FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	dbs.fit.cvut.cz - Refaktoring testů II
Student:	Pavel Jordán
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 úprava a přepis tvorby testů v portálu dbs.fit.cvut.cz, dále jen portál, s ohledem na snadnou použitelnost pro vyučující. Důvodem je aktuální nutnost navázat na práci Bc. Andriiho Plyskache, který připravil nový backend testů.

Postupujte v těchto krocích:
Analyzujte současný stav tvorby testů (otázek, šablon, apod.) v portálu.
Na základě analýzy navrhněte lepší možnosti tvorby testů tak, abyste minimalizovali nárůst složitosti pro vyučující.
Na základě analýzy implementujte funkční prototyp.
Realizujte vhodné sady testů vašeho prototypu a ověřte jeho použitelnost.
Integrujte Vaše řešení po odladění prototypu do portálu pro budoucí možné použití.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 19. února 2021 v Praze.

Bakalářská práce

DBS.FIT.CVUT.CZ - REFAKTORING TESTŮ II

Pavel Jordán

Fakulta informačních technologií ČVUT v Praze
Katedra softwarového inženýrství
Vedoucí: Ing. Jiří Hunka,
9. května 2022

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2022 Pavel Jordán. Všechna práva vyhrazena.
Tato práce vznikla jako školní díla na Českém vysokém učení technické v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bez uplatněných zákonných licencí nad rámec oprávnění uvedených v Prohlášení je nezbytný souhlas autora.

Odkaz na tuto práci: Pavel Jordán. *dbs.fit.cvut.cz -*
Refaktoring testů II. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych poděkovat především svému vedoucímu, panu Ing. Jiřímu Hunkovi, za možnost volby tohoto tématu a také za veškeré poskytnuté informace týkající se testového modulu DBS portálu a podporu při jeho tvorbě. Dále bych chtěl poděkovat Ing. Oldřichu Malcovi za jeho pomoc v pochopení systému a cenné rady ohledně programování modulu. Také moc děkuji týmu 1 předmětu BI-SP1.2, který se mnou spolupracoval a nakonec bych rád poděkoval své rodině za jejich podporu během studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů 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 předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 9. května 2022 .....

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací prototypu nového frontendu testového modulu DBS portálu za použití nového backendu. Na začátku je provedena kompletní analýza funkcí systému jak současného řešení tak i nových funkcí připraveného backendu pomocí případů užití. Dále se podrobně rozebírají hlavní nedostatky současné implementace. Na základě této analýzy je pak upraven a vytvořen návrh testového modulu s důrazem na jednoduchost a snadné použití. Poslední část této práce se věnuje implementaci prototypového řešení, včetně jeho testování.

Klíčová slova refaktorování frontendu, návrh frontendu, implementace frontendu, DBS portál, testový modul, Vue.js

Abstract

This bachelor thesis deals with the design and implementation of a new frontend prototype for the DBS portal test module using the new backend. At the beginning I perforfm a complete analysis of the system functions for both the current system and the new funcions of the prepared backend using use cases. Furthermore the main shortcomings of the current implementation are discussed in detail. Based on this analysis, the module is then modified and created with emphasis on simplicity and ease of use. The last part of this work deals with the implementation of a prototype solution, including its testing.

Keywords frontend refactoring, frontend design, frontend implementation, DBS portal, testo module, Vue.js

Shrnutí

Motivace

Na DBS portále jsem pracoval už v průběhu studia a hlavní motivací je pro mě pomoc vyučujícím zrychlit a zjednodušit vytváření testů a také možnost vylepšení výuky předmětu DBS, jelikož bude vytváření testů jednodušší. Dále také to, že se portál stále vyvýjí, neustále se používá a tudíž moje řešení bude maximálně využité.

Cíl práce

Cílem práce je vytvořit nový, uživatelsky přívětivější prototyp uživatelského rozhraní testového modulu portálu DBS s ohledem na snadnou použitelnost a zavedení nových funkcionalit připravených Bc. Andriim Plyskachem, který navrhl a vytvořil nový backend.

Postup

Začal jsem analýzou stávajícího řešení a návrhem jednoduchého papírového modelu. Dalším krokem byla analýza nového backendu a následné úpravy papírových modelů. Po této úpravě jsem ve spolupráci s vývojovým týmem 1 z předmětu BI-SP1 na portálu DBS začal s implementací jednotlivých komponent a sestavení rozhraní podle papírového modelu.

