horní části stránky se bude nacházet „hlavička“. Ta bude u obou typů rozvržení stejná a bude obsahovat název právě opravované otázky, tlačítka na přepnutí režimu rozvržení stránky, tlačítko na řazení otázek, a tlačítko „Zpět“. To vrátí uživatele na stránku s tabulkou otázek k opravovanému testu. Dále zde bude umístěno vyhledávací pole, které uživateli umožní na základě zadaného textu filtrovat již opravené otázky (více v kapitole 2.2.3).

2.2.2 Zadání, úkol a referenční odpověď

Všechna zadání přiřazená k právě opravované otázce budou umístěna pod sebou v boxu *Zadání*, ve výchozím rozvržení na levé straně stránky. Jak již bylo zmíněno, jednotlivá zadání bude možné minimalizovat. Změna pořadí zde však k dispozici nebude, protože u zadání může na pořadí záležet a vyučující by měl zadání vidět stejně, jako ho viděl student když odpověď na otázku vyplňoval.

V levém horním rohu každého zadání bude umístěno tlačítko, které uživateli umožní zadání upravit. Kliknutím na toto tlačítko se otevře stránka s editačním formulářem. Bude se jednat o odkaz do jiné části portálu, která se bude starat o správu zadání. Stránka se v prohlížeči otevře na nové kartě

a uživatel tak při kliknutí na úpravu zadání neopustí stránku s hodnocením odpovědí. Jelikož se možnost úpravy zadání z tohoto místa skoro nevyužívá, toto řešení bude dostačující.

Na obrázku 2.1 je ukázán příklad, kdy má otázka pouze jedno zadání. Obsah se zadáním se bude lišit dle typu zadání. Všechny typy zadání budou zobrazeny stejným způsobem jako ve staré verzi. V kapitole 1.4 byl popsán nový typ zadání – *databáze*, který bude přidán s novým backendem. Ten má využití u otázek typu *SQL* a *RA* a na tomto místě se mezi zadáními nijak zobrazovat nebude.

Úkol u otázky je pouze textové sdělení toho, co měl student při psaní testu v rámci dané otázky udělat. Box s úkolem tak pouze zobrazí tento text.

Referenční odpověď se bude lišit podle toho, jaký typ odpovědi právě opravovaná otázka má. U některých typů bude zobrazení referenční odpovědi odlišné od zobrazení studentských odpovědí, ale řadu prvků budou obě zobrazení sdílet. Jak konkrétně budou jednotlivé typy odpovědí zobrazeny je popsáno v kapitole 2.2.5.

Některé otázky budou moci obsahovat více referenčních odpovědí. Jedná se o otázky, jejichž odpovědi jsou typu *text*, *SQL*, *RA*, *diagram* a *transformace*. Pokud bude mít otázka více referenčních odpovědí, bude je možné zobrazit kliknutím na volbu „Zobrazit další“ pod primární referenční odpovědí. Primární referenční odpovědí je myšlena ta odpověď, která je zobrazena na prvním místě a zobrazuje se stále, i když je seznam s ostatními referenčními odpověďmi skrytý. Uživatel bude moci během opravování za primární označit jinou referenční odpověď.

Nové referenční odpovědi bude možné k otázce přiřazovat také při jejich hodnocení přímo na stránce s hodnocením otázky. Může tak nastat situace, že stejná odpověď se bude nacházet jak mezi referenčními, tak také mezi studentskými odpověďmi. V tomto případě bude možné kliknutím na tlačítko u referenční odpovědi tuto odpověď mezi studentskými dohledat.

2.2.3 Studentské odpovědi

Studentské odpovědi budou zobrazeny ve výchozím rozvržení na pravé straně stránky. Nejprve zde budou zobrazeny odpovědi, které vyučující již opravil a pod nimi budou dosud neopravené odpovědi. Odpovědi budou rozděleny do boxů (pravá strana na obrázku 2.1). Již opravené odpovědi budou po příchodu na stránku minimalizované a v hlavičce jejich boxu bude uvedeno, kolika body byly ohodnoceny.

Jeden box s odpovědí bude moci reálně představovat více odpovědí od různých studentů. Stejné odpovědi budou totiž seskupeny a budou při opravování zobrazeny jako jedna odpověď. V hlavičce každé odpovědi bude uveden údaj o počtu odpovědí, které se opravou odpovědi v boxu reálně ohodnotí. Strategie seskupování odpovědí se bude lišit dle typu odpovědi.

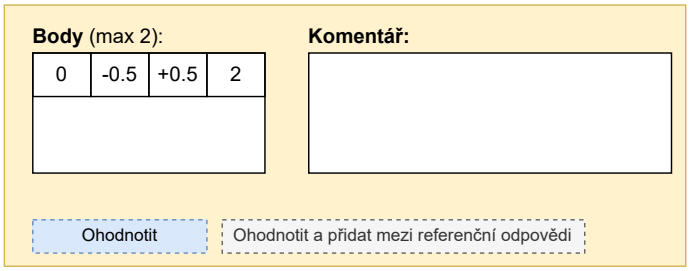
Opravené odpovědi bude možné filtrovat pomocí vyhledávacího pole v hlavičce. Pokud bude toto pole prázdné, budou se na stránce zobrazovat všechny opravené odpovědi. Jakmile začne uživatel do pole psát text, budou se zobrazovat pouze boxy s odpověďmi, které nějakým způsobem obsahují text zadaný v poli. Způsob, jakým se bude v odpovědích zadaný text hledat se bude lišit dle typu odpovědi. Všechny typy odpovědí bude možné vyhledávat podle komentáře, který vyučující zadal při opravě. U typů *text*, *SQL*, *RA* a *transformace* se bude vyhledávání vztahovat také na text, který obsahují odpovědi studentů. U odpovědi typu *diagram* se budou prohledávat názvy entit a atributů, které obsahuje diagram v odpovědi. Tento způsob prohledávání opravených odpovědí umožní vyučujícímu si zpětně vyhledat odpověď, kterou již opravil, aniž by musel minimalizované boxy s opravenými odpověďmi postupně rozbalovat a nahlížet do nich.

2.2.4 Hodnocení odpovědí

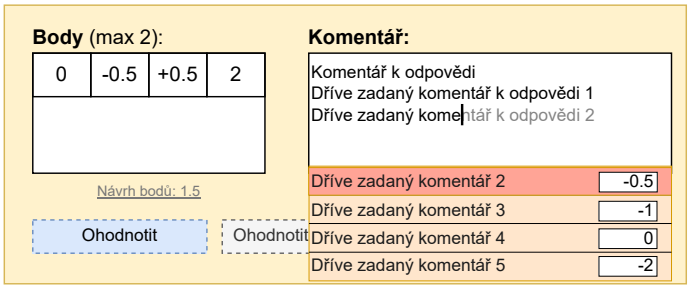
Na základě rozhovorů s cvičícími bylo zjištěno, že způsob jakým se zadává hodnocení (body a komentář) je vyhovující. Návrh formuláře pro hodnocení odpovědí je zobrazen na obrázku 2.2. Formulář na zadávání bodů bude stejný jako v současné verzi. Bude se skládat z textového pole, kam uživatel zadá počet bodů, které za otázku uděluje. To bude moci učinit buď pomocí klávesnice nebo pomocí tlačítek nad vstupním polem. Vedle pole pro zadávání bodů bude pole pro zapsání komentáře. Také to bude na první pohled stejné, jako v současné verzi, ale bude navíc obsahovat funkci „našeptávač komentářů“.

V rozhovorech se cvičícími bylo zjištěno, že řada cvičících při hodnocení odpovědí používá stále stejné komentáře. Právě z tohoto důvodu byla do návrhu nového frontendu funkce našeptávače zapracována. Její návrh je zobrazen na obrázku 2.3. Pokud uživatel začne psát komentář, který začíná stejně jako komentář, který se v hodnocení vyskyt již dříve, zobrazí se mu pod textovým polem seznam. V tomto seznamu uvidí všechny dříve zadané komentáře, které začínají stejně jako dosud zadaný text na řádku. Uživatel bude moci seznamem procházet a nakonec si v něm zvolit požadovaný komentář, který se pak do textu doplní. Při procházení možností uvidí uživatel náhled doplňovaného textu také přímo v textovém poli. Komentáře budou v textovém poli seřazeny dle četnosti využití v dříve opravených odpovědích. To znamená, že čím více vyučující použil daný komentář při hodnocení odpovědí, tím výše v seznamu bude a tím rychleji se k němu vyučující při opravě dalších odpovědí dostane.

Při vytváření návrhu se předpokládalo, že komentář obvykle popisuje nějakou chybu, za kterou bude chtít opravující také strhnout vždy stejný počet bodů. Proto bude na konci každého řádku s nabízeným komentářem pole, kde je počet bodů uveden. Tento počet bodů si bude moci uživatel vždy po kliknutí do daného pole přepsat a dále se mu bude při opravování u navrženého komentáře zobrazovat změněný počet bodů. Záporné body z navržených komentářů, které budou ze seznamu přidány k hodnocení odpovědi se sečtou.



Obrázek 2.2: Formulář pro hodnocení odpovědí



Obrázek 2.3: Zadávání komentáře s našeptáváním

Na základě nich se pak spočítá návrh bodů, který se zobrazí uživateli (obrázek 2.3 pod polem na zadávaní bodů). Kliknutím na tento text se navržené body zkopírují do pole na zadávání bodů. Tento návrh bude vypočítán na základě komentářů vybraných ze seznamu a nově zadané komentáře ho nijak neovlivní. Opravující tak tento návrh bodů může vzít například jako nové maximum, ale nakonec otázku ohodnotit jiným počtem bodů.

Pod prvky na zadávání bodů a komentáře se nachází dvě tlačítka: „Ohodnotit“ a „Ohodnotit a přidat mezi referenční odpovědi“. Po kliknutí na první tlačítko bude hodnocení otázky uloženo, otázka se minimalizuje a v její hlavičce se zobrazí počet udělených bodů. Je třeba brát v potaz to, že po kliknutí na toto tlačítko se musí ohodnotit všechny odpovědi, které jsou pod hodnocenou otázkou seskupeny.

Na druhé hodnotící tlačítko bude možné kliknout pouze v případě, že v poli pro zápis bodů bude zapsán maximální počet bodů. Toto tlačítko uloží hodnocení odpovědi a zároveň odpověď označí za referenční. Tlačítko bude k dispozici pouze u těch typů odpovědí, které mohou mít referenčních odpovědí více a takto označená odpověď se hned po ohodnocení zobrazí mezi ostatními referenčními odpověďmi v boxu *Referenční odpověď*.

V případě, že studentská odpověď bude již zařazena mezi referenčními, bude tlačítko, které slouží pro přidání mezi referenční odpovědi obsahovat informaci, že odpověď již referenční je. Kliknutím na toto tlačítko pak uživatel odpověď z referenčních odstraní. Pokud by odpověď byla referenční a uživatel

snížil počet jejích bodů, pak tato odpověď musí být vyřazena i z referenčních odpovědí a uživatel na toto musí být upozorněn.

Po odeslání hodnocení se uživateli na nějaký čas zobrazí v pravém horním rohu informační prvek s počtem ohodnocených odpovědí a případně také s informací o přidání odpovědi mezi referenční odpovědi (popř. odebrání z referenčních odpovědí).

2.2.5 Zobrazení odpovědí dle typu

V této části se podíváme na návrh zobrazení konkrétních typů odpovědí. U některých typů se bude zobrazení odpovědí lišit dle toho, jestli se jedná o odpověď referenční nebo studentskou. Dále jsou v této kapitole popsány další prvky, které bude obsahovat stránka pro hodnocení odpovědí a které jsou specifické pro daný typ odpovědi.

2.2.5.1 Text

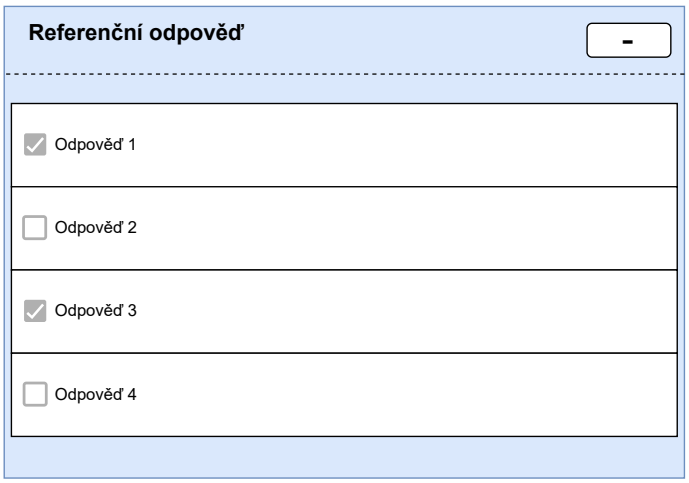
Oproti současné verzi nebude ve zobrazení odpovědí typu *text* žádná změna. Jak referenční, tak studentská textová odpověď bude zobrazena jako prostý text. Studentské textové odpovědi bude možné přidávat mezi referenční odpovědi a referenčních odpovědí tak bude moci mít jedna otázka více.

2.2.5.2 Checkbox a radio list

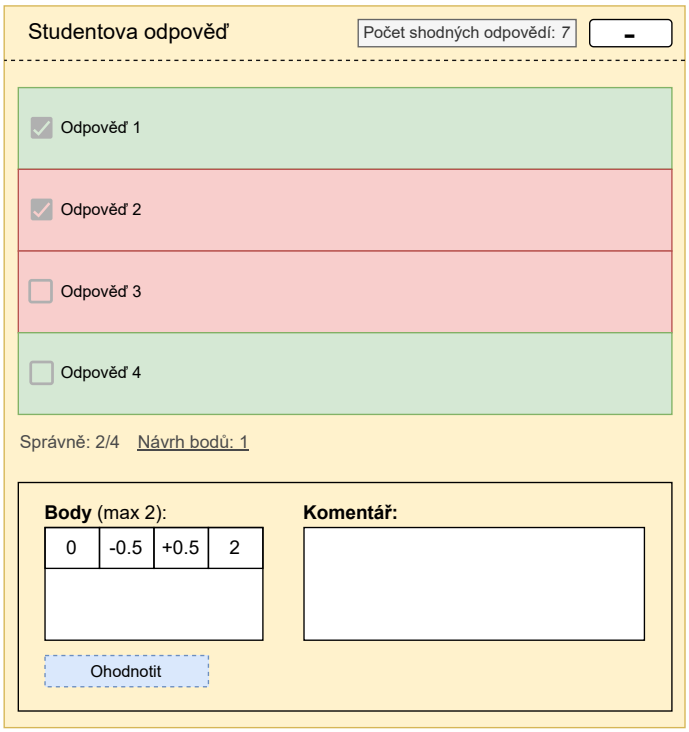
Typ *checkbox* i *radio list* budou mít vždy jen jednu referenční odpověď. Typ *radio list* se pokaždé opraví automaticky a cvičícímu se zobrazí pouze pokud si bude chtít prohlédnout automaticky opravené odpovědi. Odpovědi typu *checkbox* sice není možné opravit vždy automaticky, ale díky tomu, že stejné odpovědi budou seskupeny do jednoho boxu, nebude jich k manuální opravě moc.

Referenční odpověď u obou typů bude znázorněna výčtem všech možností, vedle kterých bude odpovídajícím formulářovým prvkem vyznačeno, zda má být daná odpověď zaškrtnuta či nikoliv. Návrh zobrazení referenční odpovědi typu *checkbox* je na obrázku 2.4. Referenční odpovědi u typu *radio list* bude vypadat obdobně, akorát zaškrtnutá možnost tam bude vždy jen jedna. V současné verzi je referenční odpověď těchto typů znázorněna podobným způsobem, ale ještě navíc jsou zaškrtnuté možnosti podbarveny zeleně a nezaškrtnuté červeně. Tato podbarvení byla v novém návrhu odstraněna, protože nám dávala stejnou informaci, jako formulářové prvky.

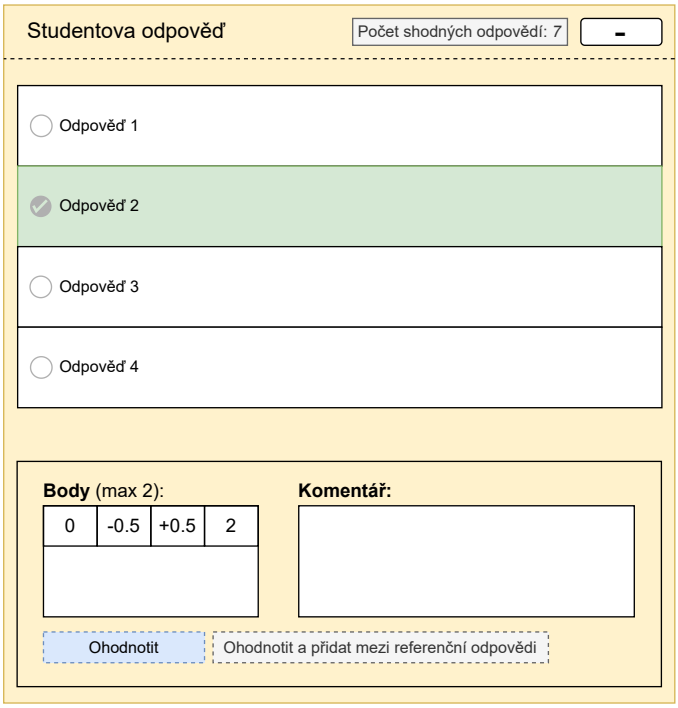
Studentská odpověď typu checkbox, jejíž návrh je na obrázku 2.5, bude tímto podbarvením vyznačovat, jestli se student u dané možnosti rozhodl správně pro její zaškrtnutí. Zeleně budou podbarveny odpovědi, které student buď zaškrtnul a dle referenční odpovědi mají být zaškrtnuty, nebo odpovědi, které nezaškrtnul a ani správně být zaškrtnuty nemají. V opačných případech bude možnost podbarvena červeně.



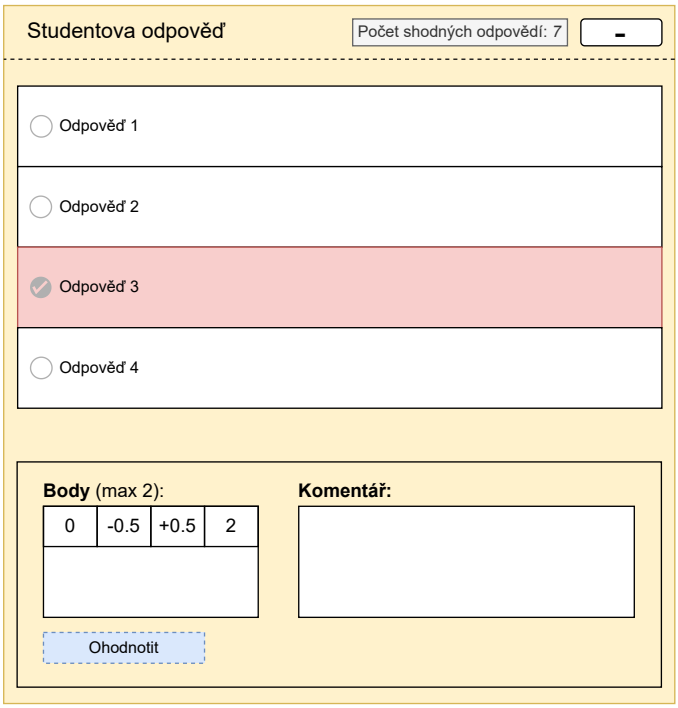
Obrázek 2.4: Referenční odpověď typu checkbox



Obrázek 2.5: Studentská odpověď typu checkbox



Obrázek 2.6: Správná studentská odpověď typu radio list



Obrázek 2.7: Špatná studentská odpověď typu radio list

U studentských odpovědí typu *checkbox* bude uvedeno kolik možností z kolika měl student správně. Dále zde bude zobrazen návrh bodů, který bude vypočítán právě na základě poměru správných a špatných odpovědí. Přesto, že hodnocení nelze plně zautomatizovat, tento návrh opravujícímu alespoň ukáže, za kolik bodů by byla odpověď ohodnocena, kdyby se o její opravě rozhodovalo prostým výpočtem poměru správných a špatných voleb. Je pak na rozhodnutí opravujícího, jestli návrh využije. Kliknutím na text s návrhem se navržené body vloží do pole pro zadávání bodů.

Zobrazení studentské odpovědi u typu *radio list* bude jednodušší. Bude obsahovat výčet možností a studentova volba bude vyznačena zaškrtnutým prvkem *radio button*. Možnost se studentovou volbou bude podbarvena červeně, pokud se jedná o špatně zvolenou možnost, nebo zeleně, pokud je odpověď správná. Správně zodpovězená otázka tohoto typu je na obrázku 2.6, špatná odpověď je pak na obrázku 2.7. Správnou odpověď opravující uvidí v referenční odpovědi.