Výsledky práce

Výsledkem práce je prototyp uživatelského rozhraní s ukázkovými daty. V následující fázi vývoje už jenom stačí prototyp připojit na funkční API, které v této době ještě není hotové, ale bude součástí diplomové práce Bc. Andriiho Plyskache.

Závěr

Závěrem lze říci, že cíle byly v rámci možností splněny. Nově navržené uživatelské rozhraní splňuje všechny požadavky a vytvořený prototyp lze použít. Nový testový modul jako celek má před sebou ještě dlouhou cestu, avšak základ části nového testového modulu přístupné ze strany učitele je vytvořen a půjde dál efektivně rozvíjet a další části uživatelského rozhraní systému mohou na této práci stavět. V současné době po dokončení práce již tento další vývoj proběhl a stále probíhá.

Seznam zkratek

API	Application Programming Interface
APT	Advanced Packaging Tool
BI-DBS	Databázové systémy
BI-SP1	Softwarový týmový projekt 1
BI-SP2	Softwarový týmový projekt 2
BP	Bakalářská práce
ČVUT	České vysoké učení technické
DBS	Databázové systémy
DBS portál	Systém pro výuku předmětu BI-DBS
DML	Data manipulation language
DOM	Document Object Model
FDD	Feature driven development
FIT	Fakulta informačních technologií
MVC	Model-View-Controller
MVP	Model-View-Presenter
RA	Relační algebra
SQL	Structured query language
XP	Extrémní programování

Kapitola 1

Co je to DBS portál

Tato práce se zabývá vývojem testové části portálu DBS, který se používá pro výuku předmětu BI-DBS na FIT ČVUT v Praze. Ještě než tedy začnu psát o mé práci na portálu, v rychlosti seznámím čtenáře s předmětem samotným.

1.1 Výuka BI-DBS

Předmět databázové systémy je vyučován na katedře softwarového inženýrství a v současném studijním programu se s ním setkají studenti bakalářského programu Informatika ve druhém semestru. Studenti jsou seznámeni s problematikou ukládání dat především pomocí relačních databází. Naučí se navrhovat menší databáze a jejich implementaci. Důraz je kladen na jazyky RA a SQL.

1.1.1 Semestrální práce

Semestrální práce tvoří zhruba čtvrtinu známky studenta. Každý student si v rámci své semestrální práce vymyslí projekt ke kterému si vytvoří vlastní databázi, naplní ji testovacími daty, a následně vymýšlí dotazy. Semestrální práce ve finální podobě obsahuje alespoň 25 dotazů a pokrývá všechny standardní klauzule DML, jednak z jazyku relační algebry tak i z SQL.

1.1.2 Testy v semestru a zkouška

Kromě semestrálních prací jsou součástí hodnocení studentů také průběžné testy v semestru a závěrečná zkouška. Testy v semestru se týkají především praktické části - tedy používání RA a SQL či modelování schémat databáze. Závěrečná zkouška poté kromě praktických částí obsahuje i teoretické otázky, které se probrali na přednáškách.

1.2 Vznik DBS portálu

Obě výše popsané části výuky jsou náročné na opravu vyučujícím. Typicky nebylo možné zajistit, aby vyučující zkontroloval každý dotaz, který student ve své semestrální práci vytvořil. Oprava testů, které se psali na papír, trvala zbytečně dlouho a nebylo možné ji automatizovat. Z těchto důvodů v roce 2013 přišel Jiří Hunka - jeden z vyučujících BI-DBS - s nápadem, že se vytvoří portál zaměřený na podporu výuky Databázových systémů, který umožní efektivnější opravu jak semestrálních prací tak i rychlejší opravu testů a zkoušek. [1]