2.2.5.3 Diagram

U odpovědi typu diagram zůstává hlavním prvkem komponenta *Kreslítko*. Tato komponenta se používá na stránce pro opravu i v současné verzi. Také v nové verzi se bude využívat pro zobrazování jak referenční, tak studentské odpovědi. Otázka s odpovědí tohoto typu bude moci obsahovat více referenčních odpovědí.

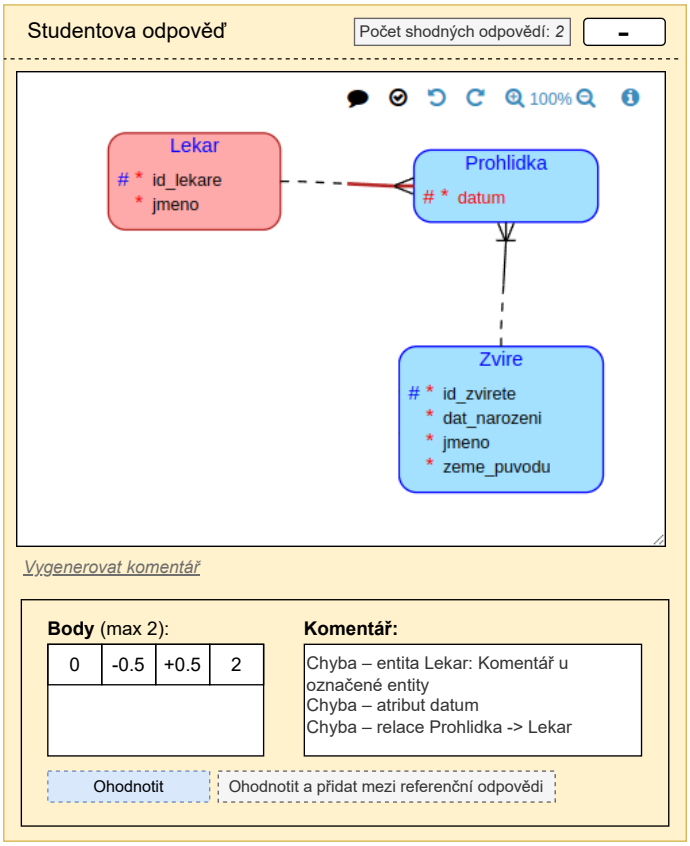
Kreslítko nabízí několik různých režimů a jeden z nich lze využít také při opravování studentských odpovědí. Umožňuje uživateli přímo v diagramu zvýrazňovat chybné entity, atributy nebo vazby. Uživatel pak může k jednotlivým vyznačeným prvkům také přidat textový komentář. Toto zvýrazňování chyb kromě uživatele provádí také systém během automatické opravy odpovědí.

Návrh nového frontendu počítá s tím, že funkcí, které již *Kreslítko* obsahuje, využije. Jejich využití bude vypadat tak, že opravující klikne na volbu pod diagramem, a tím do textového pole pro komentář vloží text vygenerovaný na základě oprav diagramu. Na obrázku 2.8 je zobrazen návrh studentské odpovědi typu *diagram* s vloženým snímkem *Kreslítka*. Zároveň je zde ukázka vygenerovaného komentáře na základě chyb zvýrazněných v diagramu.

Je také důležité i v nové verzi zachovat to, aby se diagram ve studentské odpovědi uspořádal stejně, jako diagram v odpovědi referenční (pokud je to možné).

2.2.5.4 SQL a RA

U odpovědí typu *SQL* a *RA* je nutné kromě samotného dotazu zobrazit také jeho výsledek. Návrh počítá s tím, že oba typy budou zobrazovány podobně. Typy *SQL* a *RA* budou podporovat více referenčních odpovědí u jedné otázky. Ukázka návrhu studentské odpovědi typu *RA* je na obrázku 2.9. Odpovědi

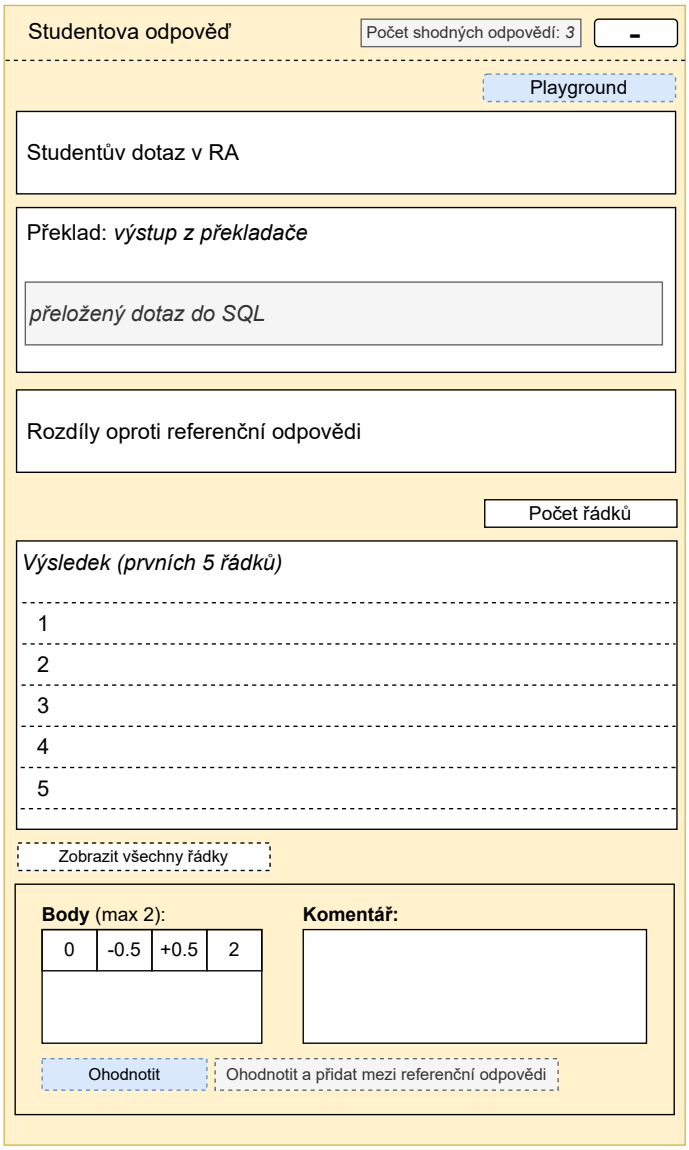


Obrázek 2.8: Studentská odpověď typu diagram s vygenerovaným komentářem

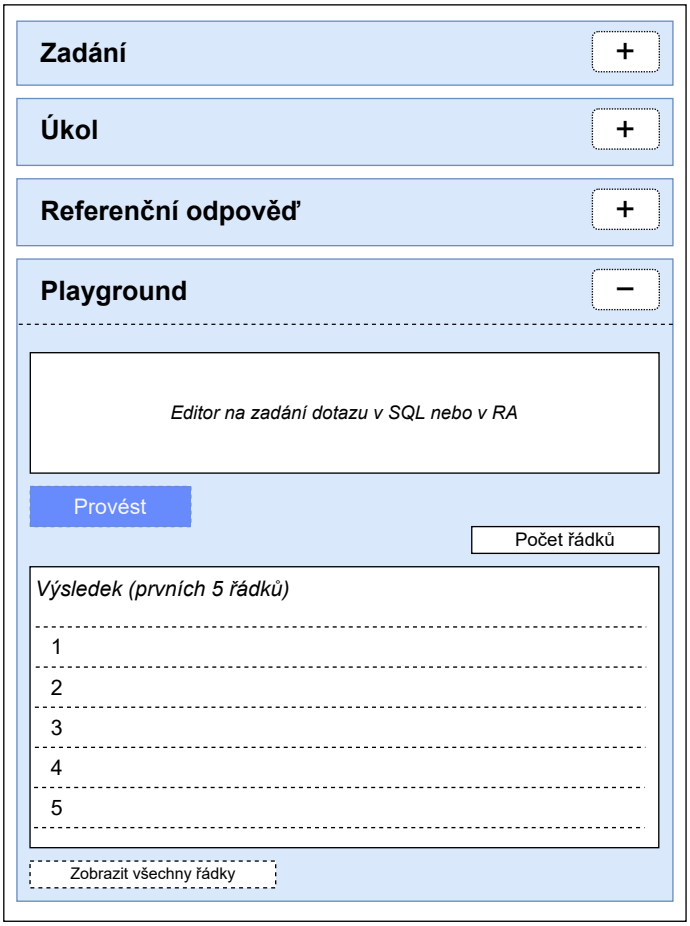
typu *SQL* se od typu *RA* liší jen velmi málo. U všech těchto odpovědi bude vždy nejprve zobrazen samotný dotaz a pod ním už bude jeho výsledek.

Relační algebra se musí nejprve na serveru přeložit do SQL. Proto bude v případě typu *RA* pod dotazem v relační algebře uveden výsledek překladu a přeložený dotaz v SQL. Studentské odpovědi obou typů budou dále obsahovat výpis rozdílů oproti odpovědi referenční (např. rozdíly v počtu řádků či sloupců, hodnoty v řádcích). Porovnání bude vždy provedeno vůči referenční odpovědi, která je označena jako primární (popsáno v kapitole 2.2.2).

Dále již bude u studentských i referenčních odpovědi zobrazen výsledek dotazu. Zde bude uveden celkový počet řádků a prvních pět řádků výsledku. Pokud má výsledek více než pět řádků, uživatel bude moci zobrazit všechny řádky kliknutím na tlačítko „Zobrazit všechny řádky“. Při provádění dotazu může nastat chyba. V takovém případě se žádný výsledek zobrazit nemůže. Pod dotazem se ale vypíše chybová hláška, kterou zaslal server.



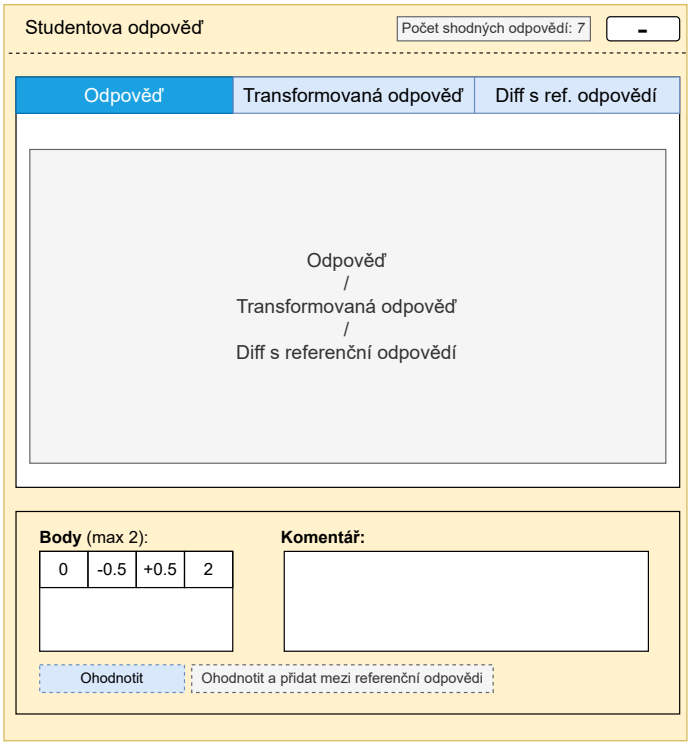
Obrázek 2.9: Studentská odpověď typu RA



Obrázek 2.10: Playground u otázek typu SQL a RA

Během analýzy bylo zjištěno, že někteří uživatelé by uvítali u těchto dvou typů odpovědí tzv. *playground*, tedy editor, do kterého by mohli zadat SQL dotaz nebo dotaz v relační algebře a provést ho nad stejným připojením, jako má otázka. Playground bude umístěn ve vlastním boxu v místech, kde se budou nacházet také boxy se zadáním, úkolem a referenční odpovědí (obrázek 2.10. Ve výchozím zobrazení stránky se tedy bude box nacházet na levé straně a pomocí funkce *drag and drop* jej bude možné přesunout na jiné místo mezi dalšími zmíněnými boxy.

Box pro playground bude vypadat velmi jednoduše. Bude v něm umístěn editor, do kterého bude dotaz zadán. Pod ním se bude nacházet tlačítko pro provedení dotazu. Po kliknutí na toto tlačítko se zobrazí výsledek dotazu, který bude vypadat stejně, jako výsledek dotazu u referenčních a studentských odpovědí. Na obrázku 2.10 je znázorněn playground již s provedeným dotazem a zobrazeným výsledkem.



Obrázek 2.11: Studentská odpověď typu transformace

Na obrázku 2.9, ve kterém je studentská odpověď typu RA, si pak ještě můžeme nad dotazem všimnout tlačítka „Playground“. Toto tlačítko bude u všech dotazů jak v referenčních, tak ve studentských odpovědích a po kliknutí na něj se dotaz pod ním zkopíruje do pole v boxu playground. Předpokládá se totiž, že cvičící budou playground využívat v případě, že si chtějí nějak upravit dotaz ve studentské odpovědi a prohlédnout si výsledek.

2.2.5.5 Transformace

Zobrazení referenční odpovědi u *transformace* bude obsahovat pouze samotnou odpověď, ve formě prostého textu. Studentská odpověď bude umožňovat zobrazení odpovědi ve třech variantách. Ukázka návrhu je na obrázku 2.11. Všechny varianty odpovědí se budou zobrazovat na stejném místě a uživatel si bude moci vybrat, kterou variantu chce právě vidět. Přepínat mezi variantami bude možné pomocí přepínače, který bude umístěn nad odpovědí (na obrázku 2.11 je vyznačen modře).

První variantou zobrazení bude odpověď ve stavu takovém, jak ji student zadal. Druhá varianta bude obsahovat studentovu odpověď v transformovaném tvaru stejně, jako v současné verzi portálu (tzn. se seřazenými entitami, atributy a relacemi). Poslední varianta zobrazení pak uživateli vyznačí rozdíly

mezi referenční odpovědí a příslušnou studenskou odpovědí v transformovaném tvaru (diff). Otázka typu transformace může mít referenčních odpovědí více. Diff se bude provádět vždy s tou referenční odpovědí, která je označena za primární.

2.2.5.6 Normalizace

Všechny odpovědi typu *normalizace* se opravují sami a tak s nimi vyučující ani v současné verzi testového modulu nepřicházeli moc do kontaktu. Je ale nutné ji při realizaci také zpracovat, protože musí jít zobrazit mezi automaticky opravenými odpověďmi. Vzhled samotné odpovědi ale u tohoto typu zůstává stejný jako v současné verzi, která je popsána v kapitole 1.3.2.2.

Kapitola 3

Realizace

Tato kapitola se zabývá popisem realizace nového frontendu manuálních oprav. Nejprve zde budou popsány technologie, které budou při implementaci využívány a dále samotný postup realizace s vyzdvihnutím některých implementačních detailů.

3.1 Použité technologie

Výsledné řešení musí být integrováno do DBS portálu. Bude se tak využívat řada technologií, na kterých je portál již dnes postaven. Velká část DBS portálu stojí na PHP frameworku *Nette*.

Při vývoji frontendu se na DBS portálu využívá *Bootstrap*, což je framework, který slouží pro vytváření responzivních webových stránek. Je napsán v HTML, CSS a JavaScriptu a nabízí vývojářům řadu předpřipravených stylů a funkcí, které mohou značně urychlit vývoj [12]. Dále portál využívá šablonu pro *Bootstrap* jménem *Adminlte*.

Téměř všechny části portálu mají v současné době frontend implementován pomocí šablonovacího systému *Latte*, který je je součástí frameworku *Nette* [13]. Tento šablonovací systém však nebude při vývoji nového testového modulu využit. Místo něj se použije javascriptový framework *Vue.js*. Primárně v něm bude vyvíjena také aplikace popisovaná v této práci.

3.1.1 Vue.js

Vue (vyslovuje se jako anglické slovo „view“) je javascriptový framework určený pro tvorbu uživatelských rozhraní. Nabízí deklarativní vykreslování a umožňuje tak definovat HTML výstup pomocí JavaScriptu. Další vlastností, kterou se *Vue* vyznačuje je reaktivita. Ta se projevuje tak, že dokáže sledovat změny stavu aplikace a na základě těchto změn aktualizovat to, co je zobrazeno na stránce. [14]

Základním prvkem *Vue* aplikací jsou komponenty. Ty umožňují rozdělit uživatelské rozhraní do nezávislých částí a jednu komponentu lze použít na více místech. Komponenty bývají do sebe různě vnořeny, čímž tvoří větší celky nebo celou aplikaci. Každá komponenta má svůj vlastní obsah a logiku. Nadřazené komponenty mohou svým vnořeným komponentám předávat obsah pomocí tzv. *props*, což jsou uživatelem definované atributy.

Pokud potřebuje vnořená komponenta komunikovat se svou rodičovskou komponentou, může vyslat event, který rodičovská komponenta zachytí. Používá se k tomu vestavěná metoda *emit*. Té se předá název eventu a případně také nějaká data. [15]

Jak bylo zmíněno, komponenty pracují odděleně a každá má svou vlastní logiku. Někdy ale můžeme chtít, aby více komponent sdílelo stejnou funkcionalitu. K tomu slouží ve *Vue* objekty nazývané *mixin*. Tyto objekty mohou obsahovat všechno, co mohou obsahovat komponenty. U komponent je pak možné uvést názvy všech objektů *mixin*, které má daná komponenta využít. Komponenta pak může používat všechny funkce, které jsou v objektu *mixin* nastaveny tak, jako kdyby je měla ona sama. [16]

3.1.2 Mirage.js

Mirage.js je javascriptová knihovna, která umožňuje vytvořit „falešnou“ API jako náhradu za plnohodnotný backend. Jedná se o tzv. mocking. Knihovna vytváří „server“, který je spuštěn u klienta. Funguje tak, že zachycuje všechny požadavky, které by aplikace normálně posílala na nějakou API. Na tyto požadavky pak *Mirage* pošle odpověď dle konfigurace vývojáře. Výhodou této knihovny je, že na straně frontendu se s *Mirage* pracuje stejným způsobem jako by se pracovalo se skutečným API. [17]

3.2 Začátek implementace

Na počátku byl vytvořen nový *Vue* projekt a v rámci něj byl vyvinut první prototyp. Jeho účelem bylo především ověřit si, zda bude možné rozumně realizovat výchozí rozložení dle návrhu a zda bude funkční. V rámci prototypu byla realizována pouze část s hodnocením otázky (bez tabulek s testy a otázkami).

Bylo potřeba rozdělit stránku na dvě poloviny, které bude možné zvlášť skrolovat a dělící čarou mezi nimi umožnit poloviny zmenšovat a zvětšovat. Pro tyto účely byl do prototypu přidán balíček *splitpanes*[1], který umožňuje vytvořit libovolný počet panelů a dělících čar (splitterů). Balíček je navíc plně responzivní a dokáže reagovat i na ovládání pomocí dotykového displeje. Další výhodou je, že komponenta *splitpanes*, která je součástí balíčku, umožňuje roz-

[1] spiltpanes, verze 3.1.1, (c) 2018 Antoni Andre

dělit stránku také horizontálně, čehož bude využito při druhém typu rozvržení stránky. Snímek prototypu je na obrázku 3.1.

3.2.1 Struktura projektu a základní komponenty

Komponenty se ve Vue.js obvykle vkládají do složky components. Již při vytváření prototypu byla uvnitř této složky, z důvodu snazší orientace v komponentách, vytvořena adresářová struktura. Všechny komponenty tak nejsou umístěny v jedné složce. Adresářová struktura uvnitř složky components v prototypu vypadá takto:

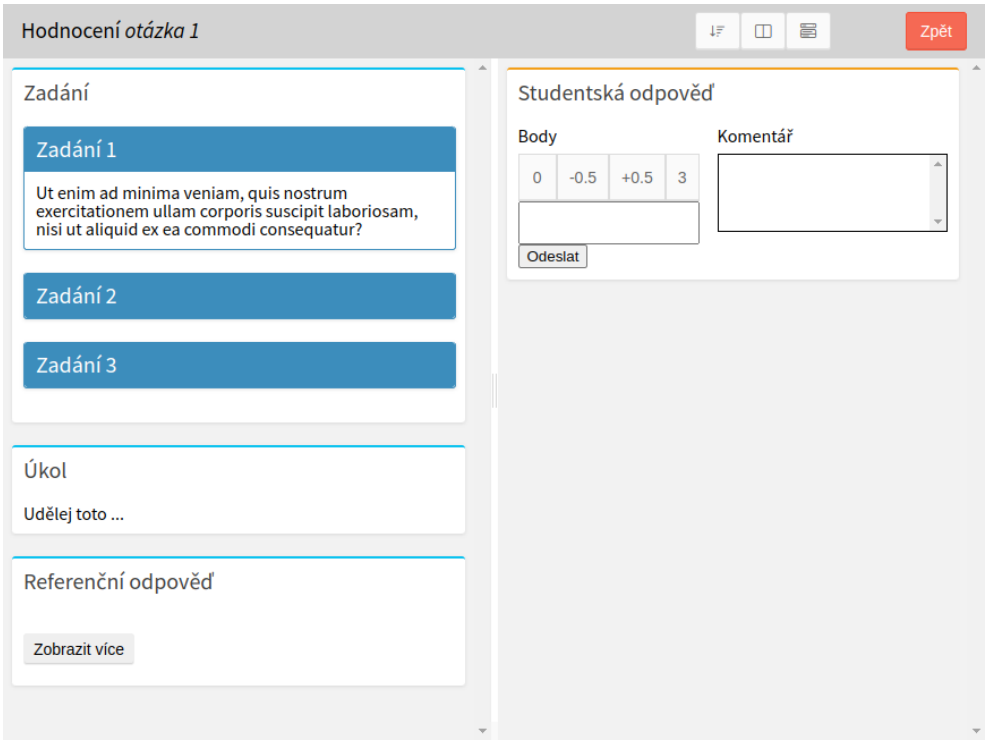
```
components
├── question-pane
│   ├── assignments
│   └── reference-answers
├── answer-pane
└── answer-types
```

Přesto, že v rámci prototypu nakonec velká část projektu vyvíjena nebyla, tato struktura je zachována i ve finálním řešení. Přímo ve složce components se nachází komponenty *QuestionCorrection* a *CorrectionHeader*. Komponenta *QuestionCorrection* je vložena do kořenové komponenty *App* a je základní komponentou celé stránky pro opravu otázek. Nachází se v ní komponenty *CorrectionHeader* (hlavička stránky pro opravu otázek) a *splitpane*, který má dva panely. V prvním (levý ve výchozím rozvržení stránky) jsou komponenty související s údaji o otázce a ve druhém (pravý ve výchozím rozvržení) jsou komponenty se studentskými odpověďmi.

Komponentami, které souvisí s údaji o otázce se rozumí komponenty zajišťující zobrazení zadání, úkolů a referenčních odpovědí. Všechny tyto komponenty jsou umístěny ve složce question-pane. Hlavní komponentou je zde komponenta *QuestionPane*, která zastřešuje všechny ostatní komponenty s údaji o otázce. Dále se v této složce nachází *boxy*, což jsou komponenty, které odpovídají boxům z návrhu. V komponentě *AssignmentsBox* se zobrazují zadání a v *TaskBox* úkoly. Pro zobrazení boxu s referenčními odpověďmi je určena komponenta *ReferenceAnswerBox*. Později (není v prototypu) byla na toto místo přidána také komponenta *RaSqlPlayground*, ve které se nachází playground pro otázky typu *SQL* a *RA*.

V adresářové struktuře si také můžeme všimnout složek assignments a reference-answers. První zmíněná složka obsahuje komponenty pro jednotlivé typy zadání, druhá pak pomocné komponenty pro zobrazování referenčních odpovědí.

Složka answer-pane obsahuje komponenty související se studentskými odpověďmi. V této části bylo nutné vyřešit samotné zobrazování odpovědí studentů a jejich hodnocení. V rámci vývoje prototypu byla větší pozornost směřována na komponenty kolem zobrazení údajů o otázce. Už zde však byly



Obrázek 3.1: Prototyp aplikace s hodnocením otázky

zkoumány možnosti pro realizaci funkce našeptávání komentářů. Kompletně byl ale tento požadavek vyřešen až po integraci do portálu a tak je celé jeho řešení popsáno v kapitole 3.5.4.3.

Komponenty určené pro hodnocení otázky byly tedy rozděleny do dvou skupin. Do skupiny zabývající se údaji o otázce a do skupiny zabývající se odpověďmi studentů. Jeden typ komponent ale mohou obě tyto skupiny sdílet. Jedná se o komponenty určené pro zobrazení konkrétních typů odpovědí. V první skupině chceme zobrazovat odpovědi referenční a v druhé pak ty studentské. Z tohoto důvodu je ve složce components umístěna ještě složka answer-types. V ní se tyto komponenty nacházejí.

3.2.2 Integrace do portálu

Postupem času bylo v prototypu třeba čím dál více využívat knihovny a části kódu, které obsahuje DBS portál (např. správná verze *Adminlte*, *Kreslítko*). Bylo by samozřejmě možné tyto součásti integrovat také do prototypu, ale jelikož má být výsledek nakonec součástí DBS portálu, byla by tato práce zbytečná. Navíc by pozdější integrace prototypu ve složitější podobě mohla být komplikovanější. Z tohoto důvodu byl tak prototyp do portálu integrován poměrně brzy.

DBS portál je verzován na fakultním GitLabu, konkrétně v projektu jménem *newDBS*. Za účelem vývoje nového frontendu byla v repozitáři tohoto projektu vytvořena větev *chaluto3_newtests_corrections_frontend*. Tato větev vznikla oddělením od větve *jordapav_newtests_test*, ve které byl již vytvořen základ jiných částí nového testového modulu.

DBS portál je komplexní projekt, který stojí především na *Nette*. Pro části psané ve *Vue* se používá balíček *Webpack*. Všechen kód související s Vue komponentami se vkládá do složky webpack, která se nachází v kořenovém adresáři projektu *newDBS*. Ve složce webpack se nacházejí další složky, ve kterých je umístěn kód pro jednotlivé části portálu, které jsou napsané ve Vue. Nejdůležitější je složka components, která obsahuje Vue komponenty rozdělené do složek dle jednotlivých modulů. Na tomto místě se nachází také složka Newtest, kde je umístěn všechen kód pro frontend nového testového modulu. Komponenty tohoto modulu jsou dále rozděleny do složek dle části, kam v novém testovém modulu patří. Pro část manuálních oprav testů byla na tomto místě vytvořena složka TestCorrection, do které byly nejprve přesunuty komponenty z prototypu a posléze zde také byly vytvářeny komponenty nové.

Každá část vznikajícího nového testového modulu je vyvíjena jako samostatná Vue aplikace. Proto i pro část věnující se opravám testů je potřeba vytvořit kořenovou komponentu App.vue a soubor main.js. Tyto soubory jsou, v souladu s ostatními částmi nového testového modulu, umístěny do složky NewTest/pages/. Zde byla pro tyto soubory rovněž vytvořena složka jménem TestCorrection.

Dále bylo nutné vytvořit v portálu stránku, na které se budou Vue komponenty zobrazovat. Proto byla v Nette části projektu vytvořena PHP třída *TestCorrectionPresenter* a příslušný latte soubor. V tomto souboru se nachází HTML element, do kterého se Vue komponenty vykreslují.

Prototyp byl tak integrován do DBS portálu a při dalším vývoji bylo možné využívat ostatní části portálu. Mohla tak být například dokončena realizace zobrazení zadání typu *diagram*, protože už byla k dispozici komponenta *Kreslítko*. Stejně tak mohl pokračovat vývoj rozvíjením komponent z prototypu nebo vytvářením komponent nových.

3.3 Mock API

Lehce omezujícím prvkem při vývoji nového frontendu manuálních oprav bylo to, že pro něj ještě neexistuje backend. Bylo proto potřeba vytvořit náhradu skutečného API, který bude frontendu dočasně data poskytovat – mock API. K tomu byla využita knihovna Mirage.js.

Jiné části nového testového modulu využívají jiný způsob mockování dat a soubory k němu se nachází ve složce webpack/api. Zde byla v rámci této práce vytvořena složka test-correction, do které byly umístěny všechny soubory, které zajišťují data pro vytvářený frontend.

```
models: {
  test: Model.extend({
    questions: hasMany(),
    templateAuthor: belongsTo('user'),
    testAuthor: belongsTo('user')
  }),
  question: Model.extend({
    test: belongsTo(),
    answers: hasMany(),
    author: belongsTo('user'),
    corrector: belongsTo('user'),
  }),
  answer: Model.extend({
    test: belongsTo(),
    question: belongsTo()
  }),
  user: Model,
},
```

Ukázka kódu 3.1: Definice modelů pro Mirage v souboru server.js

Obsah souboru *server.js* byl vytvořen pomocí návodu na webových stránkách mirage.js [18]. Obsahuje funkci, která se zavolá při spuštění aplikace a která provede konfiguraci mock API. Jsou zde definovány tyto modely: *test*, *question*, *answer* a *user*. Definice těchto modelů a vazeb mezi nimi je vidět v ukázce kódu 3.1.

V souboru *server.js* jsou také definována data, která mock API poskytuje hned po spuštění aplikace. Bylo potřeba mít od každého typu alespoň jednu otázku a ke každé otázce několik odpovědí. Pro větší přehlednost byla vzorová data definována v oddělených souborech, nacházejících se ve složce seed. V souboru *questions.js* byly staticky vytvořeny otázky všech typů a soubor *answers.js* stejným způsobem vytvářel studentské odpovědi. Data v těchto souborech jsou nereálná a byla určena pouze pro potřeby vývoje. Až pro potřeby testování byla získána z databáze DBS portálu skutečná data (více v kapitole 4.2.2).

Dále byly v konfiguraci mock API definovány jednotlivé endpointy, které poskytují data. Funkce které daný endpoint obsluhují jsou napsány tak, aby frontendu poskytovala data v požadovaném formátu. Detailnější popis endpointů, ale také struktura poskytovaných dat je popsána v kapitole 3.8.1.

3.4 Router

Tabulka s testy, tabulka s otázkami z testu a hodnocení otázky jsou tři části, které v tomto textu označujeme pojmem stránka. Je ale důležité zmínit, že vytvářená aplikace je single-page. To znamená, že se uživatel ve skutečnosti nachází stále na jedné stránce, kterou neopouští a na této stránce je dynamicky

```
const routes = [
 {
  path: '/', component: TestsList
 },
 {
  path: '/tests/:testId/questions',
  component: QuestionsList
 },
 {
  path: '/tests/:testId/questions/:questionId',
  component: QuestionCorrection
 }
];

const router = VueRouter.createRouter({
  history: VueRouter.createWebHistory('/new_test/test-correction/'),
  routes,
})
```

Ukázka kódu 3.2: Router – vytvoření

přepisován obsah. V tomto textu se bude pro zjednodušení i nadále nahlížet na tyto tři části jako na stránky.

Tři stránky, definované v návrhu, jsou vlastně jen komponenty, které se na stránce „vyměňují". Tyto komponenty jsou: *TestsList* pro zobrazení tabulky s testy, *QuestionsList* pro zobrazení tabulky s otázkami a *QuestionCorrection* pro zobrazení jedné otázky a studentských odpovědí.

Z pohledu uživatele se ale skutečně tyto tří části jeví jako tři různé stránky. Kromě toho, že se například při výběru a rozkliknutí hodnocení nějaké otázky změní celý obsah stránky, změní se také URL adresa v liště prohlížeče. Aplikace se navíc zachová očekávaným způsobem, pokud se uživatel rozhodne vrátit se na předchozí stránku pomocí tlačítka zpět v prohlížeči. Toto je řešeno pomocí balíčku *vue-router*[1].

Router je vytvořen v souboru *main.js*. V ukázce kódu 3.2 je vidět vytvoření routeru a definice *routes*, tedy toho, jaká komponenta (component) se má zobrazit při zadání specifikované cesty (path). V kódu je vidět, že pokud uživatel vstoupí na stránku, kde je aplikace umístěna a cesta není dál specifikována, zobrazí se mu komponenta *TestsList*. Komponenta *QuestionsList* se vykreslí po zadání cesty /tests/{*id testu*}/questions, ve které je uveden identifikátor testu. Například v případě zadání cesty /tests/1/questions se zobrazí otázky z testu, který má identifikátor 1. Pokud se budeme chtít dostat v testu s identifikátoerm 1 na otázku s identifikátorem 3, použijeme cestu /tests/1/questions/3.

Aby *vue-router* fungoval, bylo nutné změnit nastavení *Nette* routeru, který používá DBS portál. Zde bylo nakonfigurováno nové směrovaní na presen-

[1] vue-router, verze 4.0, (c) 2020 Eduardo San Martin Morote

ter *TestCorrection*, tak že pouští dále všechny požadavky, které jsou serveru posílány na new_test/test-correction/, nezávisle na tom, co je za posledním lomítkem. Obsah za posledním lomítkem si pak zpracovává *vue-router*.

3.5 Hodnocení otázky

V následujícím textu si detailněji popíšeme realizaci části aplikace věnující se hodnocení jedné otázky. Základ implementace této části vznikl již v prototypu a je popsán v kapitole 3.2. Jádrem stránky pro hodnocení otázky je komponenta *QuestionCorrection*. Ta se stará o nastavení rozvržení stránky a o získání dat o otázce z API.

3.5.1 Rozvržení stránky

Po integraci prototypu do portálu bylo potřeba obsah stránky s hodnocením otázky nastavit tak, aby se zobrazoval v souladu s návrhem. To zahrnovalo především to, aby se ve výchozím rozvržení stránka neskrolovala jako jeden celek, ale aby se skroloval pouze obsah v jednom z panelů. Oba panely pak bylo potřeba zakotvit k dolní hraně stránky (respektive k patičce DBS portálu).

V prototypu bylo toto vyřešeno pomocí absolutního pozicování elementu, ve kterém byly oba skrolovatelné panely umístěny. Toto řešení ale nebylo možné aplikovat v portálu, protože aplikace je součástí stránky, která obsahuje také další prvky (hlavička, patička, postranní menu). Vkládaný obsah musí být umístěn do šablony, kterou používají všechny stránky v portálu. Portál tím tedy poměrně omezoval možnosti s pozicováním vkládaného obsahu.

Bylo učiněno několik pokusů, jak dosáhnout požadovaného výsledku. Nakonec se ukázalo, že tohoto výsledku lze docílit tím, že se upraví styly HTML prvků, které jsou nadřazené prvkům, ve kterých se aplikace vykresluje. Problém vyřešilo především aplikování CSS layoutu *flex box* na tyto prvky. Aby nebyly narušeny žádné další části portálu, byla tato úprava stylů provedena pomocí JavaScriptu přímo v komponentě *QuestionCorrection*. Po načtení této komponenty se zavolá funkce *setFlexAddition*, která potřebné styly nastaví. Před odchodem ze stránky je pak nutné vrátit styly prvků do původního stavu. To obstará funkce *unsetFlexAddition*.

Také druhý režim rozvržení stránky, kdy jsou všechny prvky pod sebou, zajišťuje komponenta *QuestionCorrection*. Tento režim se aplikuje, když je stránka zobrazena na užším displeji nebo když si toto zobrazení uživatel sám zvolí kliknutím na ikonku v hlavičce.

3.5.2 Práce s daty

Jelikož byla při vývoji použita knihovna Mirage.js, jsou data z API obstarána stejným způsobem, jako kdyby se komunikovalo s reálným API. Získání dat probíhá ve dvou fázích. Nejprve si aplikace vyžádá údaje o otázce (ty které

budou při výchozím rozvržení stránky na levé straně) a v druhé fázi pak API poskytne všechny studentské odpovědi, které k otázce v rámci daného testu patří.

Všechny tyto výsledky si musí aplikace zpracovat pro další použití. Bylo by určitě vhodnější, kdyby se některá tato zpracování dělala na backendu a na frontend pak přišla data v požadovaném tvaru. Například by stejné odpovědi mohl frontend obdržet už seskupené a celý proces zpracování by se tím na straně frontendu zjednodušil. Jelikož ale backend ještě neexistuje, bylo po konzultaci autora této práce s vedoucím rozhodnuto, že se tento proces provede na frontendu. Jedná se ale spíše o dočasné řešení.

3.5.2.1 Otázka a referenční odpovědi

Data spojená s údaji o otázce jsou získána ve funkci *fetchQuestion* v komponentě *QuestionCorrection*. Na API jsou na tomto místě vzneseny dva požadavky. Nejprve jsou obdrženy údaje o otázce a poté i všechny její referenční odpovědi. Dále se ve funkci data zpracují a pokud má odpověď více referenčních odpovědí, je nutné provést jejich seskupení.

Seskupení referenčních odpovědí se provádí, protože se může stát, že ze serveru přijde více referenčních odpovědí, které mají totožný obsah. Takové odpovědi chceme zobrazit na stránce pouze jednou. Seskupování odpovědí se využívá také u odpovědí od studentů, protože dle návrhu se mají stejné studentské odpovědi zobrazovat na stránce jako jedna odpověď. Návrh také definoval pojem primární referenční odpovědi u otázek, které mají referenčních odpovědí více. Jako primární se při zpracování více referenčních odpovědí nastaví ta odpověď, kterou server zaslal v seznamu referenčních odpovědí jako první.

3.5.2.2 Studentské odpovědi

Studentské odpovědi k opravované otázce se získávají ve funkci *fetchAnswers* v komponentě *AnswerPane*. Tato komponenta si tyto odpovědi ukládá do dvou polí. Do prvního si vloží ty odpovědi, které již byly opraveny, do druhé pak ty neopravené. Po úspěšném načtení z API se odpovědi musí zpracovat pomocí funkce *processDataFromApi*.

Zde je volána funkce *processAnswers*, která seskupí stejné odpovědi a rozřadí je do jednoho z polí, podle toho, zda jsou opravené či nikoliv. Každý typ odpovědi má svou vlastní seskupující funkci. Více se seskupování odpovědí věnuje část 3.5.2.3. Po seskupení opravených i neopravených odpovědí máme dvě pole, kde jsou odpovědi unikátní.

Jedním z požadavků bylo umožnit vyhledávání v opravených studentských odpovědích. To je vyřešeno tak, že po seskupení odpovědí je vytvořeno pole objektů. Toto pole se vytváří ve funkci *createSearchArray*. Každá seskupená odpověď má svůj vlastní objekt. V něm je uložen její identifikátor a pole

řetězců, dle kterých bude možné odpověď najít. Vytváření těchto řetězců se liší dle typu odpovědi a odpovídá návrhu, který je popsán v kapitole 2.2.3.

Aby bylo možné ve výpisu referenčních odpovědí snadno zjistit, jestli se daná referenční odpověď vyskytuje přímo v právě opravovaném testu, odešlou se identifikátory všech opravených odpovědí, které jsou zároveň referenční, komponentě *QuestionCorrection*.

Komponenta *AnswerPane* používá mixin *sqlRaQueryExecuteMixin*. Ten ji rozšiřuje o funkce pro provádění dotazu nad databázovým připojením. Všechny dotazy typu *SQL* i *RA* je totiž nutné provést společně už při jejich zpracování, protože dle výsledků jejich provedení je budeme chtít dále seřadit.

Funkce pro řazení odpovědí poskytuje *sortAnswersMixin*. Odpovědi jsou seřazeny vždy při novém načtení stránky nebo po kliknutí na tlačítko pro seřazení odpovědí v hlavičce stránky s opravou otázky. Aktuálně je realizováno pouze řazení u odpovědí typu *SQL* a *RA*, které se řadí dle podobnosti výsledků.

3.5.2.3 Seskupování odpovědí

Funkce na seskupování odpovědi jsou v *groupAnswersMixin*. Samotný způsob seskupování i jeho výsledek se liší podle typu odpovědi. Pokud by bylo v budoucnosti rozhodnuto o změně strategie seskupování odpovědí, změna by se provedla právě na tomto místě.

Všechny seskupující funkce fungují na podobném principu. Jejich výsledkem je mapa, která obsahuje páry, kde klíčem je řetězec, podle kterého se seskupování provádí a hodnotou je objekt se seskupenou odpovědí. Uvnitř tohoto objektu je pole s identifikátory všech stejných odpovědí. To je poté využito ke zpětnému dohledání všech odpovědí, které jsou v rámci jedné odpovědi seskupeny. Zpětné dohledání se pak hodí při odesílání hodnocení, kdy je potřeba ohodnotit všechny odpovědi, které se pod hodnocenou seskupenou odpovědí nacházejí.

Odpovědi typu *text* a *transformace* mají společnou funkci na seskupení *groupTextAnswers*. Jako klíč pro seskupení se použije přímo řetězec s odpovědí studenta. Protože u typu *transformace* chceme zobrazovat vždy odpověď ve tvaru, ve kterém ji student napsal, není odpověď seskupována dle odpovědi v transformovaném tvaru.

U typů *checkbox* a *radio list* se do jedné odpovědi seskupí ty odpovědi, které jsou, co se týče zvolených a nezvolených možností, stejné. O jejich seskupování se stará funkce *groupCheckboxAnswers*.

Při seskupování odpovědí typu *diagram* ve funkci *groupDiagramAnswers* je potřeba porovnat objekty reprezentující diagramy mezi sebou. Kdybychom objekt serializovali do formátu JSON a následně tento řetězec použili jako klíč v mapě pro seskupení, některé stejné diagramy by mohly zůstat zvlášť. U tohoto způsobu se totiž nelze spolehnout na pořadí, v jakém se atributy serializují. Místo toho jsou proto objekty s diagramy hashovány pomocí balíčku

object-hash[1]. Ten poskytuje funkci, které je předán objekt a ona vrátí jeho hash. Zároveň umožňuje před hashováním uvést vlastnosti objektu, které se mají při hashování ignorovat. Toho je využito u položky *transform* a balíček nám tak umožní získat stejnou hash i pro diagram, který je sice stejný, ale je jinak uspořádán. S výsledným hashem se pak pracuje jako s klíčem mapy se seskupenými odpověďmi.