1.2.1 Vývojový tým

Vývoj portálu takových rozměrů však nebylo možné financovat běžně dostupnými prostředky, kterými fakulta disponuje. Z toho důvodu Jirka Hunka rozhodl, že bude portál vyvíjen v rámci předmětů BI-SP1 a BI-SP2. Avšak katedra Softwarového Inženýrství a hlavně Michal Valenta se zasloužili historicky i o prostředky pro studenty, kteří pracovali nad rámec předmětu na tomto projektu. Jedná se o předměty vyučované v oboru Softwarové inženýrství, které si studenti typicky zapisují ve svém 4., respektive 5. semestru studia. Cílem předmětů je vytvořit 3-5 členné týmy, které budou pracovat na softwarovém projektu, počínaje návrhem, analýzou požadavků atp., a konče hotovým softwarem. Proto je vypsáno zadání na realizaci DBS portálu každý semestr a v průměru se přihlásí více jak 10 studentů. Tento způsob získávání pracovní síly pro další vývoj portálu je využíván dodnes. Jelikož BI-SP1 a navazující BI-SP2 má celkové trvání jeden akademický rok, jsou každý rok nabíráni noví studenti. [1]

V aktuálně používané aplikaci pro řízení projektu DBS je vykázáno více něž 5300 hodin práce, ale na systému se pracovalo i dlouhou dobu před použitím této aplikace. Skutečný odpracovaný čas na portálu DBS je tedy mnohem větší.

Kapitola 2
Analýza

Nedílnou součástí vývojového cyklu jakéhokoli software je analýza domény a sběr požadavků na vývoj systému. Analýza je jeden z velmi významných kroků celého vývojového cyklu. Kvalita provedené analýzy má významný dopad na další fáze vývoje. Chybná nebo neúplná analýza může mít za následek nesprávné pochopení domény, jejímž důsledkem je vytvoření software, který se značně liší od představ zákazníka.

Cílem této kapitoly bude důkladně analyzovat část systému pro vytváření testů. Jedná se o vše potřebné pro vytvoření testové šablony, z které následně učitel spouští testy pro studenty. Pro vytvoření testu je potřeba vytvořit otázky, které se v testu objeví. Otázky potřebují nějaké zadání, úkol, který má student vyřešit a odpověď pro jednoduchou opravu testu.

Výstupem bude seznam funkčních a nefunkčních požadavků spolu s případy užití, které budou základním stavebním kamenem pro návrh nového frontendu. Jelikož navazuji na současné řešení a nově vytvořený backend, který byl připraven v rámci bakalářské práci Bc. Andriiho Plyskache [2], bude během analýzy kladen důraz právě na analýzu současného řešení a na nové funkcionality zavedené Bc. Andriim Plyskachem. Zaměřím se na detailní popis funkčnosti a také na uživatelské role, které mohou se systémem pracovat. V poslední části této kapitoly se následně zaměřím na hlavní nedostatky současného uživatelského rozhraní, které se v další kapitole pokusím vyřešit.

2.0.1 Postup analýzy

Za účelem provedení důkladné analýzy všech potřebných částí systému jsem se nejprve zaměřil na současně používané řešení tvorby testů, u kterého jsem se zaměřil na veškeré jeho funkčnosti. Výstupem této analýzy jsou případy užití a seznam funkčních požadavků. Také jsem si sepsal věci, které by z mého pohledu mohli vytváření testů zjednodušit nebo zrychlit. Ty jsem následně konzultoval se zadavatelem práce Ing. Jiřím Hunkou.

Po analýze současného řešení jsem si přečetl bakalářskou práci Bc. Andriiho Plyskache, ve které byl navržen nový backend pro celou testovou část systému. Zároveň jsem s Andriim ověřil nové funkce a jejich využití, především se jednalo o štítky a štítkové skupiny, které byli nově zavedené pro lepší přehlednost otázek. Výstup z této části byli především nové funkční požadavky a změny v požadavcích současných.

Následně jsem porovnal obě části analýzy a z nich sepsal požadavky nového systému, případy užití aktuálního systému, z kterých jsem vycházel v tvorbě nového frontendu a poznámky s částmi současného řešení, které by šli zjednodušit nebo zrychlit v novém návrhu.

2.1 Současné řešení

Současné řešení představuje systém, který se skládá z několika hlavních komponent:

- semestrální práce,
- testový modul,
- databázový modul,
- administrace uživatelů,
- kreslící nástroj.

Jelikož se tato práce zabývá pouze testovou částí, bude pro nás důležitá pouze testová komponenta a na ni se v dané kapitole omezím. Testová část je používána studenty a vyučujícími pro psaní cvičných testů, testů v semestru a také zkoušek.

Portál se používá od roku 2016. V průběhu dalších několika let byl portál modifikován a rozšiřován. V současnosti se objevily nové požadavky na testový modul, které by se obtížně implementovaly. Proto se Andrii Plyskach rozhodl tyto problémy vyřešit refaktorizací celého modulu, a tak v roce 2020 vznikla bakalářská práce zabývající se refaktorováním backendu testového modulu. [2]

2.1.1 Aktuální struktura aplikace

DBS portál je webová aplikace postavená na frameworku Nette. Nette je kompletní Framework pro jazyk PHP, který výrazně zjednodušuje tvorbu webových aplikací. Je založen na samostatných knihovnách, které dohromady tvoří celý framework. Jedním z hlavních zaměření Nette je vysoká míra zabezpečení pomocí technologií, které eliminují bezpečnostní mezery. Nette je postavené na architektuře MVC. Autorem je český vývojář David Grudl. [3]

Model-View-Controller je softwarová architektura, která dělí aplikaci na 3 logické části, aby šli samostatně upravovat a dopad změn byl na ostatní části co nejmenší. Tyto části jsou Model, View a Controller.[4]

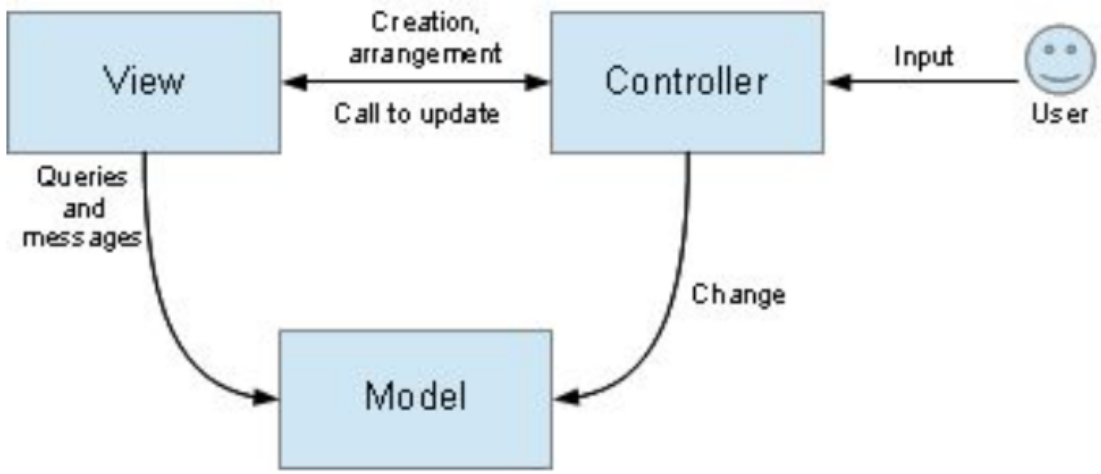
- Model - Reprezentuje data a business logiku aplikace. Jakákoliv akce uživatele (přihlášení, změna hodnoty v databázi, zobrazení stránky) představuje akci modelu. Model si spravuje svůj vnitřní stav a ostatním částem aplikace nabízí pevně dané rozhraní. Model o existenci view nebo kontroleru neví.
- View - Vrstva aplikace, která má na starost zobrazení výsledku požadavku. Obvykle používá šablonovací systém a ví jak zobrazit jednotlivé komponenty nebo výsledky získané z modelu.
- Controller - Zpracovává požadavky uživatele a na jejich základě volá patřičný model. Poté požádá view o vykreslení dat. V Nette Framework jsou obdobou kontrolerů presentery.[5]

Komunikace mezi jednotlivými částmi je znázorněna na obrázku 2.1.

2.2 Případy užití současného řešení

V této kapitole popisuji případy užití systému odpovídající k existujícím funkčním požadavkům.

Diagramy případů užití zobrazují chování systému z pohledu uživatele. Účelem těchto diagramů je popis očekávané funkcionality systému. Případ užití říká, co má systém umět, ale neříká, jakým způsobem to bude dělat. Je to důležitý diagram, pomocí kterého je jednoduché shodnout se na tom, co má systém umět. Diagram případů užití se skládá z případu užití, aktérů a vztahů mezi nimi.[6]



Obrázek 2.1 Architektura MVC [4]

2.2.1 Aktéři

Aktéři jsou role, které v systému vykonávají určité činnosti. Aktér (účastník) může označovat člověka, skupinu lidí, systém nebo dokonce i čas, který má v systému právo provést nějakou konkrétní akci nebo sekvenci akcí. [6] V DBS portálu existují následující účastníci: nepřihlášený uživatel, student, vyučující a garant. Jelikož jediná možná akce, která je dostupná nepřihlášenému uživatelovi, je přihlášení, dále se více rozebírat nebude. Kromě toho se tato práce věnuje pouze testovému modulu ze strany vyučujícího v DBS portálu, proto se ani role studenta nebude rozebírat.