Typy *SQL* a *RA* se seskupují společně ve funkci *groupSqlRaAnswers*. Aktuálně seskupování vypadá tak, že do jedné seskupené odpovědi spadnou ty odpovědi, které mají úplně stejný dotaz. V tomto případě je potřeba aby dotaz ve studentské odpovědi přišel ze serveru v podobě, kde nebude záležet na tom, jaké student použil odsazení, kde používal velká písmena a tak podobně. Seskupování odpovědí tohoto typu podle výsledků by se dalo jednoduše udělat například přes *object-hash*, jako je tomu u diagramů. Dotazy se stejnými výsledky ale mohou být napsány rozdílně. Muselo by se tak určit, který dotaz se bude u seskupené odpovědi zobrazovat.

Podobně jako u diagramů i u typu *normalizace* se používá *object-hash*. Tím je hashován objekt s řešením každé podúlohy. Výsledné hashe se spojí a výsledek tohoto spojení se pak používá jako klíč v mapě. O seskupování odpovědí tohoto typu se stará funkce *groupNormalizationAnswers*.

3.5.2.4 Ukládání hodnocení

Kromě získávání dat z API a jejich zpracování, musí aplikace také odesílat hodnocení studentských odpovědí na server. O ukládání hodnocení se stará funkce *saveEvaluation* v komponentě *AnswerPane*. Této funkci je předán objekt z komponenty *EvaluationForm*, která reprezentuje formulář pro hodnocení odpovědi. Tento objekt obsahuje počet udělených bodů, komentář opravujícího, údaj, zda je odpověď referenční a jestli se tento údaj při prováděném hodnocení změnil. V rámci tohoto objektu se také předává studentova opravená odpověď. Ta se získává také v komponentě *EvaluationForm* pomocí funkce *getCorrectedAnswer* a aktuálně má využití pouze u odpovědí typu *diagram*, kde je potřeba uložit opravu, kterou cvičící provedl přímo do *Kreslítka*. V případě potřeby lze ale tuto funkci snadno rozšířit i o další typy.

V komponentě *AnswerBox* se pak ještě k objektu s hodnocením přidá ID ohodnocené seskupené odpovědi a pole s ID všech odpovědí, které jsou pod touto odpovědí seskupeny.

Toto pole se poté v komponentě *AnswerPane* prochází a pro každou odpověď se odesílá hodnocení na server metodou PATCH. Změna hodnocení se totiž na datech projeví tak, že se u studentské odpovědi změní hodnota u počtu bodů, případně u komentáře. Metoda PATCH slouží právě pro úpravu vybraných hodnot již existujícího zdroje v API. Při vývoji backendu je možné zvážit, zda hodnocení všech totožných odpovědi neposílat na server jedním požadavkem.

[1] object-hash, verze 3.0, (c) 2014 object-hash contributors

Po ohodnocení odpovědi se ID této odpovědi v komponentě *AnswerPane* uloží do pole *lastEvaluatedIDs*. Tímto způsobem si aplikace pamatuje v jakém pořadí uživatel odpovědi hodnotil. Dokud nejsou odpovědi seřazeny, jsou v tomto pořadí zobrazeny mezi ohodnocenými odpověďmi. Po ohodnocení odpovědí jsou všechny odpovědi znovu staženy ze serveru funkcí *fetchAnswers*. V rámci znovunačtení dat dojde také k obnovení výpisu referenčních odpovědí, pokud byla při hodnocení přidána nová.

3.5.3 Panel s údaji o otázce

Data získaná z API v komponentě *QuestionCorrection* jsou předána komponentě *QuestionPane*. O této komponentě jsme se krátce zmínili při popisu prototypu v kapitole 3.2.1. Slouží pro vykreslení boxů se zadáním, úkolem a referenční odpovědí. U typů *SQL* a *RA* se zde vyskytuje také playground.

Již v prototypu byla do této komponenty implementována funkce, která umožní uživateli libovolně měnit pořadí boxů (*drag and drop*). Tato funkce je realizovaná pomocí balíčku *vue-draggable-next*[1]. To, jak si uživatel jednotlivé boxy uspořádá je potřeba uložit a při příchodu na stránku s opravou otázky se stejným typem odpovědi pak boxy zobrazit ve stejném pořadí. Tyto preference se ukládají do *localStorage*. Stejným způsobem se uloží také nastavení toho, které boxy mají být po příchodu na stránku minimalizované.

3.5.3.1 Zadání

Zadání k otázce jsou zobrazována v boxu, který reprezentuje komponenta *AssignmentsBox*. Tento box ve svém těle obsahuje komponentu *Assignment*, která slouží jako „rozcestník“ pro různé typy zadání. Zároveň je komponentě *Assignment* definovaná hlavička, která se zobrazuje u všech typů zadání, s výjimkou zadání typu *databáze*, které se na tomto místě nezobrazuje vůbec.

V hlavičce je uveden název zadání a také odkaz na úpravu zadání, jak specifikoval návrh. Zde je důležité upozornit, že editace zadání bude probíhat v části portálu, která se zaobírá správou zadání. Tato část však není ještě ve verzi nového testového modulu, kterou má autor k dispozici, implementována. Tlačítko „Úprava zadání“ tak odkazuje pouze na stránku s tvorbou zadání a po propojení těchto dvou částí testového modulu bude nutné odkaz přepsat.

Dále se zobrazení zadání liší dle jeho typu. Každý zobrazitelný typ zadání má vlastní komponentu. O zobrazení zadání typu *text* se stará komponenta *TextAssignment* a analogicky jsou zde pak pro ostatní typy zadání ještě komponenty *ImageAssingment*, *DiagramAssignment* a *NormalizationAssignment*.

[1] vue-draggable-next, verze 2.1.1, Anish George

3.5.3.2 Referenční odpovědi

Stejně jako u zadání, také referenční odpovědi mají box, který realizuje komponenta *ReferenceAnswerBox*. Zde je třeba zohlednit to, že otázky s typem odpovědi *text*, *SQL*, *RA*, *diagram* a *transformace* mohou mít referenčních odpovědí více.

Jednu referenční odpověď představuje komponenta *ReferenceAnswer*. Zobrazení více referenčních odpovědí je pak realizováno pomocí komponenty s názvem *MoreReferenceAnswers*. Tato komponenta vykreslí potřebný počet komponent *ReferenceAnswer*. O tom, jakým způsobem se tyto komponenty vykreslí se rozhodne na základě typu odpovědi a počtu referenčních odpovědí v komponentě *ReferenceAnswerBox*. Komponenta *ReferenceAnswer* zobrazuje obsah jedné referenční odpovědi v závislosti na jejím typu. Každý typ odpovědi má svou vlastní komponentu, která se využívá jak u referenčních, tak také u studentských odpovědí. Tyto komponenty jsou popsány v kapitole 3.5.5.

3.5.3.3 Playground

Playground zajišťuje komponenta *SqlRaPlaygroundBox*. Tato komponenta využívá komponentu *AceEditor*, která byla v novém testovém modulu obsažena již dříve. *AceEditor* obsahuje editor, který se v portálu již využívá na více místech a umožňuje zobrazovat a zapisovat dotazy v SQL i v relační algebře. Kopírování dotazu do pole editoru je provedeno pomocí event listeneru *copy-to-playground*, který se v komponentě *SqlRaPlaygroundBox* zaregistruje po jejím vytvoření a při jejím zničení se zase odstraní.

Kterákoliv komponenta tedy může vyslat event tohoto typu i s obsahem, který se má do editoru vložit. Komponenta *SqlRaPlaygroundBox* toto zachytí a pomocí metody *setQuery* text do editoru vloží, pokud je její obash minimalizován, rozbalí se a vyšle emit *scroll-to* rodičovské komponentě *QuestionPane*, která metodou *scrollTo* na box s playgroundem zaskroluje.

SqlRaPlaygroundBox používá *sqlRaQueryExecuteMixin*, který ji rozšiřuje o funkce pro provádění dotazů. Poté co uživatel dotaz zapsaný do editoru nechá provést, zobrazí se výsledek, který databázový stroj vrátí, v komponentě *SqlRaResult*. Ta zobrazí tabulku s řádky, které dotaz vrátil, popřípadě chybovou hlášku. Tato komponenta je využita také pro zobrazování výsledků u odpovědí typu *SQL* a *RA*.

Dotazy zadané do pole playground se provádějí nad databázovým připojením, které je přiřazené k otázce v zadání typu *databáze*. Pokud je odpověď typu *SQL*, provádí se dotaz v SQL a pokud se jedná o *RA*, je vyslán požadavek na provedení dotazu v relační algebře.

3.5.4 Panel se studentskými odpověďmi

Hlavní komponentou, která zobrazuje studentské odpovědi je *AnswerPane*. Kromě toho, že obstarává komunikaci s API a získává data související se stu-

dentskými odpověďmi, zajišťuje také řadu dalších funkcí, jako je například zpracování komentářů určených pro našeptávač nebo vyhledávání mezi opravenými odpověďmi. Dále tato komponenta umožňuje ve svých metodách skrolování na box s určenou studentskou odpovědí. To se využívá při skrolování na první neopravenou odpověď po příchodu na stránku nebo při skrolování na poslední opravenou odpověď po odeslání hodnocení.

3.5.4.1 Zobrazení odpovědí

Uvnitř *AnswerPane* se nachází dvakrát stejná komponenta, která zobrazuje seznam boxů se studentskými odpověďmi (*StudentAnswersList*). První instance této komponenty zobrazuje odpovědi studentů, které již byly opraveny, druhá pak ty neopravené.

Pokud je vyplněné vyhledávací pole v hlavičce nad odpověďmi, zobrazí se v první komponentě *StudentAnswersList* jen ty opravené studentské odpovědi, které zadané heslo obsahují. Vyhledávání v opravených odpovědích se věnovala již kapitola 3.5.2.2.

Jednomu boxu s odpovědí studenta odpovídá komponenta *AnswerBox*. Ta zajišťuje správně zobrazení údajů o odpovědi v hlavičce boxu (počet seskupených odpovědí, počet bodů). Během vývoje byl návrh této hlavičky u opravených odpovědí rozšířen také o informaci o vyplněném komentáři a zda je daná odpověď referenční.

AnswerBox pak ve svém těle obsahuje dvě komponenty. První z nich, *StudentAnswer* slouží pro zobrazení studentské odpovědi. Tato komponenta, podobně jako komponenta *ReferenceAnswer*, pouze rozdělí příslušně údaje o odpovědi dalším komponentám, které již zobrazují konkrétní typ odpovědi (zobrazení typů odpovědí je popsáno v kapitole 3.5.5). Druhou komponentou uvnitř komponenty *AnswerBox* je *EvaluationForm*.

3.5.4.2 Hodnocení odpovědí

EvaluationForm obsahuje formulář pro hodnocení konkrétní odpovědi. Tento formulář se skládá z pole pro zadávání bodů (komponenta *PointsInput*) a z pole pro komentář (komponenta *CommentInput*). V *PointsInput* je samotný vstup pro zadání bodů a také tlačítka pro zadávání bodů pomocí klikání myší. Komponenta má funkce pro obsluhu všech způsobů zadávání bodů.

Samotné pole pro zadávání komentáře v komponentě *CommentInput* je realizováno pomocí elementu *div*, který má nastavený atribut *contenteditable*. Tento atribut umožní uživateli obsah tohoto elementu editovat a zároveň, na rozdíl od elementu *textarea*, poskytne více možností při realizaci našeptávače komentářů.

3.5.4.3 Našeptávač komentářů

Po příchodu na stránku s opravou otázky se nejprve musí řetězce, které budou uživateli při psaní komentářů nabízeny, inicializovat. Toto má na starosti funkce *initializeHints* v komponentě *AnswerPane*. Ta se zavolá hned po zpracování odpovědí.

Řetězce pro nápovědu jsou buď uloženy v *localStorage*, nebo se načtou z komentářů v již opravených odpovědích. Pokud se na obou místech vyskytuje stejný řetězec, přednost má ten z *localStorage*, protože může obsahovat uživatelem nastavené záporné body (popsáno v návrhu 2.2.4). Během vývoje byl autor práce upozorněn vedoucím, že *localStorage* je hodně využíváno také dalšími součástmi DBS portálu a může se tak stát, že v něm nebude místo. V takovém případě budou řetězce pro nápovědu načteny pouze na základě starších komentářů.

Řetězce se ukládají do mapy. Klíčem v této mapě je samotný řetězec a hodnotou je objekt, který obsahuje počet stržených bodů, pole s identifikátory odpovědí, kde se komentář nachází a vlastnost *setByUser*, která udává, zda si body nastavil uživatel a nebo je aplikace vypočítala. Pokud jsou řetězce získány ze starších komentářů, vypočítá se návrh stržených bodů takto:

(max. bodů za otázku) − (*získané body za otázku*/*počet dílčích komentářů*).

Aby bylo možné text z pole pro komentář rozdělit na jednotlivé dílčí komentáře, zavedl autor již v návrhu pravidlo, že dílčí komentáře budou psát uživatelé pod sebe. Dílčí komentáře pak bude možné získat rozdělením vstupu podle symbolu reprezentující nový řádek.

Mapa s nápovědou je po inicializaci předána všem komponentám typu *EvaluationForm*. Tyto komponenty obsahují jak komponentu pro zadávání bodů (*PointsInput*), tak komponentu pro komentář (*CommentInput*). Tvoří tak prostředníka mezi tím, co je napsané v komentáři a tím jaké se navrhují body. Komponenta *EvaluationForm* detekuje změnu textu v poli pro komentář, komentář rozdělí na dílčí a hledá je v mapě. Poté spočítá doporučený počet bodů za odpověď a výsledek předá komponentě *PointsInput*, která jej zobrazí jako návrh.

Řetězec z řádku, na který uživatel právě píše komentář je průběžně zpracováván v komponentě *CommentHint* ve funkci *findAndSetHints*. Zde se v seznamu všech řetězců s nápovědou vyhledají ty, které začínají stejně jako dosud zadaný řetězec na řádku. Vynechávají se řetězce, které už v komentáři u opravované odpovědi jsou. Pokud jsou nalezeny nějaké vhodné nápovědy, zobrazí se v seznamu pod vstupním polem komentáře. V tomto seznamu může uživatel buď pomocí myši, nebo pomocí šipek vybírat návrhy na doplnění komentáře.

Jak již bylo zmíněno, vstupní pole pro komentář je řešeno pomocí elementu *div*, který je *contenteditable*. Toto řešení bylo zvoleno proto, že do tohoto

elementu můžeme snadno vložit další element, ve kterém zobrazíme návrh textu pro doplnění řádku.

Nápovědu může uživatel potvrdit buď kliknutím na danou možnost nebo stisknutím klávesy *enter*. Funkce *setHint* v komponentě *CommentInput* se pak postará o doplnění zbytku řetězce na řádek.

Pokud chce uživatel změnit počet bodů, které mu jsou navrhovány, musí kliknout do pole s informací, kolik bodů se odečítá. Toto pole je umístěno u každé jednotlivé nápovědy v seznamu. Prvky v seznamu nápověd reprezentuje komponenta *CommentHintItem*. V okamžik, kdy uživatel na pole u nějaké nápovědy klikne, se celý našeptavač přepne do režimu editace bodů.

Nápověda zůstává v tomto režimu otevřena, i když pole pro psaní komentáře není aktivní. Uživatel má v tomto režimu možnost body přepsat, stisknout *enter* a změnu uložit a nebo stisknout *escape* a tím do pole s odečítanými body vrátit původní hodnotu. Když uživatel změnu odečítaných bodů potvrdí, komponenta *AnswerPane* ve funkci *changeHintPoints* uloží změnu do *localStorage*. Tato změna se provádí v této komponentě, protože je potřeba aby se změna bodů projevila v komponentách všech studentských odpovědí.

3.5.5 Zobrazení odpovědí dle typu

Jak již bylo zmíněno dříve, každý typ odpovědi využívá pro své zobrazení vlastní komponentu. Pomocí těchto komponent jsou zobrazeny referenční i studentské odpovědi. Tyto, a s nimi související komponenty, jsou umístěny ve složce answer-types.

V některých případech se zobrazení referenční a studenské odpovědi mírně liší. Z tohoto důvodu mají některé typy atribut *reference-mode*. U studentských odpovědí typu *checkbox*, *diagram*, *transformace*, *SQL* a *RA* je pak potřeba provést porovnání s referenční odpovědí. Toto porovnání provádí přímo komponenty zobrazující dané typy a proto jim je potřeba předat také objekt s referenční odpovědí (atribut *reference-answer*).

3.5.5.1 Odpovědi typu checkbox a radio list

Odpovědi typu *checkbox* a *radio list* mají společnou komponentu, která se nazývá *CheckboxType*. Zobrazení těchto odpovědí je skoro stejné. Komponenta porovná odpověď studenta s referenční odpovědí a vyznačí správné a chybné volby v odpovědi způsobem, který byl uveden v návrhu.

U typu *checkbox* pak umí tato komponenta spočítat návrh bodů, který je dán výpočtem:

$$(\textit{počet správných voleb}/\textit{počet všech voleb}) \times \textit{(maximální počet bodů)}.$$

Navržený počet bodů se zobrazí pod studentskou odpovědí. Pokud na něj uživatel klikne, vloží se tento návrh do pole pro zadávání bodů.

3.5.5.2 Odpovědi typu diagram

O zobrazení odpovědí typu *diagram* se stará komponenta *DiagramType*. Stejně jako u zadání, je i zde pro vykreslení diagramu použito *Kreslítko*. Ve studentských odpovědích je u *Kreslítka* povolen režim oprav, kdy je možné vyznačovat a komentovat chyby přímo do diagramu. Komentáře se do *Kreslítka* zadávají pomocí JavaScriptového dialogového okna *prompt*.

Zde by se hodilo, kdyby se komentáře, které uživatel takto zadá, přímo vypisovali do komentářů k hodnocení odpovědi. Je tu ale problém s tím, že u *promptu* nelze jednoduše odchytávat událost, že bylo pole vyplněno a potvrzeno. Tento problém byl vyřešen tak, že pod diagram byl umístěn odkaz. Po kliknutí na něj se z diagramu vygeneruje komentář. Toto generování probíhá tak, že se prochází celý objekt s diagramem a hledají se v něm komentáře k jednotlivým prvkům.

Pří testování s vedoucím práce (kapitola 4.2.4) se ukázalo, že plocha, na které se diagram vykresluje bývá často o dost větší, než je samotný diagram. Proto byl vytvořen mixin *diagramShowMixin*, do kterého se přesunula funkce pro zobrazení diagramu a také zde byla vytvořena funkce *setCanvasHeight*. Tato funkce nalezne nejníže umístěnou spodní hranu entity v diagramu a podle ní nastaví výšku plochy *Kreslítka*. Tím se docílí toho, že se diagram do plochy přesně vejde. Tento mixin byl posléze přidán také ke komponentě pro zobrazení zadání typu *diagram* (*DiagramAssignment*).

3.5.5.3 Odpovědi typu SQL a RA

O zobrazování odpovědí typů *SQL* a *RA* se stará komponenta *SqlRaType*. Tato komponenta je rozšířena o mixin *sqlRaQueryExecuteMixin* a umí tak sama dotazy provádět. U studentských odpovědí jsou ale všechny dotazy provedeny naráz v komponentě *AnswerBox*, aby bylo možné odpovědi dle výsledků seřadit. V tomto případě tak komponenta *SqlRaType* dotazy sama neprovádí, ale obdrží výsledek přes *props*.

Komponenta *SqlRaType* obsahuje již dříve zmíněný *AceEditor*, který ale tentokrát nemá povolen režim editace. V něm se zobrazuje dotaz zadaný v odpovědi. Dále je v této komponentě tlačítko, které zkopíruje dotaz v odpovědi do pole playground.

Pokud nemá komponenta *SqlRaType* nastaven referenční režim (vlastnost *referenceMode*), očekává, že dostane, podobně jako odpověď typu *Checkbox*, referenční odpověď pro porovnání. Komponenta vezme výsledek dotazu ve své studentské odpovědi a porovná jej s výsledkem referenční odpovědi. Nejdřív se porovná, zda sedí počet sloupců, dále počet řádků a nakonec se hledají rozdíly v jednotlivých řádcích. Pokud se během porovnání narazí na nějaký rozdíl, zobrazí komponenta informaci, že se příslušná studentská odpověď liší od té referenční a vypíše také důvody.

Provedení dotazu může buď vrátit množinu řádků, nebo může skončit chybou. U relační algebry pak může nastat také chyba překladače do SQL. Výsledek dotazu se předává komponentě *SqlRaResult*, která ho vypíše. V případě úspěchu vypíše počet vrácených řádků a v *SqlRaResultTable* řádky vypíše. V případě neúspěchu pak komponenta *SqlRaResult* vypíše chybovou hlášku. Pokud je odpověď typ *RA*, zajistí ještě komponenta *RaTranslationResult* zobrazení výsledku překladu do SQL a buď vypíše přeložený SQL dotaz nebo chybovou hlášku.

3.5.5.4 Odpovědi typu transformace

Komponenta *TransformationType* obsahuje řadu funkcí pro vytvoření transformované odpovědi a diffu mezi transformovanou a referenční odpovědí. Tyto funkce byly v podstatě převzaty ze současné verze portálu, kde byly umístěny v souboru *StudentTransformationShow.latte*, a byly jen mírně upraveny. Odpověď typu transformace může mít více referenčních odpovědí. Diff je proto vždy vytvářen s tou referenční odpovědí, která je označena za primární.

3.5.5.5 Odpovědi typu normalizace

Komponenty pro odpovědi typu *normalizace* bylo nutné vytvořit, protože se musí zobrazovat v automatických opravách. Přestože ale zobrazování tohoto typu nebylo nějak změněno, muselo být implementováno ve Vue.

Zobrazení odpovědí typu *normalizace* řeší hned několik komponent. Hlavní komponentou je *NormalizationType*, ta ale zajišťuje pouze vypsání jednotlivých podúloh. Tyto podúlohy pak reprezentuje komponenta *Subtask*.

Obsah každé podúlohy má svou vlastní komponentu, která využívá řadu dalších komponent. Tyto dílčí komponenty vznikly proto, že podúlohy mají často nějaký společný prvek (například výpis množiny funkčních závislostí) a je zde tak možnost stejnou komponentu využít na více místech. Komponenty pro jednotlivé podúlohy jsou umístěny ve složce NormalizationSubtasks a dílčí komponenty, které jsou v podúlohách použity, jsou umístěny ve složce NormalizationSubtasks/parts.

3.6 Zobrazení testů a otázek k opravě

Stránka pro zobrazení tabulky s testy je tvořena komponentou *TestsList*. Tabulka s otázkami z testů je v komponentě *QuestionsList*. Všechny tabulky by měly být co nejvíc jednotné nejen v rámci této práce, ale i celého DBS portálu.

V portálu je pro tabulky využívána komponenta *ublaboo/datagrid*, která je ale určena především pro Nette. Styl tabulek z této komponenty je ale využíván také v dosud existujících částech nového testového modulu, které jsou napsány ve Vue. V komponentách je vytvořena běžná HTML tabulka a na ní a také na okolní elementy jsou aplikovány styly, díky kterým se tabulka

alespoň vizuálně podobá té z komponenty *ublaboo/datagrid*. Takto je řešeno například zobrazení tabulky pro výpis testových šablon v části *TestTemplate* v komponentě *TestTemplateList*.

V naší aplikaci je potřeba mít tabulek více. Vypisovat v každé komponentě celou tabulku se statickým počtem sloupců by bylo neefektivní, protože bychom několikrát psali ten samý kód. Proto byla vytvořena komponenta s názvem *TableList*, která umí tabulku ve stylu *ublaboo/datagrid* vytvářet dynamicky. Komponentě je přes *props* předán jeden objekt s daty. Tento objekt musí obsahovat pole *columns* a pole *rows*.

V *columns* je definice sloupců, které se v tabulce vykreslí. Aby tabulka poskytovala stejné možnosti, jako tabulka v Nette, je možné v definici sloupců nastavit dle kterých sloupců půjde tabulku řadit a také ve kterých půjde vyhledávat. Typ vyhledávací vstupu lze také určit v definici sloupců. Zatím je podporováno vyhledávání přes textové pole, select box a datum. V *rows* je pak pole objektů představující řádky, které se mají do tabulky vypsat.

Komponenta *TableList* zobrazuje předložená data. Nijak je ale neumí řadit ani v nich vyhledávat. Pouze své nadřazené komponentě pošle přes *emit* řetězec obsahující vybrané filtry a očekává, že rodičovská komponenta poskytne aktualizovanou množinu dat. Například na základě změny filtrů si znovu vyžádá data z API.

Rodičovskou komponentou je v tomto případě *QuestionsList*, která ve funkci *fetchQuestions* získá ze serveru všechny otázky z příslušného testu. Zatím se nepočítá s tím, že API bude poskytovat u otázek informaci o počtu neopravených odpovědí. Proto je v této komponentě nutné u každé otázky seskupit neopravené odpovědi pomocí funkce *groupStudentAnswers*. Počet seskupených neopravených odpovědí každé otázky se pak vypíše v tabulce v jednom ze sloupců. Komponenta *QuestionsList* dále obsahuje funkce pro obsluhu filtrů, jejichž aktivaci nám oznámí komponenta *TableList*. Komponenta *TableList* může být v budoucnu dále rozšiřována a vylepšována. Při její implementaci byl kladen důraz na její univerzálnost a může být použita i v jiných částech DBS portálu, které se budou realizovat pomocí Vue.

Podobným způsobem jako je řešena komponenta *QuestionsList*, je realizována také komponenta *TestsList*. Definuje sloupce tabulky pro zobrazování testů, zpracovává údaje o testech z API a seskupuje neopravené odpovědi.

3.7 Automatické opravy

Automatické opravy jsou, stejně jako v současné verzi testového modulu, zvlášť od oprav manuálních a jsou přístupné z vlastní položky v levém menu portálu. Bylo tedy nutné zopakovat proces vytvoření presenteru a konfiguraci Nette routeru stejným způsobem, jako u manuálních oprav. Automatické opravy představují v podstatě jinou aplikaci. Mají vlastní komponentu *App* i soubor *main.js*. Většinu komponent ale sdílí s aplikací pro manuální opravy.

Jediné komponenty, které sdílené nejsou, jsou komponenty s tabulkami testů a otázek. V těchto tabulkách jsou totiž jiné sloupce, než v manuálních opravách. I zde je ale využita komponenta *TableList* a tak realizace těchto dvou stránek pro automatické opravy nebyla složitá.

Stránka pro prohlížení a změnu hodnocení automatických oprav je stejná, jako stránka pro opravu otázky v manuálních opravách. Využívají se zde proto stejné komponenty, které byly v textu popsány dříve.

3.8 Budoucí integrace do testového modulu

Řešení popisované v této práci bude moci být nasazeno až po realizaci dalších částí testového modulu. Nejvíce chybí backendová část pro opravy testů. Nový testový modul bude mít backend realizovaný formou mikroslužeb.

Po vytvoření backendu, který je pro nasazení řešení této práce nezbytný, bude možné aplikaci používat k opravování studentských odpovědí všech typů, které v portálu v současnosti jsou. Nelze vyloučit, že se během vývoje backendu změní struktura poskytovaných dat a že v takovém případě bude nutné upravit také frontendovou část. Stejně tak bude muset být provedena úprava v případě změny způsobu ukládání hodnocení.

V této práci již bylo zmíněno, že seskupování odpovědí, které aktuálně probíhá na frontendu, by bylo lepší provádět na backendu. Tím by se způsob komunikace aplikace se serverem musel upravit. Může také dojít ke změně strategie, kterou jsou otázky seskupovány (popsáno v kapitole 3.5.2.3).

Dále se nabízí možnosti provázání aplikace s jinými částmi testového modulu. Po vytvoření části modulu pro správu zadání tak bude například nutné změnit odkaz pro úpravu zadání na stránce pro hodnocení otázky.

Ve frontendu celého nového testového modulu bude muset být také definován způsob, jakým zjistit právě přihlášeného uživatele. V novém frontendu manuálních oprav potřebujeme tento údaj pro stránky s tabulkami testů a otázek. Aktuálně je uživatel definován staticky v souboru *main.js*.

3.8.1 Struktury dat v API

Tato práce se zabývá tvorbou frontendu a data jí poskytuje mock API. Během realizace tak musel autor práce učinit rozhodnutí, týkající se struktury a způsobu získání dat. V této kapitole je popsán způsob, jak aplikace komunikuje s API a jakou mají data formu.

3.8.1.1 Získání testů (GET)

Data, která se zobrazují v tabulce, ze které si uživatel vybírá test k opravě, získává aplikace když pošle požadavek na */api/tests*.

V odpovědi aplikace obdrží testy a jejich data potřebná pro zobrazení hodnot v tabulce. V této odpovědi tedy každý test musí mít:

- id,
- name – název testu,
- start_time – čas spuštění testu,
- end_time – čas ukončení testu,
- status – stav testu,
- templateAuthor – autor šablony,
- testAuthor – autor testu,
- questions – otázky z testu.

Autoři jsou posláni jako objekt, který obsahuje *id*, *username*, *first_name* a *last_name*. V *questions* je pole, ve kterém jsou objekty reprezentující všechny otázky z testu. Každá tato otázka obsahuje atribut *uncorrectedAnswers*, což je pole objektů neopravených otázek. Všechny neopravené otázky se při získávání testů vrací proto, že v tabulce s testy je nutné zobrazit jak celkový počet neopravených odpovědí, tak také kolik zbývá opravit právě přihlášenému uživateli. Tyto odpovědi je potřeba nejprve seskupit, protože chceme zobrazovat počet unikátních odpovědí. To se aktuálně provádí na frontendu.

Několikrát v této práci bylo zmíněno, že seskupování by měl v budoucnu provádět backend. Až tomu tak bude, bude možné v této fázi posílat počet neopravených odpovědí ze serveru pouze jako číslo. Tím se proces zpracování na straně frontendu zjednoduší.

Tabulka s testy u automatických oprav vyžaduje stejná data, akorát u otázek má místo všech neopravených otázek položku *autocorrectedAnswers*, která obsahuje počet všech automaticky opravených otázek. Automatické i manuální opravy obsluhuje v rámci mock API ten samý *endpoint*, na který se posílá u automatických oprav query parametr *autocorrected* s hodnotou *true*, tedy: */api/tests?autocorrected=true*.

Na API se posílá požadavek vždy, když je změněn filtr v tabulce. V takovém případě by měl backend odpovědi filtrovat nebo řadit. Nastavení filtru je odesláno v požadavku jako query parametr. Požadavek se tedy odesílá například na */api/tests?jmenoFilteru=hodnota1&dalsiFilter=hodnota2*. Při nastavení řazení tabulky dle nějakého sloupce se posílá, podle jakého sloupce se řadí (parametr *sortBy*), a směr řazení (parametr *sortOrder*). Názvy parametrů při vyhledávání odpovídají názvům sloupců v tabulce tak, jak jsou definovány v komponentě *TestsList*. Hodnotou tohoto parametru je obsah příslušného filtru. Tedy požadavek s vyhledáním testů, které v názvu obsahují slovo „semestr“ by vypadal takto: */api/tests?name=semestr*.

3.8.1.2 Získání otázek (GET)

Získávání dat pro tabulku s otázkami je podobné jako v případě testů. Otázky se na stránce vypisují vždy pro vybraný test. Do požadavku je nutné uvést ID tohoto testu. Požadavek se posílá na */api/tests/{ID testu}/questions*.

Odpověď ze serveru by měla obsahovat položky *questions* a *test*. Test může mít pouze atribut *name*, protože slouží jen pro zobrazení jména testu v nadpisu nad tabulkou. Atribut *questions* obsahuje pole s otázkami. Každá otázka musí mít tyto atributy:

- id,
- name – název otázky,
- answer_type – typ odpovědi,
- author – autor otázky,
- corrector – přiřazený opravující,
- uncorrectedAnswers – neopravené odpovědi.

Objekty *author* a *corrector* mají stejnou strukturu jako autoři u testů. Také význam pole s neopravenými odpověďmi *uncorrectedAnswers* je stejný jako u předchozí tabulky. Zde je potřeba spočítat počet neopravených seskupených odpovědí. Kdyby se seskupování provádělo na serveru, stačilo by v odpovědi vracet jedno číslo. Jelikož jsou všechny tabulky realizovány komponentou *TableList*, platí zde pro filtry stejná pravidla jako u tabulky s testy.

V automatických opravách nemáme tentokrát ani neopravené odpovědi, ani automaticky opravené odpovědi, ale všechny odpovědi. V tabulce s výpisem otázek se totiž vypisují hned tři údaje: počet všech odpovědí, automaticky opravené odpovědi a odpovědi, které jsou automaticky ohodnoceny za 0 bodů. Filtrování automatických odpovědí z tohoto pole probíhá až při zpracování na frontendu. Opět je zde poměrně složitý proces seskupování všech zaslaných odpovědí jen proto, aby byl získán jejich počet. V případě seskupování odpovědí na serveru by stačilo z něj poslat tato tři zmíněná čísla.

3.8.1.3 Získání otázky a odpovědí pro opravování (GET)

Data pro stránku s hodnocením otázky jsou získávána pomocí tří různých požadavků na server: získání údajů o otázce, získání referenčních odpovědí a získání studentských odpovědí. V adrese je vždy obsaženo ID testu i ID opravované otázky. Údaje k otázce jsou získány na */api/tests/{ID testu}/questions/{ID otázky}*.

V odpovědi musí mít každá otázka tyto atributy:

- id,
- name – název otázky,

- answer_type – typ odpovědi,
- assignments – zadání,
- task – úkol,
- max_points – maximum bodů,
- subtasks – podúlohy (pouze u typu *normalizace*).

Atribut *answer_type* nabývá vždy některé z těchto hodnot: *Text*, *Checkbox*, *Radio*, *Diagram*, *SQL*, *RA*, *Transformation*, *Normalization*. Atribut *subtasks*, který má pouze otázka typu *normalizace* je pole s objekty, které reprezentují jednotlivé podúlohy. Tyto objekty obsahují atributy *subtask_type* s typem podúlohy a *solution*, kde se nachází ukázkové řešení dané podúlohy. Atribut *subtask_type* může být: *Canonical Coverage*, *Transitive Closure*, *Find Keys* nebo *Conversion*.

Pole se zadáními je v atributu *assignments*. Struktura jednoho zadání vypadá takto:

- id,
- type – typ zadání,
- name – název zadání,
- order – pořadí zadání v rámci otázky,
- data – obsah zadání,
- max_points – maximum bodů,
- db_connection_id – ID databázového připojení (pouze u typů *SQL* a *RA*).

Možné hodnoty u typu zadání jsou: *text*, *image*, *diagram*, *normalization* a *database*.

Pro získání referenčních odpovědí k opravované otázce se posílá požadavek na */api/tests/{ID testu}/questions/{ID otázky}/reference-answers*. V případě odpovědí studentů je to */api/tests/{ID testu}/questions/{ID otázky}/answers*.

Odpovědi mají následující atributy:

- id,
- reference – zda je odpověď referenční,
- corrected – zda je odpověď opravená,
- autocorrected – zda byla odpověď opravena automaticky,
- earnedPoints – za kolik bodů byla odpověď ohodnocena (*null* pokud nebyla),
- comment – komentář u ohodnocené odpovědi (*null* pokud nebyla ohodnocena).

Tyto atributy nejsou potřeba u referenčních odpovědí. V jejich nás zajímá pouze jejich obsah. Tento obsah se pak liší dle typu odpovědi.

Typ *text* obsahuje text s odpovědí v atributu *answerOriginal*. Odpověď typu *checkbox* má odpověď uloženou v poli *choices*, které obsahuje objekty s dvojicí atributů *answer* (text u volby k zaškrtnutí) a *checked* (to, zda je odpověď zaškrtnuta). Stejně je vyřešena také odpověď typu *radio list*.

Serializovaný objekt s diagramem ve formátu JSON se může nacházet buď v atributu *answerTransformed* nebo *answerOriginal*. Odpovědi typu *SQL* a *RA* mohou být buď původním tvaru v atributu *answerOriginal*, nebo v normalizovaném tvaru v atributu *answerNormalized*. Obsah odpovědi typu *transformace* je uložen v *answerOriginal*.

Odpovědi typu normalizace obsahují pole jménem *subtasks*. Toto pole obsahuje objekty s těmito atributy:

- subtask_type – typ podúlohy,
- solution – řešení podúlohy,
- correct – zda je řešení správně.

3.8.1.4 Provedení dotazu (POST)

Vykonání dotazů v SQL nebo relační algebře nad databází se provádí metodou POST. V požadavku se pošle dotaz a v odpovědi se vrátí výsledek dotazu, případně chybová hláška. Provedení dotazu se posílá na */api/execute* a obsah požadavku vypadá takto:

- connection_id – ID databázového připojení,
- ra – dotaz v relační algebře (může být prázdný, pokud je dotaz v SQL),
- sql – dotaz v SQL (může být prázdný, pokud je dotaz v relační algebře).

Odpověď serveru má pak tuto strukturu:

- columns – sloupce výsledku dotazu,
- rows – řádky výsledku dotazu,
- query – provedený dotaz v SQL,
- error – chybová hláška,
- translationError – chybová hláška z překladu relační algebry.

Aktuálně jsou v mock API definovány u tohoto endpointu dvě statické odpovědi. Při předložení jakéhokoliv dotazu nad jakýmkoliv připojení se tak vrátí vždy jedna z těchto dvou odpovědí. Je to z toho důvodu, že zatím nelze snadno vytvořit připojení, které by se vázalo ke konkrétní otázce.

3.8.1.5 Změna hodnocení odpovědi (PATCH)

Při odesílání hodnocení studentské odpovědi dochází k její modifikaci. Proto se používá metoda PATCH. Pomocí této metody se komunikuje s endpointem */api/answers/{ID odpovědi}*, kde dojde ke změně některých atributů odpovědi. Atributy odpovědi, které se při hodnocení mohou měnit jsou: *corrected*, *earnedPoints*, *comment* a také *reference* v případě přidání odpovědi mezi referenční (či odebrání).

Kapitola 4

Testování

Prošli jsme si analýzou, kde jsme se seznámili s aktuálním stavem manuálních oprav a specifikovali si požadavky. Dále jsme vytvořili návrh, kde se objevily inovované i úplně nové prvky a funkce. Nakonec jsme si popsali také samotnou realizaci a některé aspekty nové verze jsme tak rozebrali ještě detailněji.

Dá se tedy předpokládat, že v této fázi umíme nový frontend používat. Po koncových uživatelích ale samozřejmě nemůžeme chtít, aby si před prvním opravováním testů v nové verzi nejprve přečetli tuto práci a teprve poté byli „proškoleni“ k jejímu používání. Systém musí být sám o sobě intuitivní a uživatel musí být schopen funkce v něm sám najít a správně používat.

Návrh výsledného produktu je v řadě věcí inspirován současným řešením. Můžeme se tedy domnívat, že koncoví uživatelé se v základních věcech rychle zorientují. Byly ale přidány také nové funkce (našeptávání komentářů, vyhledávání otázek atd.), které by měly vyučujícím opravu usnadnit. Musíme se přesvědčit, že koncoví uživatelé rozumí způsobu, jakým byly nové funkce realizovány, a jestli jim jejich forma vyhovuje. I přesto, že návrh systému se na první pohled podobá předešlé verzi, vývoj probíhal zcela od nuly. Je proto také nutné zjistit, zda nebyla odstraněna nebo nežádoucím způsobem změněna nějaká funkce, která je pro vyučující při opravě podstatná.

Z těchto důvodů proběhne test použitelnosti s koncovými uživateli. V této kapitole si nejprve popíšeme problematiku uživatelského testování po teoretické stránce, abychom věděli, jak takové testování provést. Dále zde bude popsána již samotná příprava, průběh a zhodnocení testování frontendu manuálních oprav.

4.1 Testování použitelnosti

Použitelnost je vlastnost (nejen) webových stránek, která vyjadřuje snadnost jejich používání. Pod tento pojem je zahrnuto to, jakým způsobem se noví uživatelé se systémem učí pracovat, jak intuitivní pro ně je jejich používání

a jak jsou ve své práci s efektivní [19]. Dobrá použitelnost se vyznačuje tím, že uživatel dokáže v rozumném čase na webu udělat přesně to, co chce. Pokud má web dobrou použitelnost, uživatelé se na web raději vrací a tyto weby pak bývají úspěšnější [20].

Testování použitelnosti probíhá přímo s cílovým uživatelem systému. Cílem tohoto testování je nalézt nedostatky, kterých si sami vývojáři stránek nevšimli. Příkladem těchto nedostatků můžou být například špatně nazvané prvky na stránce, nestandardní chování prvků, problém nalézt nějakou informaci a tak dále [21].

4.1.1 Persony

Aby bylo testování použitelnosti přínosné, je třeba jej dobře naplánovat. Při pozdějším vytváření scénářů a úkolů nám velmi pomůže když budeme přesně vědět, kdo je cílovým uživatelem testovaného systému a tedy jací lidé se budou testování účastnit. Z tohoto důvodu se vytváří tzv. persony [22]. Jejich smyslem je vytvořit sice imaginární, ale typické uživatele systému. Do těchto person je poté možné projektovat naše představy o tom, jaké mají naši budoucí uživatelé od systému očekávání a jak budou systém používat. Persony se vytváří například na základě kvalitativního či kvantitativního výzkumu [23].