2.2.1.1 Vyučující

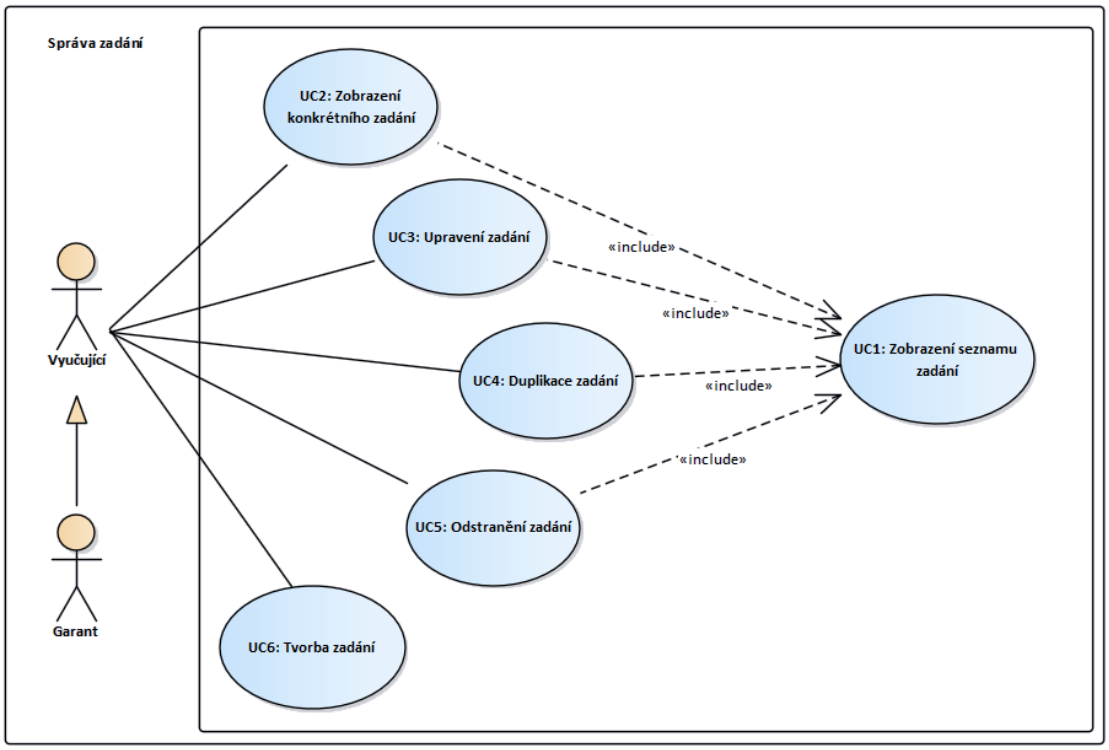
Zodpovědným za tvorbu a správu testů je vyučující. Systém dovoluje učitelům vytvářet nové otázky a zadání, které pak mohou být použity v testech. Vyhodnocení testů je zcela automatické, ale vyučující může ohodnocené testy procházet a opravovat hodnocení, pokud systém nezvládl ohodnotit otázku nebo ji ohodnotil špatně.

2.2.1.2 Garant

Garant je role, jež má stejná práva jako vyučující. Navíc ale může upravovat podmínky klasifikace, např. počet bodů, kterého je potřeba dosáhnout, aby student uspěl u testu.

2.2.2 Správa zadání

V této podkapitole jsou popsané případy užití pro současnou správu testových zadání (F1 2.4.1.1). Diagram případů užití je zachycen na obrázku 2.2



Obrázek 2.2 Diagram případů užití: Správa zadání

2.2.2.1 UC1: Zobrazení seznamu zadání

Případ užití umožní uživateli zobrazit všechna již vytvořená zadání.
Aktéři: vyučující, garant.
Scénář:

1. Případ užití začíná, když se uživatel chce podívat na všechna testová zadání v systému.
2. Uživatel zvolí v levém menu tlačítko „*Zobrazit zadání.*“
3. Systém zobrazí tabulku, která obsahuje veškerá vytvořená zadání.