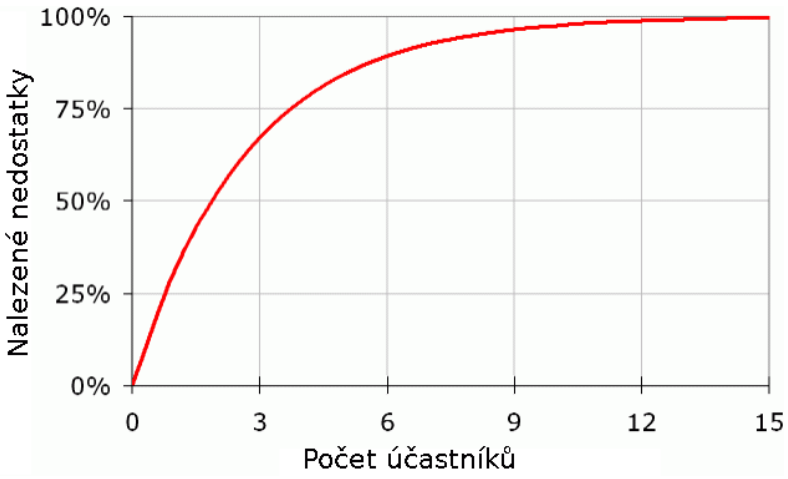
4.1.2 Počet účastníků testování

Otázka, kterou si položí asi každý, kdo připravuje testování použitelnosti je, kolik uživatelů by se mělo testování zúčastnit. Mohlo by se zdát, že čím více účastníků se testování zúčastní, tím více problémů s designem naší aplikace nalezeme. Ve skutečnosti ale počet uživatelů může být poměrně malý.

Dle Jakoba Nielsena [24] není třeba provádět testování s větším počtem uživatelů, než je pět. Je to z toho důvodu, že nejvíce problémů identifikujeme vždy s prvním uživatelem. Problémy zjištěné při testování s druhým uživatelem se pak s části překrývají s těmi, o kterých víme již z prvního testování. Čím víc testů s různými uživateli provádíme, tím méně nových informací se tak dozvídáme. Testování s větším množství uživatelů je proto zbytečné. Graf zachycující tuto skutečnost je na obrázku 4.1.

4.1.3 Scénáře a úkoly

Při provádění testování s uživateli hrají klíčovou roli scénáře. Ty popisují nějakou realistickou situaci, při které má účastník testování postupně plnit seznam úkolů. Osoba, která testování provádí (tester) uživatele při této činnosti pozoruje a zapisuje si poznámky. Velmi pomáhá, když uživatel při testování „přemýšlí nahlas“ a tester pak má lepší představu o tom, jak uživatel testovaný systém chápe. Příkladem těchto scénářů může být například vytvoření účtu v systému či objednání nějakého zboží. [25]



Obrázek 4.1: Počet účastníků testování a nalezené nedostatky [24]

Úkoly je nutné vytvářet tak, aby nebyly příliš návodné. To znamená, že znění úkolu by nemělo obsahovat příliš velkou nápovědu, jak něco vyřešit. Pokud bychom totiž uživateli zadáváním úkolu napovídali, dostaneme zkreslenou představu o tom, jak by se choval, kdyby činnost zadanou v úkolu potřeboval splnit ve skutečnosti. Je proto vhodné vyvarovat se přímému navádění na nějaký prvek na stránce (např. název tlačítka). [26]

4.2 Příprava na testování

Prošli jsme si teoretickou přípravou na testování použitelnosti a na základě ní se nyní připravíme na vlastní testování nového frontendu manuálních oprav. Jak již víme, je klíčové testování dobře připravit, abychom zjistili co nejvíce informací a odhalili co nejvíce problémů aplikace.

4.2.1 Popis persony

Nejprve si popíšeme běžného uživatele testovaného produktu. Jsou to cvičící předmětu BI–DBS, což je poměrně úzká a specifická skupina. Definovat tedy budeme pouze jednu personu, protože se nepředpokládá, že jiní, méně informovaní uživatelé, by systém používali. Vycházet budeme ze znalostí získaných při rozhovorech s cvičícími, které proběhly v rámci analýzy.

Je pravděpodobné, že většina cvičících, kteří budou nový testový modul z počátku využívat, bude mít zkušenosti se stávající verzí. Postupně ale budou přibývat uživatelé, u kterých toto neplatí a tak je potřeba dbát na to, aby byl systém snadno použitelný také pro ně.

Persona, popsána v této části, se bude využívat při očekávaném chování uživatele při průchodu scénářem. Persona bude označována jako „cvičící“.

Cvičící je technicky vzdělaný uživatel, který často využívá i jiných informačních systémů. Vyučuje předmět BI–DBS a rozumí tak pojmům, které se v předmětu testují. Chápe význam všech typů odpovědí a je seznámen s názvoslovím, které portál používá při popisu otázek (zadání, úkol, referenční odpověď). Předtím, než cvičící poprvé otázky opravuje, ve většině případů je v jiné části testového modulu vytváří. Cvičící se rád učí ovládat nové prvky systému, pokud pro něj znamenají, že mu zefektivní jeho pracovní proces. V případě nejasností s fungováním DBS portálu má cvičící možnost obrátit se na své kolegy či vývojáře portálu.

4.2.2 Získání vzorových dat

Abychom mohli provést plnohodnotné testování, je třeba získat vzorová data, která jsou více realistická než ta, se kterými probíhal vývoj. S reálnými daty z portálu bude tvorba scénáře snazší. Také pro cvičící, se kterými bude testování prováděno, bude celý proces jednodušší, protože si budou moci snáze představit, že opravují skutečné odpovědi.

Z tohoto důvodu autor této práce kontaktoval Ing. Oldřicha Malce, který má přístup do databáze, kterou používá produkční verze DBS portálu. Cílem bylo především získat skutečné otázky a studentské odpovědi k nim. Jelikož jsou ale tabulky v současné databázi velmi úzce provázány bylo možné poskytnout pouze data (dump) z celého testového modulu. Dump z databáze byl ve formě souboru s SQL příkazy, především CREATE a INSERT. Tímto způsobem se tak do souboru uložil aktuální stav databáze a provedením příkazu ze souboru by bylo možné celou databázi, včetně dat, rekonstruovat.

Soubor ale obsahoval citlivé údaje uživatelů portálu – jména a příjmení. S těmito údaji nemohl být autorovi této práce poskytnut. Proto bylo s Oldřichem Malcem dohodnuto, že autor této práce vytvoří skript pro anonymizaci těchto údajů. Autorovi práce bylo jasně zadáno, jak má skript fungovat.

Skript byl napsán v jazyku bash. Funguje tak, že na vstupu obdrží soubor, určený k anonymizaci, a konfigurační soubor. Konfigurační soubor musí mít na každém řádku dva řetězce oddělené čárkou. První řetězec je hodnota, která se má nahradit (typicky citlivý údaj), druhým řetězcem je pak řetězec, kterým se bude nahrazovat. Výstupem skriptu je pak nový soubor, který odpovídá souboru vstupnímu, akorát v něm jsou nahrazeny všechny řetězce, které byly uvedeny v konfiguračním souboru. Celý skript na anonymizaci souboru je v příloze B.1.

Z výsledného anonymizovného souboru poté autor práce vybral otázky, zadání, referenční odpovědi a odpovědi studentů, které použil pro testování.

4.2.3 Příprava otázek

Cílem přípravy bylo vytvořit takový scénář, který uživatele přiměje vyzkoušet všechny významné prvky nového frontendu. Dále bylo třeba otestovat, jak se

budou uživatelé chovat při opravování odpovědí konkrétních typů. V konfiguraci serveru Mirage.js byl pro tyto účely vytvořen test jménem „testovani“, do kterého byly přidány některé otázky a odpovědi, pocházející z reálných dat.

Testování se bude soustředit výhradně na stránku pro hodnocení otázky, jelikož na stránkách s tabulkami nebyla provedena žádná výrazná změna a ani cvičící při sběru požadavků nepoukazovali na žádné nedostatky. Scénář bude stavěn na výchozí rozvržení této stránky (otázka vlevo, odpovědi vpravo), protože i na základě rozhovorů s cvičícími se předpokládá, že bude v praxi využíváno v naprosté většině případů.

Do testování budou zařazeny pouze ty typy odpovědí, které jsou uživatelé zvyklí manuálně opravovat a se kterými se budou setkávat i v novém modulu. Oprava otázek s odpověďmi typu *normalizace* a *radio list* je navíc velice podobná současnému řešení. Dále nebyla do testování zařazena odpověď typu *RA*, protože je téměř totožná s opravou typu *SQL* a o to víc se autor práce mohl soustředit na přípravu testování opravy tohoto typu odpovědi.

4.2.3.1 Příprava testování otázky s odpovědí typu SQL

U otázek s odpovědí typu *SQL* je pro jejich plné fungování potřeba databázové připojení, které obsahuje schéma definované v zadání otázky. Dotazy v odpovědích se nad tímto připojením provádí. Nad stejným připojením pracuje také playground. Nový backend, který by umožňoval dotazy provádět nad databází přímo spojenou s otázkou ale ještě není implementován. Bylo tak nutné si při přípravě otázky typu *SQL* vystačit se současnými možnostmi DBS portálu.

Při přípravě části scénáře, věnující se této otázce, byla nejprve z reálných dat vybrána vhodná otázka. Tato otázka měla zadání, které obsahovalo popis relačního schématu. Podle tohoto popisu vytvořil autor práce SQL skript, který v relační databázi vytváří tabulky a definuje vztahy mezi nimi (create script). Při tvorbě tohoto skriptu byly použity nástroje *data modeller* a *transformace modeller*, které poskytuje DBS portál. Vytvořený skript je uveden v příloze B.2. Dále se autor v produkční verzi DBS portálu připojil k databázi, kterou má jako student FIT k dispozici, a spustil nad touto databází vytvořený skript. Tímto byla připravena databáze se strukturou, která odpovídala zadání otázky.

Do této databáze byla poté pomocí autorem vytvořeného skriptu vložena vzorová data (insert script). Dále bylo připojení ke zmíněné databázi nakonfigurováno také v instanci DBS portálu, kde bude probíhat testování. Tímto způsobem tak bylo docíleno toho, že při opravování otázek typu SQL bude možné využívat naplno playground a výsledek, který bude vracet, bude odpovídat zadanému dotazu.

4.2.3.2 První verze scénáře

Při přípravě scénáře byla nejprve vytvořena první verze, ta byla poté otestována s vedoucím práce a následně byl scénář v některých bodech upraven. Už u první verze scénáře se počítalo s tím, že se bude postupně procházet pět připravených otázek k opravě, kde každá bude mít odpověď jiného typu.

Pro každý typ odpovědi byla vytvořena série úkolů, ve kterých probíhalo testování různých, pro daný typ specifických, ale i obecných aspektů nového frontendu. V tomto textu se budeme detailněji zabývat až popisem scénáře, který byl použit v ostrém testování. Celá první verze scénáře je uvedena v příloze C.1.

4.2.4 Testování s vedoucím práce

První testování aplikace proběhlo s vedoucím této práce Ing. Jiřím Hunkou. Jednalo se v podstatě o zkoušku toho, zda připravený scénář plní svou roli. Vedoucí práce je ale zároveň také cvičícím předmětu BI–DBS a uživatelem DBS portálu a tak se výstupy z tohoto testování využijí pro zlepšení výsledného řešení. Už při tomto testování totiž vyšlo najevo několik nedostatků nového uživatelského rozhraní. V této části textu jsou tyto nedostatky popsány.

Testování probíhalo během osobní konzultace na notebooku autora této práce. Průběh testování byl pomocí programu OBS studio zaznamenán na video (záběr obrazovky a z webkamery). Záznam z testování se nachází na přiloženém médiu ve složce testing/recordings/testing_0. Vedoucí práce plnil úkoly dle první verze scénáře (příloha C.1).

4.2.4.1 Nalezené nedostatky

První nedostatek se objevil už při hodnocení první odpovědi, kdy vedoucímu práce nebylo na první pohled jasné, jaký je maximální počet bodů za otázku. V tomto případě se jednalo o opomenutí autora práce při implementaci, protože návrh informaci o maximálním počtu bodů obsahoval. Pro ostré testování tak byl tento nedostatek odstraněn.

Vedoucí práce nevyužil možnosti kliknout na návrh bodů pod odpovědí typu checkbox a body přepisoval. Uvedl, že by bylo lepší mít body předvyplněné rovnou v poli pro zadávání bodů. Stejně tak by se měly rovnou vyplňovat i body navržené na základě vyplněných komentářů.

Během testování se ukázalo, že u našeptávače komentářů by bylo vhodné mít možnost zobrazit všechny možnosti, které našeptávač nabízí a nepožadovat po uživateli zadávat do komentáře nějaký text. Uživatel si totiž nemusí přesně pamatovat jakou formulaci komentáře během dřívějšího opravování zvolil, ale rád by znovu napsal stejný komentář.

Vedoucí práce dále uvedl, že informační okno, které se zobrazí po přidání odpovědi mezi referenční by mělo obsahovat možnost toto přidání vzít zpět. Stejně tak by mělo být možné přidat mezi referenční odpovědi právě ohodnoce-

nou odpověď za plný počet bodů, u které ale uživatel kliknul pouze na tlačítko „Ohodnotit“.

Dále při testování práce s více referenčními odpověďmi nebylo vedoucímu práce jasné, která odpověď byla přidána jako poslední. Vedoucí také zmínil, že po kliknutí na tlačítko pro vyhledání referenční odpovědi mezi studentskými by se měl box obsahující tuto odpověď rozbalit.

V průběhu testování se vedoucí pozastavil nad možnostmi odstraňování referenčních odpovědí, které nejsou v rámci probíhající opravy mezi studentskými odpověďmi. Odstraňování všech referenčních odpovědí by mělo být umožněno v části stránky nového testového modulu, která je určena pro správu otázek a odpovědí.

Další nedostatek se projevil při testování přesouvání boxů na levé straně stránky. Vedoucí práce sice věděl, že boxy jdou přesouvat, ale jen proto, že o této možnosti byl informován již dříve. U boxů tedy chybí označení, že je možné je přesouvat.

Co se týče oprav odpovědí typu *diagram*, tak zde vedoucí práce nevyužil možnosti kliknutí na odkaz pod diagramem ke zkopírování opravy a komentářů z diagramu do pole pro komentář. Tohoto odkazu si totiž při testování nevšiml. Dále se u více načtených diagramů ukázalo, že plocha *Kreslítka*, ve které je diagram vykreslen je o dost větší, než samotný diagram v ní. Každý diagram tak na stránce zabírá o dost víc místa, než potřebuje.

Při testování vyhledávání v již opravených odpovědích, začal vedoucí nejprve rozbalovat již opravené odpovědi a až po čase si všiml vyhledávacího pole. Uvedl, že by pole mělo být označeno ikonou lupy. Dále vedoucímu na stránce chyběla informace, že je filtr aktivní.

U otázky typu *SQL* bylo testováno, zda uživatel nalezne playground a pochopí jeho ovládání. Také zde byl test ovlivněn tím, že vedoucí práce věděl, že řešení tuto funkci obsahuje. Uvedl ale, že by stránka měla uživatele nějak na playground upozornit. Při hodnocení a následném stahování odpovědí se také projevil nedostatek, že hodnocení každé odpovědi typu *SQL* trvalo poměrně dlouho. Pokaždé se totiž znovu na serveru prováděly všechny dotazy.

Posledním testovaným typem odpovědi byla *transformace*. Opravování tohoto typu bylo nedávno upraveno ještě v původní verzi testového modulu. Některá vylepšení, která tato úprava přinesla nebyly zapracovány do návrhu nového frontendu a ani zde nebyly implementovány. Vedoucí práce uvedl, že mu v testovaném řešení chybí předvyplněné body a seznam rozdílů mezi studentskou a referenční odpovědí, který je lepší, než diff zobrazený v testovaném řešení.

4.2.4.2 Realizované změny

Testování s vedoucím této práce poskytlo autorovi velmi cennou zpětnou vazbu na nové řešení frontendu manuálních oprav. Ukázalo se, že některé nalezené nedostatky bude snadné odstranit. Proto se autor práce rozhodl, že provede

některé změny v řešení ještě předtím, než proběhne testování s dalšími cvičícími.

U návrhu bodů, které se zobrazují u odpovědí typu *checkbox* nebo v rámci našeptávače komentářů byl změněn způsob jejich zobrazení. Nově se návrh bodů vkládá přímo do pole pro zadávání bodů a není tak nutné klikat na žádný odkaz. Pokud má uživatel aktivní kurzor na prázdném řádku v poli pro psaní komentáře, může nově pomocí kláves *ctrl+space* zobrazit seznam všech dříve zadaných komentářů. Není tak nutné pro aktivaci našeptávače psát začátek komentáře.

Po kliknutí na vyhledání referenční odpovědi ve studentských odpovědích se na pravé straně nejen zobrazí box s danou odpovědí, ale také se tento box rozbalí. U boxů s údaji o otázce byla dále přidána do levého horního rohu ikona značící, že boxy je možné přesouvat. Zda je toto řešení dostačující se ukáže při dalším testování.

Po testování s vedoucím práce byla také upravena funkce pro zobrazování diagramů. Nově se výška plochy, do které se vykresluje diagram přizpůsobuje velikosti diagramu. Přesný popis, jak byl tento nedostatek vyřešen byl popsán již v kapitole věnující se realizaci (3.5.5.2).

Další dvě drobné změny jsou cíleny na lepší orientaci uživatele po stránce. Jedná se ikonku lupy u pole pro vyhledávání mezi opravenými odpověďmi a upozornění na playground. To je realizováno pomocí tooltipu, který se zobrazí u tlačítka „Playground“ nad dotazem, když uživatel na tento dotaz dvakrát klikne myší. Zde totiž autor práce předpokládá, že pokud uživatel bude chtít dotaz upravit, nejprve bude klikat přímo na něj. Pokud tak učiní, stránka ho tooltipem upozorní na tlačítko.

Další změna se týkala optimalizace provádění dotazů u odpovědí typu *SQL* a *RA*. Výsledky provedených dotazů si nyní systém ukládá. V případě, že je výsledek dotazu v budoucnu požadován znovu, neposílá se požadavek na server, ale použije se dříve uložený výsledek. Díky této optimalizaci se zkrátilo čekání na ohodnocení a opětovné načtení odpovědí typu *SQL* a *RA*.

Další nedostatky, které byly identifikovány během testování s vedoucím práce, byly ponechány i na další testování. Důvodem byly jak časové možnosti, tak také to, že si autor práce chtěl ověřit, jak se tyto nedostatky projeví v testování s dalšími uživateli.

4.2.5 Finální scénář

Kromě úpravy v samotném kódu nového frontendu manuálních oprav, doznal menších změn také scénář. V této kapitole si uvedeme a popíšeme jednotlivé části a úkoly scénáře, který byl použit při ostrém testování. U jednotlivých úkolů je nejprve tučným písmem uvedeno jeho znění, přibližně tak, jak bylo zadáno účastníkům testování. Pod zněním úkolu je uveden jeho bližší popis, význam a předpokládaný postup cvičícího. Úkoly jsou očíslovány a na některé se bude tento text v dalších kapitolách odkazovat.

Scénář začíná na stránce s tabulkou otázek z testu, které se budou během testu procházet. Je rozdělen do částí dle právě procházené hodnocené otázky. Začátek každé části obsahoval úkol, který účastníka testování vyzval k tomu, aby z tabulky s otázkami vybral konkrétní otázku a seznámil se s jejím zadáním, úkolem a referenční odpovědí.

4.2.5.1 Typ checkbox

1) Opravte první odpověď.

Úvodní úkol, jehož smyslem je zjistit, jak se cvičící na stránce orientuje, zda pochopí způsob vyznačení správně a chybně zaškrtnutých možností a zda porozumí návrhu bodů, který mu systém předkládá.

Předpokládaný postup: Cvičící si prohlédne odpověď, zorientuje se v možnostech, které zaškrtnul student a porovná je s referenční odpovědí. Poté zváží, zda souhlasí s navrhovaným počet bodů. Pokud ano, odpověď ohodnotí, pokud ne, před hodnocením body změní.

2) Sedm studentů zvolilo při odpovídání na otázku stejné možnosti. Jedním hodnocením opravte odpověď všem těmto studentům.

Smyslem tohoto úkolu je vyzkoušet, zda cvičící porozumí tomu, že stejné odpovědi jsou nyní seskupeny do jedné a že se v pravém horním rohu boxu s odpovědí nachází údaj o celkovém počtu otázek.

Předpokládaný postup: Cvičící najde mezi neopravenými odpověďmi tu, kde se v pravém horní rohu nachází údaj „Počet odpovědí: 7“ a odpověď ohodnotí.

3) Vraťte se na stránku se všemi otázkami z testu.

Tento úkol se nachází na konci každé části s otázkami a její postup je vždy stejný (popsán je proto pouze zde). Cvičící se vrátí zpět na stránku s tabulkou se všemi otázkami, aby mohl přejít k opravě další otázky.

Předpokládaný postup: Cvičící klikne na tlačítko „Zpět“ v pravém rohu stránky.

4.2.5.2 Typ text

1) Opravte první tři neopravené odpovědi. Pokud neudělujete plný počet bodů, uveďte důvod do komentáře.

Žádná z těchto tří studentských odpovědí není správně. První a třetí odpověď má stejný typ chyby. Úkol testuje to, jak cvičící bude využívat našeptávač komentářů.

Předpokládaný postup: Cvičící opraví první dvě odpovědi. V souladu se zadáním za ně udělí počet bodů nižší než maximální a uvede komentář s popisem chyby. Při opravování třetí odpovědi bude chtít uvést stejný komentář, jako v případě první odpovědi. Buď otevře našeptavač se staršími komentáři pomocí *ctrl+space*, nebo začne psát znovu komentář, který napsal u první odpovědi. V našeptavači pak zvolí vhodný komentář na doplnění a následně odpověď ohodnotí.

2) Opravte další neopravenou odpověď a zařaďte ji mezi referenční odpovědi.

Předpokládaný postup: Cvičící udělí odpovědi plný počet bodů a přidá klikne na tlačítko „Ohodnotit a přidat mezi referenční odpovědi“.

3) Dále zařiďte, aby se tato odpověď zobrazovala na levé straně, aniž by byly zobrazeny ostatní referenční odpovědi.

Úkol testuje to, jak se cvičící zorientuje v práci s více referenčními odpověďmi. Cílem je nahradit na levé straně zobrazenou referenční odpověď tou, která byla mezi referenční přidána v minulém úkolu.

Předpokládaný postup: Cvičící klikne v boxu s referenční odpovědí na možnost „Zobrazit ostatní“ a ve více referenčních odpovědích nalezne tu, kterou sem přidal v minulém úkolu. Poté v pravém horním rohu této odpovědi klikne na ikonu hvězdičky, čímž ji přesune na první místo v seznamu referenčních odpovědí. Následně kliknutím na možnost „Skrýt ostatní“ skryje všechny referenční odpovědi, kromě té na prvním místě.

4) Zjistil jste, že se Vám někdo snažil pomoci a opravil již dříve část odpovědí za Vás. Udělal ale chybu a označil chybnou odpověď za referenční. V této odpovědi student dokonce sám uvedl „*Nazvy nevim, hadam*“. Najděte tuto odpověď a odstraňte ji z referenčních odpovědí, aniž byste opustili stránku s hodnocením otázky.

Úkol testuje, jakým způsobem cvičící zmíněnou odpověď vyhledá a zda dokáže najít možnost, jak ji odstranit z referenčních odpovědí.

Předpokládaný postup: V případě vyhledání odpovědí se nabízí dva rychlé způsoby řešení. První je, že cvičící si zobrazí všechny referenční odpovědi a bude v nich hledat odpověď obsahující zmíněná slova. Po jejím nalezení klikne v pravém horním rohu této odpovědi na možnost „Najít ve studentských odpovědích“. Druhý způsob, jak najít danou odpověď, je využít vyhledávání mezi opravenými odpověďmi. Poté cvičící u nalezené odpovědi přejede na tlačítko „Zařazeno mezi referenční odpovědi“, jehož text se po najetí myší změní na „Odstranit z referenčních odpovědí“, a klikne na něj.

4.2.5.3 Typ diagram

1) Zařiďte, abyste na levé straně viděl zadání i referenční odpověď bez nutnosti levou stranou skrolovat.

Úkol testuje, jestli uživatel odhalí možnost přesouvání boxů na levé straně stránky. Pro to, aby ho splnil musí přesunout box s úkolem tak, aby nebyl mezi boxem se zadáním a boxem s referenční odpovědí.

Předpokládaný postup: Cvičící uchopí kurzorem myši box s úkolem a přesune ho na první nebo poslední pozici. Dále posune scrollbar tak, aby viděl zadání i referenční odpověď (případně může ještě uspořádat diagram, aby se na obrazovku vešel celý).

2) Napište komentář k první odpovědi, kde vypište a popište chyby, které v diagramu označila automatická oprava. Odpověď ohodnoťte.

Tento úkol zjišťuje, zda cvičící píše komentáře u odpovědi typu *diagram* přímo do *Kreslítka* a pokud ano, tak jestli poté využije odkazu pod diagramem ke vložení opravy z *Kreslítka* do pole komentáře.

Předpokládaný postup: Uživatel dle svého uvážení napíše pomocí režimu oprav v *Kreslítku* komentáře k vyznačeným chybám. Poté klikne na odkaz pod diagramem a tím výpis chyb vloží do pole komentáře.

3) Prohlédněte si další odpověď. Máte pocit, že řešení ve kterém student použil entitu *Wishlist* jste už dříve viděl a ohodnotil. Na první pohled se vám zdá odpověď stejná a chcete se podívat, v čem se liší. Najděte dříve ohodnocenou odpověď s entitou *Wishlist* a zjistěte rozdíl.

Otázka typu *diagram* obsahuje už na začátku více opravených odpovědí. Jedna z těchto odpovědí je podobná právě opravované odpovědi. Cílem je dříve opravenou odpověď najít. Úkol testuje, zda uživatel použije vyhledávací pole v horní části stránky.

Předpokládaný postup: Cvičící zadá do vyhledávacího pole slovo „wishlist“ a zobrazí si tak mezi opravenými odpověďmi pouze jednu odpověď. Tuto odpověď rozbalí.

4) Zařiďte, abyste na pravé straně stránky viděl všechny dříve opravené odpovědi.

Cílem úkolu je zrušit aktivní filtr z minulého úkolu a zobrazit opět všechny odpovědi.

Předpokládaný postup: Cvičící klikne na křížek ve vyhledávacím poli, případně z něj sám vymaže zadaný řetězec.

4.2.5.4 Typ SQL

1) Prohlédněte si první odpověď a přímo na stránce ověřte, že po úpravě překlepu v názvu filmu se zobrazí stejný výsledek jako u referenční odpovědi.

Úkol cvičícímu napovídá, že se v dotazu nachází překlep. Zde se testuje, jestli cvičící přijde na to, jak najít a použít playground.

Předpokládaný postup: Cvičící u dotazu ve studentské odpovědi klikne na tlačítko „Playground“, dotaz se mu zkopíruje do boxu playground a on v jeho editoru upraví překlep. Po provedení upraveného dotazu zjistí, že výsledky jsou stejné.

2) Pokud se stejný výsledek zobrazí, napište k odpovědi komentář: „Překlep v názvu filmu“. Upravte návrh bodů tak, aby vám nabízel plný počet bodů a odpověď ohodnoťte.

Nyní se více otestuje funkce našeptávače komentářů. Komentář zmíněný v úkolu se totiž vyskytuje v již opravené odpovědi, ale tam za tuto chybu byly strženy body. Cílem je přimět našeptávač, aby změnil navrhovaný počet bodů u tohoto komentáře na maximum.

Předpokládaný postup: Cvičící začne psát komentář a po napsaní jeho části si všimne, že se mu zobrazil návrh na doplnění. Ten cvičící pravděpodobně nejprve potvrdí. Když zjistí, že mu návrh nezobrazuje plný počet bodů, smaže část komentáře a v našeptavači přepíše navrhovaný počet odečtených bodů na 0. Návrh komentáře poté potvrdí a odpověď ohodnotí dle návrhu za plný počet bodů.

3) Pro účely testování vyplňte u další odpovědi pouze komentář „má více řádků“ a udělte za ni 2,5 bodů. Pak opravte ještě další odpověď, napište komentář „má více sloupců“ a udělte za ni 2 body.

Tento úkol cvičícímu jasně říká, jak má odpovědi ohodnotit a slouží pouze jako příprava na další úkol.

Předpokládaný postup: Přesně dle znění úkolu.

4) Obdobně opravte i třetí odpověď. Přidejte dva dílčí komentáře: „má více sloupců“ a „má více řádků“ tak, aby počet odečtených bodů v návrhu odpovídal součtu odečtených bodů ze dvou předchozích odpovědí, tj. aby byl 0,5 bodu. Odpověď ohodnoťte.

Cílem tohoto úkolu je otestovat, zda bude cvičícímu jasný způsob zadávání více dílčích komentářů a následné jejich využití při návrhu bodů.

Předpokládaný postup: Cvičící zadá do pole pro komentáře v libovolném pořadí oba komentáře tím způsobem, že každý napíše na vlastní řádek. Systém poté z těchto dílčích chyb spočítá návrh.

4.2.5.5 Typ transformace

1) Podívejte se na první studentskou odpověď a zobrazte ji v transformovaném tvaru. Dále zobrazte diff s referenční odpovědí a odpověď ohodnoťte.

Úkol testuje, jak se cvičící zorientuje v novém způsobu zobrazování studentské odpovědi v základním tvaru, transformovaném tvaru a ve vyznačených rozdílech s referenční odpovědí.

Předpokládaný postup: Cvičící si prohlédne studentovu odpověď, tak jak ji student napsal, dále zobrazí transformovanou odpověď a diff s referenční odpovědí. Přepínáním mezi transformovanou odpovědí a zobrazením diff nalezne rozdíly v mezi studentskou a referenční odpovědí a odpověď opraví.

2) U další odpovědi nastavte plný počet bodů a přidejte ji mezi referenční odpovědi.

Jedná se o přípravu na následující úkol.

Předpokládaný postup: Přesně dle znění úkolu.

3) Zobrazte diff mezi další neohodnocenou odpovědí a odpovědí, kterou jste právě přidal mezi referenční.

Úkol zjišťuje, zda je cvičícímu zřejmé fungování primární referenční odpovědi při zobrazování rozdílů mezi studentskou a referenční odpovědí. U rozdílu se totiž používá vždy primární odpověď.

Předpokládaný postup: Cvičící klikne na možnost „Zobrazit ostatní“ v boxu s referenčními odpověďmi a nalezne nově přidanou odpověď. Poté kliknutím na ikonu hvězdičky v pravém horním rohu přesune odpověď nahoru a tím ji označí za primární.

4.3 Průběh testování

Testování bylo provedeno se čtyřmi cvičícími. Vzhledem k tomu, že spuštění verze DBS portálu, ve které se nachází řešení nového frontendu manuálních oprav, vyžaduje složitější konfiguraci a přístup k repozitáři projektu na GitLabu, testování probíhalo osobní formou na notebooku autora práce.

Z každého testování byl pořízen záznam pomocí programu *OBS Studio*. Cvičící, kteří se testování z účastnili byli o smyslu testování a cílech této práce informováni už dříve, protože se všichni účastnili také rozhovorů při získávání požadavků. Na začátku testování byli cvičící požádáni, aby komentovali svůj postup a popisovali své myšlenky. V následujícím textu je popsáno, jak jednotlivá testování probíhala a jestli probíhala v souladu s očekávaným postupem, zmíněným u úkolů v části 4.2.5.

4.3.1 První testování

Účastníkem prvního testování byl Ing. Jan Matoušek. Záznam z testování se nachází na přiloženém médiu ve složce testing/recordings/testing_1. Všechny úkoly v části, věnující se typu *checkbox* byly splněny přesně dle předpokládaného postupu. Stejně tak byly splněny také všechny úkoly, věnující se opravě odpovědí typu *text*. U typu *text* v úkolu číslo 4 vyhledal cvičící referenční odpověď sám v referenčních odpovědích a nepoužil vyhledávání.

V části s otázkou typu *diagram* účastník testování nejprve bez problému splnil první úkol a přesunul boxy pomocí funkce drag and drop. Ve druhém úkolu chtěl psát cvičící komentáře přímo do komponenty *Kreslítko*, ale nevěděl, že v něm musí být aktivován režim opravy. Autor práce mu tuto skutečnost napověděl. Cvičící poté vyplnil opravu přímo do komentáře, ale už ji nepřidal do pole s komentářem. Nebylo mu jasné, co přesně odkaz pod diagramem dělá. Další úkoly v části *diagram* byly již splněny bez problémů.

Také splnění prvního úkolu u typu *SQL* proběhlo dle očekávání. Cvičící ale zmínil, že by mělo jít snáze porovnat výsledek z funkce playground s referenční odpovědí. Při úpravě navrhovaných bodů z komentáře ve druhém úkolu zkoušel nejprve účastník testování klikat na šipky v návrhu bodů. Autor mu v této fázi poskytl drobnou radu, na základě které cvičící přišel na způsob, jakým navrhované body upravit. Všechny další úkoly v části věnující se *SQL* byly splněny dle předpokládaného postupu. Žádné problémy se nevyskytly ani při procházení úkolů u otázky typu *transformace*.

4.3.2 Druhé testování

Druhé testování proběhlo se cvičícím Ing. Cyrilem Černým. Při plnění úkolu č. 1 v části *checkbox* hodnotil cvičící dle předkládaného návrhu. U druhého úkolu nebylo cvičícímu hned jasné, že se otázky seskupují, ale po menší radě od autora práce přišel na správné řešení. Ocenil by, kdyby se k informaci o počtu seskupených odpovědí zobrazoval *tooltip*, který by lépe popsal, co tento počet znamená.

Všechny úkoly s odpovědí *text* byly bez problémů splněny. Čtvrtý úkol splnil cvičící tak, že měl zobrazené všechny referenční odpovědi na levé straně a hledanou odpověď našel pomocí vyhledávání ve stránce, které nabídne prohlížeč po stisknutí *ctrl+f*. V úkolu č. 2 u typu *diagram* cvičící zapsal komentář do komentářového pole s hodnocením. Později uvedl, že komentování přímo do komponenty *Kreslítko* nevyužívá nikdy. Při plnění třetího úkolu u typu diagram cvičící, po krátkém hledání řešení, sám objevil možnost použít vyhledávací pole a úkol splnil.

Dále při plnění prvního úkolu u typu *SQL* cvičící uvedl, že by se měl výstup, který vrátí playground, porovnat s referenční odpovědí a případně uvést, že jsou oba výsledky identické. Všechny ostatní úkoly cvičící splnil dle očekávaného postupu.

Videozáznamy ze druhého testování se nachází na přiloženém médiu ve složce testing/recordings/testing_2. Při testování došlo k technickým problémům se zařízením, kde probíhalo testování a které ovlivnily nahrávání. Video zachycující dění na obrazovce proto nezaznamenalo celé testování a končí u druhého úkolu typu *diagram*.

4.3.3 Třetí testování

Třetího testování se zúčastnil Ing. Jan Blizničenko. Záznam z testování se nachází na přiloženém médiu ve složce testing/recordings/testing_3. V části věnované odpovědím typu *checkbox* byly všechny úkoly splněny dle předpokládaného scénáře. Stejně

tak byly splněny také první tři úkoly u typu *text*. Čtvrtý úkol splnil cvičící tak, že hledanou odpověď našel ve výpisu všech referenčních odpovědí. Chybnou referenční odpověď vyřadil z referenčních tím způsobem, že snížil počet za ní udělených bodů. Později uvedl, že by bylo vhodné napsat, že odpověď bude po snížení bodů vyřazena z referenčních odpovědí.

Při řešení prvního úkolu v části *diagram* cvičící odhalil možnost boxy na levé straně přesouvat po nápovědě autora práce. V úkolu č. 2 u typu *diagram* cvičící vypsal chyby přímo do komentáře k hodnocení. Po testování uvedl, že možnosti psát komentáře přímo do komponenty *Kreslítko* nevyužívá.

Během plnění třetího úkolu u typu diagram hledal nejprve cvičící řešení přímo v komponentě *Kreslítko*. Zde cvičícímu nejprve zřejmě nebylo moc jasné zadání úkolu, které možná až příliš klade důraz na to, že se má vyhledávat entita v diagramu. Po vysvětlení, že na typu odpovědi nezáleží, cvičící rychle nalezl vyhledávací pole a úkol splnil. Při rušení filtru s vyhledávanými odpověďmi cvičící uvedl, že se mu hůře kliká na křížek pro vymazání vyhledávacího pole.

Stejně jako v minulých testování u typu *SQL*, i zde cvičící vyjádřil názor, že by se měl nějak zobrazovat rozdíl mezi výsledkem, který vrátí playground a výsledkem, který vrátí referenční odpovědí. Práci s tímto novým prvkem ale zvládl cvičící bez problémů. Po krátkém hledání nalezl cvičící také způsob jak změnit návrh bodů v našeptavači komentářů. Při testování čtvrtého úkolu u typu *SQL* cvičící nejprve oddělovat dílčí komentáře čárkou. Sám ale brzy přišel na to, že je potřeba je psát pod sebou. Se splněním cílů úkolů u otázky typu *transformace* cvičící žádné problémy neměl.

4.3.4 Čtvrté testování

Čtvrté testování bylo provedeno s Ing. Davidem Šenkýřem. První dva úkoly u typu *checkbox* byly splněny dle očekávání. U třetího úkolu se chtěl cvičící vrátit na stránku s výpisem otázek kliknutím na položku „Manuální opravy“ v levém menu portálu s tím, že by se pak proklikal k opravovanému testu. V minulé verzi manuálních oprav bylo totiž tlačítko „Zpět“ hůře dostupné a cvičící byl zvyklý se vracet na přehled všech otázek právě tímto způsobem.

Plnění úkolů u otázky typu *text* proběhlo dle očekávání. Referenční odpověď ve čtvrtém úkolu byla nalezena procházením boxu s referenčními odpověďmi. U typu *diagram*, cvičící nepostupoval v souladu s očekávaným postupem pouze při plnění úkolu č. 2. V tomto případě popsal chyby v diagramu rovnou do komentáře. Po skončení testování uvedl, že při opravování testů, psaní komentářů do komponenty *Kreslítko* nevyužívá.

Při procházení scénáře pro opravu odpovědí typu *SQL* si cvičící u úkolu č. 2 nejprve nebyl jistý, co přesně je jeho cílem. Po vyjasnění sám nalezl způsob, jakým se body v návrhu přepisují. Všechny ostatní úkoly byly splněny dle předpokládaného postupu. Také toto testování byly zaznamenáno na video. Záznam se nachází na přiloženém médiu ve složce testing/recordings/testing_4.

4.4 Dotazník po testování

Po skončení každého testování bylo každému účastníkovi položeno několik otázek, na které měl odpovědět „ano“ nebo „ne“. Otázky se týkaly jak celkové spokojenosti s novou verzí uživatelského rozhraní, tak také s použitelností a využitelností nově

Tabulka 4.1: Dotazník po testování

Otázka	Ano	Ne
Hodnotíte uspořádání stránky a možnosti přizpůsobení jako lepší, než v minulé verzi?	4	0
Myslíte, že budete přesouvání boxů na levé straně využívat při skutečném opravování?	2	2
Myslíte si, že při skutečném opravování využijete funkci našeptávání komentářů?	4	0
Zefektivní funkce našeptávání komentářů spojená s návrhem bodů Váš proces opravování odpovědí?	4	0
Bylo pro Vás použití funkce našeptávání komentářů intuitivní?	4	0
Vyhovoval Vám způsob, jakým je možné u návrhu komentáře změnit navržené body?	3	1
Myslíte, že budete funkci na vyhledávání již opravených odpovědí využívat při skutečném opravování?	4	0
Zefektivní předvyplněné navržené body u otázky typu checkbox Váš proces opravování odpovědí?	3	1
Přišel Vám způsob práce s více referenčními odpověďmi intuitivní?	4	0
Myslíte si, že budete u otázek typu SQL (či RA) využívat funkci playground?	4	0
Chybí Vám v nové verzi manuálních oprav nějaké funkce, které byly v minulé verzi?	0	4

přidaných funkcí. Dotazník, včetně odpovědí, je uveden v tabulce 4.1. Každý řádek tabulky obsahuje nejprve otázku. V dalších sloupcích je počet cvičících, kteří na tuto otázku odpověděli „ano“ či „ne“.

V průběhu vyplňování dotazníku i po něm probíhala krátká diskuze autora práce s účastníkem testování. Při ní se mohl cvičící detailněji rozvést svou odpověď. Také výstupy z této diskuze bral autor této práce v potaz při zhodnocování výsledků testování.

4.5 Zhodnocení výsledků testování

Většina úkolů nedělala při testování cvičícím žádný problém a cvičící je splnili předpokládaným postupem. Z dotazníku vyplývá, že ne všechny nově přidané funkce budou všichni cvičící využívat (návrh u typu *checkbox*, přesouvání boxů na levé straně), ale tyto funkce jim při opravování nebudou nijak překážet.

Důležitým výstupem z dotazníku je pak poslední otázka. Žádný ze čtyř účastníků testování si při procházení testového scénáře nevšiml, že by v nové verzi chybělo něco, co ve staré verzi při opravě využívali.

Cvičící, kteří se zúčastnili ostrého testování o nově přidaných funkcích před začátkem testování nevěděli. Pokud měl některý cvičící během testování nějaký problém se splněním úkolu, buď na řešení po chvíli sám přišel, nebo mu autor práce poskytl menší nápovědu. Nikdy se nestalo, že by cvičící nedokázal úkol splnit vůbec. Pokud budou cvičící vědět, jaké možnosti jim portál nabízí a zvyknou si na nové funkce, neměli by mít problém s jejich použitím.

4.5.1 Nalezené nedostatky

I přes celkovou spokojenost cvičících s novou verzí manuálních oprav objevilo testování některé nedostatky. V řešení také zůstalo několik nedostatků, které byly zjištěny během testování s vedoucím práce (kapitola 4.2.4). Přetrvávající nedostatky ze všech testování jsou popsány v tomto textu.

Odstranění některých nedostatků bude snadné a více vyjasní uživatelům použití prvků v aplikaci. Jedním z těchto menších nedostatků je chybějící *tooltip* u informace s počtem seskupených odpovědí. Do tohoto *tooltipu* by bylo vhodné vysvětlit, že odpovědi jsou seskupeny. Informace o tom, že referenční odpověď při snížení počtu bodů, bude z referenčních odpovědí odstraněna by měla být také sdělena jasněji. Dále by při vyhledávání měla být k seznamu odpovědí přidána informace, že je u nich aktivní filtr.

Tři účastníci testování při použití funkce playground poukazovali na problém, že výsledek, který playground vrací, nelze snadno porovnat s referenční odpovědí. Jelikož smyslem funkce playground je právě to, aby se po úpravě studentova dotazu dal snadno zjistit rozdíl s referenční odpovědí, musí být tento nedostatek odstraněn.

Žádný ze cvičících nesplnil úkol č. 2 u typu otázky *diagram* dle předpokládaného postupu. Většina účastníků testování uváděla, že při opravě testů komentář k jednotlivým chybám do komponenty *Kreslítko* vůbec nepíše a popis chyb vypsala přímo do pole pro komentář. Žádnému z cvičících nebylo zřejmé, k čemu složí odkaz pod diagramem a tak pomocí něho chyby označené v *Kreslítku* nevložili. Většina účastníků testování zároveň zmiňovala, že tímto způsobem otázky tohoto typu ani neopravují.

Během plnění tohoto úkolu se také ukázalo, že komentování chyb přímo v komponentě *Kreslítko* není příliš intuitivní a není jasné, že k otevření okna pro napsání komentáře je třeba mít spuštěný režim oprav. Cvičícím dále vadilo, že *Kreslítko* u jednotlivých prvků neobsahuje informaci, že tam k nim vložen komentář. Důvod vkládání komentáře z diagramu pomocí odkazu byl vysvětlen již v kapitole 3.5.5.2). *Prompt*, kam se komentář zadává, nenabízí mnoho možností, jak na vložení komentáře reagovat. Vzhledem k těmto skutečnostem se proto autorovi práce jeví jako nejlepší řešení provést aktualizaci samotné komponenty *Kreslítko*, v ní vylepšit režim oprav a poté tuto komponentu provázat s manuálními opravami v novém testovém modulu.

Během testování s vedoucím práce (kap. 4.2.4) se objevily nedostatky při opravování odpovědí typu *transformace*. V současné verzi testového modulu byly totiž k opravám tohoto typu odpovědí přidány nové prvky. Jedná se o návrh bodů a o výpis rozdílů mezi opravovanou studentskou odpovědí a s ní nejvíce shodnou referenční odpovědí. Tyto prvky nová verze zatím neobsahuje. Jejich implementace do nové verze by se inspirovala řešením v současné verzi. Nový frontend umožňuje zobrazování návrhu bodů u jiných typů odpovědí. Po vypočítání návrhu u otázky typu *transformace* by se návrh předkládal uživateli stejným způsobem. U seznamu s rozdíly by bylo nutné vyznačit, s jakou referenční odpovědí se porovnání automaticky provedlo.

Dále vedoucí práce během testování navrhl některé úpravy, které se týkaly práce s referenčními odpověďmi. Referenční odpovědi, které nejsou v rámci opravovaného testu mezi studentskými odpověďmi, není možné na stránce pro opravu otázky odstranit z referenčních odpovědí. Dalo by se například přidat do pravého horního rohu ke každé referenční odpovědi možnost tuto odpověď odebrat. Je ale otázkou, jak často by se tato možnost využívala. Odstraňování každé referenční odpovědi z otázky by mělo být možné v části portálu, kde se otázky vytváří a editují. Je možné, že se při praktickém používání ukáže, že tato možnost u manuálních oprav chybí. V takovém případě bude ale její implementace snadná.

Další nedostatek se týká informačního okna, které zobrazí po odeslání hodnocení celkový počet ohodnocených odpovědí. Uživateli zde není nabídnuta možnost ohodnocenou odpověď přidat mezi referenční (pokud je za plný počet bodů). Stejně tak, pokud by uživatel přidal odpověď mezi referenční omylem, měl by mít možnost toto rozhodnutí podobným způsobem změnit. Při opravování odpovědí může totiž cvičící snadno kliknout na jiné tlačítko pro ohodnocení odpovědi, než zamýšlel. Tento nedostatek by měl být tedy také odstraněn.

4.6 Změny po testování

Testování přineslo řadu návrhů na to, jak nový frontend manuálních oprav vylepšit. V samotném závěru tvorby této bakalářské práce tak byly v reakci na výsledky testování provedeny menší změny. Jednalo se o drobnější úpravy, které byly popsány v kapitole 4.5.1. Dále byly realizovány změny, které tato kapitola označila za podstatné.

Jednou z nich bylo rozšíření informačních oken, které se zobrazují po ohodnocení odpovědi. Při zadání plného počtu bodů, ale pouhého ohodnocení lze nyní odpověď ještě zpětně kliknutím na tlačítko v tomto okně přidat do referenčních odpovědí. Pokud je odpověď přidána do referenčních odpovědí, lze jí podobným způsobem z referenčních odpovědí odebrat.

Další změna se týkala funkce playground, ve které se nyní výsledek porovná s referenční odpovědí. Cvičící při testování byly s funkcí playground velmi spokojeni a horší způsob porovnávání s referenční odpovědí uváděli jako největší výtku. Uživateli se proto nyní zobrazí informace, zda jsou výsledky shodné, nebo v čem se liší. Podobné porovnání se už předtím provádělo u studentských odpovědí. Kvůli této úpravě byla v kódu přidána nová komponenta *ResultsDiff*, kterou nyní pro zjištění a zobrazení rozdílů využívají jak studentské odpovědi, tak také playground. Tento nedostatek byl tímto také odstraněn.

Závěr

Cílem této práce bylo vytvořit nový frontend pro manuální korekturu otázek v testovém modulu portálu dbs.fit.cvut.cz.

Nejprve byla provedena analýza současného stavu, byly nalezeny jeho nedostatky a zjištěny možnosti jeho vylepšení. Poté byl vytvořen návrh, proběhla realizace a její výsledek byl otestován na cvičících předmětu BI–DBS.

Během analýzy bylo nutné se nejprve seznámit s celým testovým modulem DBS portálu. Dále práce popsala současné řešení manuálních oprav. Tato část analýzy významně přispěla k pochopení problematiky manuálního opravovaní testů v portálu a popsala jeho důležité vlastnosti. Další podstatnou částí analýzy byly rozhovory s cvičícími. Ti velmi ochotně pomohli nalézt nedostatky současného řešení a přednesli také požadavky na novou verzi, které by člověk bez praktické zkušenosti s touto částí portálu neodhalil.

Díky takto nalezeným požadavkům pak bylo snazší vytvořit návrh, který prvky ze současného řešení vylepšil a obohatil o nové funkce. Zároveň posloužil jako základ pro samotnou implementaci. Jejím výsledkem je funkční uživatelské rozhraní, které je integrováno do části DBS portálu, v níž vzniká nový modul pro psaní testů.

Po skončení hlavní části implementace následovalo testování. V rámci něj bylo zjišťováno, jak výsledný produkt obstojí u svých koncových uživatelů. Cvičící, kteří se zúčastnili testování, hodnotili výslednou aplikaci pozitivně a zároveň poskytli návrhy na její další vylepšení. Některé z těchto návrhů byly posléze do nového frontendu manuálních oprav zapracovány.

Celý DBS portál se neustále vylepšuje a i když už je několik let nedílnou součástí výuky předmětu BI–DBS, vyučující i studenti stále přicházejí s novými nápady, které portál obohacují. Frontend pro opravu otázek, kterému se tato práce věnovala, jistě nebude výjimkou. Je pravděpodobné, že vývojáři portálu, ale také vyučující, přijdou s dalšími návrhy na jeho rozšíření. Pro plné nasazení bude potřeba především vyvinout backend a také všechny další části nového testového modulu.

Teprve po integraci ostatních částí a dlouhodobějším používání se ukáže, jak řešení, realizovaná v této práci, obstojí. Vzhledem k tomu, že k získání požadavků a testování výsledného produktu byla využita pomoc koncových uživatelů, věřím, že výsledek této práce bude v praxi užitečný a významně usnadní vyučujícím předmětu BI–DBS opravu testů.

Bibliografie

1. PLYSKACH, Andrii. *Refaktoring testové části backendu portálu dbs.fit.cvut.cz*. Praha, 2020. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií.
2. MALEC, Oldřich. *Řízení projektu a infrastruktury portálu pro podporu výuky předmětu BI-DBS*. Praha, 2017. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií.
3. PEJŠA, Petr. *Systém pro podporu testování v BI-DBS*. Praha, 2016. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií.
4. ERBEN, Marek. *Automatické generování a oprava otázek na normalizaci databáze pro předmět BI-DBS*. Praha, 2018. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií.
5. MACHALA, Filip. *Automatická oprava zjednodušeného relačního zápisu*. Praha, 2018. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií.
6. *ECMA-404: The JSON data interchange syntax*. 2nd Edition. Ženeva: Ecma International, 2017.
7. ŠACH, Martin. *Podpora tvorby a automatická oprava zjednodušeného relačního zápisu v portálu dbs.fit.cvut.cz*. Praha, 2020. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií.
8. FEDOR, Tomáš. *ER diagrams web component II*. Praha, 2017. Diplomová práce. České vysoké učení technické v Praze, Fakulta informačních technologií.
9. REICHEL, Jiří. *Kapitoly metodologie sociálních výzkumů*. Vyd. 1. Praha: Grada, 2009. ISBN 978-80-247-3006-6.
10. WIEGERS, Karl Eugene; BEATTY, Joy. *Software requirements*. 3rd ed. Redmond, WA: Microsoft press, 2013. ISBN 978-0-7356-7966-5.

11. *Wireframing* [online]. Washington, D.C.: U.S. General Services Administration [cit. 2022-04-19]. Dostupné z: https://www.usability.gov/how-to-and-tools/methods/wireframing.html.

12. OUELLETTE, Alexandre. *What is Bootstrap: A Beginner's Guide* [online]. Berlín: CareerFoundry GmbH [cit. 2022-04-24]. Dostupné z: https://careerfoundry.com/en/blog/web-development/what-is-bootstrap-a-beginners-guide/.

13. *Začínáme s Latte* [online]. Nette Foundation, © 2008–2022 [cit. 2022-04-24]. Dostupné z: https://latte.nette.org/cs/guide.

14. *Introduction* [online]. Vue.js, © 2014–2022 [cit. 2022-04-24]. Dostupné z: https://vuejs.org/guide/introduction.html.

15. *Components Basics* [online]. Vue.js, © 2014–2022 [cit. 2022-05-01]. Dostupné z: https://vuejs.org/guide/essentials/component-basics.html.

16. *Mixins* [online]. Vue.js, © 2014–2022 [cit. 2022-05-01]. Dostupné z: https://v2.vuejs.org/v2/guide/mixins.html.

17. *Introduction* [online]. Mirage JS [cit. 2022-05-01]. Dostupné z: https://miragejs.com/docs/getting-started/introduction/.

18. *Mirage JS: Tutorial* [online]. Mirage JS [cit. 2022-05-03]. Dostupné z: https://miragejs.com/tutorial/.

19. CHURM, Thomas. *An Introduction To Website Usability Testing* [online]. Usability Geek [cit. 2022-04-22]. Dostupné z: https://usabilitygeek.com/an-introduction-to-website-usability-testing/.

20. JANOVSKÝ, Dušan. *Použitelnost stránek* [online] [cit. 2022-04-22]. Dostupné z: https://www.jakpsatweb.cz/pouzitelnost.html.

21. VOJÁK, Michal. *Jak dělat uživatelské testování* [online]. Praha: Designers & Developers s.r.o. [cit. 2022-04-22]. Dostupné z: https://designdev.cz/jak-delat-uzivatelske-testovani.

22. ONIFADE, Aliu A. *Personas: Usability Testing* [online]. Luxembourg: LSBC, c2021 [cit. 2022-05-02]. Dostupné z: https://lsbc.lu/personas-and-usability-testing/.

23. *Personas* [online]. Washington, D.C.: U.S. General Services Administration [cit. 2022-05-02]. Dostupné z: https://www.usability.gov/how-to-and-tools/methods/personas.html.

24. NIELSEN, Jakob. *Why You Only Need to Test with 5 Users* [online]. Fremont, California: Nielsen Norman Group, © 1998–2022 [cit. 2022-05-02]. Dostupné z: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/.

25. PAVLÍČEK, Josef; KOL. *The Cookbook for Interaction Design and Human Computer Interaction: User Interface Testing* [online] [cit. 2022-04-22]. Dostupné z: `https://docs.google.com/presentation/d/1t-4kCvHJSqpzqff30JhoAzPvaJQQElJ990geai3K9e8/edit`.

26. VOJÁK, Michal. *Příprava testu použitelnosti* [online]. Praha: Designers & Developers s.r.o. [cit. 2022-05-02]. Dostupné z: `https://designdev.cz/priprava-testu-pouzitelnosti`.

Příloha A

Seznam použitých zkratek

API Application Programming Interface

BI–DBS Databázové systémy

BI–SP1 Softwarový týmový projekt 1

BI–SP2 Softwarový týmový projekt 2

CSS Cascading Style Sheets

ČVUT České vysoké učení technické v Praze

FIT Fakulta infromačních technologií

HTML Hypertext Markup Language

JSON JavaScript Object Notation

RA Relační algebra

PHP PHP: Hypertext Preprocessor

SQL Structured Query Language

URL Uniform Resource Locator

Příloha B

Testování – skripty

```
#!/bin/bash

# kontrola vstupu
if [ $# -eq 2 ]; then
  fileA="$1"
  fileB="$2"
else
  echo 'Musi byt zadany 2 parametry: vstupni_soubor konfiguracni_soubor'
  exit 1
fi

if [ ! -f "$fileA" ] || [ ! -f "$fileB" ]; then
  echo 'Soubor(y) nebyly nalezeny'
  exit 1
fi

outFile="${fileA%.*}-out.${fileA##*.}"
cp "$fileA" "$outFile"


while read line; do
  IFS=''
  line2=`echo $line | sed -e 's/[[:space:]]*$//'`
  IFS=','
  read -a lineContents <<< "$line2"
  if [ "${#lineContents[@]}" -eq 2 ]; then
    sed -i 's/'${lineContents[0]}'/'${lineContents[1]}'/g' "$outFile"
  fi
done < "$fileB"

echo "Vystup ulozen do ${outFile}"
```

Ukázka kódu B.1: Skript na anonymizaci souboru

```
-- Remove conflicting tables
DROP TABLE IF EXISTS film CASCADE;
DROP TABLE IF EXISTS osoba CASCADE;
DROP TABLE IF EXISTS osoba_film_zhlednuti CASCADE;
-- End of removing

CREATE TABLE film (
    id_film SERIAL NOT NULL,
        nazev VARCHAR(256) NOT NULL,
    datum_premiery DATE NOT NULL,
    delka INTEGER NOT NULL,
    zanr VARCHAR(256) NOT NULL
);
ALTER TABLE film ADD CONSTRAINT pk_film PRIMARY KEY (id_film);

CREATE TABLE osoba (
    id_osoba SERIAL NOT NULL,
    jmeno VARCHAR(256) NOT NULL,
    prijmeni VARCHAR(256) NOT NULL,
    nejoblibenejsi_film INTEGER
);
ALTER TABLE osoba ADD CONSTRAINT pk_osoba PRIMARY KEY (id_osoba);

CREATE TABLE osoba_film_zhlednuti (
        id_osoba INTEGER NOT NULL,
    id_film INTEGER NOT NULL,
    pocet_zhlednuti INTEGER NOT NULL
);
ALTER TABLE osoba_film_zhlednuti
    ADD CONSTRAINT pk_osoba_film_zhlednuti
    PRIMARY KEY (id_film, id_osoba);

ALTER TABLE osoba
    ADD CONSTRAINT fk_osoba_film
    FOREIGN KEY (nejoblibenejsi_film) REFERENCES film (id_film)
    ON DELETE CASCADE;

ALTER TABLE osoba_film_zhlednuti
    ADD CONSTRAINT fk_osoba_film_zhlednuti_film
    FOREIGN KEY (id_film) REFERENCES film (id_film)
    ON DELETE CASCADE;
ALTER TABLE osoba_film_zhlednuti
    ADD CONSTRAINT fk_osoba_film_zhlednuti_osoba
    FOREIGN KEY (id_osoba) REFERENCES osoba (id_osoba)
    ON DELETE CASCADE;
```

Ukázka kódu B.2: Create script pro otázku typu SQL

Příloha C

Testování – dokumenty

C.1 První verze scénáře

1. checkbox
   a) Otevřete stránku s opravou otázky *2017-t-left outer join význam* (typ checkbox) a prohlédněte si zadání, úkol a referenční odpověď.
   b) Opravte první odpověď.
   c) Sedm studentů zvolilo při odpovídání na otázku stejné možnosti. Opravte jedním hodnocením odpověď všem těmto studentům.
   d) Vraťte se na stránku se všemi otázkami z testu.
2. text
   a) Otevřete stránku s opravou otázky *2017-t-ACID* (typ text) a prohlédněte si zadání, úkol a referenční odpověď.
   b) Opravte první tři neopravené odpovědi. Pokud neudělujete plný počet bodů, uveďte důvod do komentáře.
   c) Opravte další neopravenou odpověď a zařaďte ji mezi referenční odpovědi. Dále zařiďte, aby se tato odpověď zobrazovala na levé straně, aniž by byly zobrazeny ostatní referenční odpovědi.
   d) Zjistil jste, že se Vám někdo snažil pomoci a opravil již dříve část odpovědí za Vás. Udělal ale chybu a označil chybnou odpověď za referenční. V této odpovědi student dokonce sám uvedl „*Nazvy nevim, hadam*“. Najděte tuto odpověď a odstraňte ji z referenčních odpovědí, aniž byste opustili stránku s hodnocením otázky.
   e) Vraťte se na stránku se všemi otázkami z testu.
3. diagram
   a) Otevřete stránku s opravou otázky *2018-Diagram-cestovani* (typ diagram) a prohlédněte si zadání, úkol a referenční odpověď.
   b) Zařiďte, abyste na levé straně viděl zadání i referenční odpověď bez nutnosti levou stranou skrolovat.

c) Napište komentář k první odpovědi, kde vypište a popište chyby, které v diagramu označila automatická oprava. Otázku ohodnoťte.

d) Prohlédněte si další odpověď. Máte pocit, že řešení ve kterém student použil entitu *Wishlist* jste už dříve viděl a ohodnotil. Na první pohled se vám zdá odpověď stejná a chcete se podívat, v čem se liší. Najděte dříve ohodnocenou odpověď s entitou *Wishlist* a zjistěte rozdíl. Neohodnocenou otázku poté ohodnoťte.

e) Zařiďte, abyste na pravé straně stránky viděl všechny dříve opravené odpovědi.

f) Vraťte se na stránku se všemi otázkami z testu.

4. SQL

a) Otevřete stránku s opravou otázky *2019 - filmy - neviděli film* (typ SQL) a prohlédněte si zadání, úkol a referenční odpověď.

b) Prohlédněte si první odpověď a přímo na stránce ověřte, že po úpravě překlepu v názvu filmu se zobrazí stejný výsledek jako u referenční odpovědi.

c) Pokud se stejný výsledek zobrazí, napište k odpovědi komentář: „Překlep v názvu filmu“. Upravte návrh bodů tak, aby vám nabízel plný počet bodů a otázku pomocí něj ohodnoťte.

d) Postupně opravte další tři odpovědi. Pište k nim komentáře a pro zjednodušení (a pro účely testování) pište do komentářů pouze rozdíly mezi referenční a studentskou odpovědí (jsou na stránce vypsány, např. „má více řádků“ atp.). Body udělte libovolné.

e) Vraťte se na stránku se všemi otázkami z testu.

5. transformace

a) Otevřete stránku s opravou otázky *2021-ucitelske-basne-transformace* a prohlédněte si zadání, úkol a referenční odpověď.

b) Opravte první studentskou odpověď, tak jako kdyby jste ji opravoval ve skutečnosti.

c) U další odpovědi nastavte plný počet bodů a přidejte ji mezi referenční odpovědi.

d) Zobrazte diff mezi další neohodnocenou odpovědí a odpovědí, kterou jste právě přidal mezi referenční.

Příloha D

Obsah přiloženého paměťového média

- `readme.txt` ..... stručný popis obsahu přiloženého média
- `gitlab-branch.txt` ..... soubor s odkazem na GitLab větev s řešením
- `wireframes` ..... návrhy prvků na stránce pro hodnocení otázky
- `testing`
  - `recordings` ..... videozáznamy z testování
  - `scripts` ..... testování – skripty
  - `docs` ..... testování – dokumenty
- `thesis_src` ..... zdrojová forma práce ve formátu LaTeX
- `thesis.pdf` ..... text práce ve formátu PDF