2.2.2.2 UC2: Zobrazení konkrétního zadání

Případ užití umožní uživateli zobrazit všechny informace o konkrétním zadání.
Aktéři: vyučující, garant.
Počáteční podmínka: v systému existuje alespoň jedno zadání.
Scénář:

1. Uživatel projde případem užití „UC1: Zobrazení seznamu zadání.“
2. Ze seznamu vybere konkrétní zadání a klikne na „*Náhled.*“
3. Systém zobrazí informace o daném zadání.

2.2.2.3 UC3: Upravení zadání

Systém dovoluje uživateli modifikovat zadání. Pokud má alespoň jedna otázka zadání přiřazené, tak se před úpravou systém zeptá, jestli uživatel opravdu chce akci provést.

Aktéři: vyučující, garant.

Počáteční podmínka: v systému existuje alespoň jedno zadání.

Hlavní scénář:

1. Uživatel projde případem užití „UC1: Zobrazení seznamu zadání.“
2. Uživatel vyhledá v seznamu zadání, které chce upravit a klikne na tlačítko „*Upravit.*“
3. Systém zobrazí formulář v závislosti na konkrétním typu zadání, které je jedním z typů: *text, diagram, obrázek nebo normalizace relačního schématu.*
4. Pokud zadání bylo textové, pak systém zobrazí formulář s údaji: název, popis a jazyk.
5. Uživatel upraví jednotlivá políčka a zvolí „*Uložit*“ nebo odmítne změny pomocí tlačítka „*Zpět.*“

Alternativní scénář: zvolené zadání bylo typu „Diagram.“

1. Scénář začíná ve 3. kroku hlavního scénáře.
2. Systém zobrazí formulář s prvky: název, jazyk a interaktivní diagram, který uživatel může modifikovat.
3. Dále scénář pokračuje v 5. kroku hlavního scénáře

Alternativní scénář: zvolené zadání bylo typu „Obrázek.“

1. Scénář začíná ve 3. kroku hlavního scénáře.
2. Systém zobrazí formulář s prvky: obrázek, název, jazyk, možnost volby jiného obrázku
3. Dále scénář pokračuje v 5. kroku hlavního scénáře

Alternativní scénář: zvolené zadání bylo typu „Normalizace relačního schématu.“

1. Scénář začíná ve 3. kroku hlavního scénáře.
2. Systém zobrazí formulář, kde uživatel může upravovat následující atributy: název, jazyk, množinu atributů, množinu funkčních závislostí s možností přidávání či odebírání dalších.
3. Po dokončení modifikace hodnot uživatel může uložit změny pomocí tlačítka „Uložit“ nebo je odmítnout pomocí tlačítka „*Zpět.*“ Systém navíc dovoluje zobrazit ukázkové řešení zvolením tlačítka „*Ukázkové řešení.*“

2.2.2.4 UC4: Duplikace zadání

Případ užití umožňuje Uživateli vytvořit kopii zvoleného zadání pomocí tlačítka „*Duplikovat.*“ Systém duplikuje zadání a zobrazí ho v seznamu spolu s hláškou o úspěchu či chybě.

Aktéři: vyučující, garant.

2.2.2.5 UC5: Odstranění zadání

V systému existuje možnost testové zadání smazat, ale pouze tehdy, když žádná otázka nemá dané zadání přiřazené, v opačném případě je tlačítko „*Odstranit*“ neaktivní.

Aktéři: vyučující, garant.

Scénář:

1. Uživatel projde případem užití „UC1: Zobrazení seznamu zadání.“
2. Uživatel ze seznamu vybere zadání a klikne „*Odstranit.*“
3. Systém se zeptá na potvrzení
4. Uživatel zvolí „*Ok,*“ pokud chce volbu potvrdit nebo „*Cancel*“ v opačném případě.

2.2.2.6 UC6: Tvorba zadání

Případ užití umožní uživateli vytvořit nové zadání. Uživatel může vytvořit zadání pomocí textu, obrázku, diagramu nebo normalizace relačního schématu.

Aktéři: vyučující, garant.

Hlavní scénář:

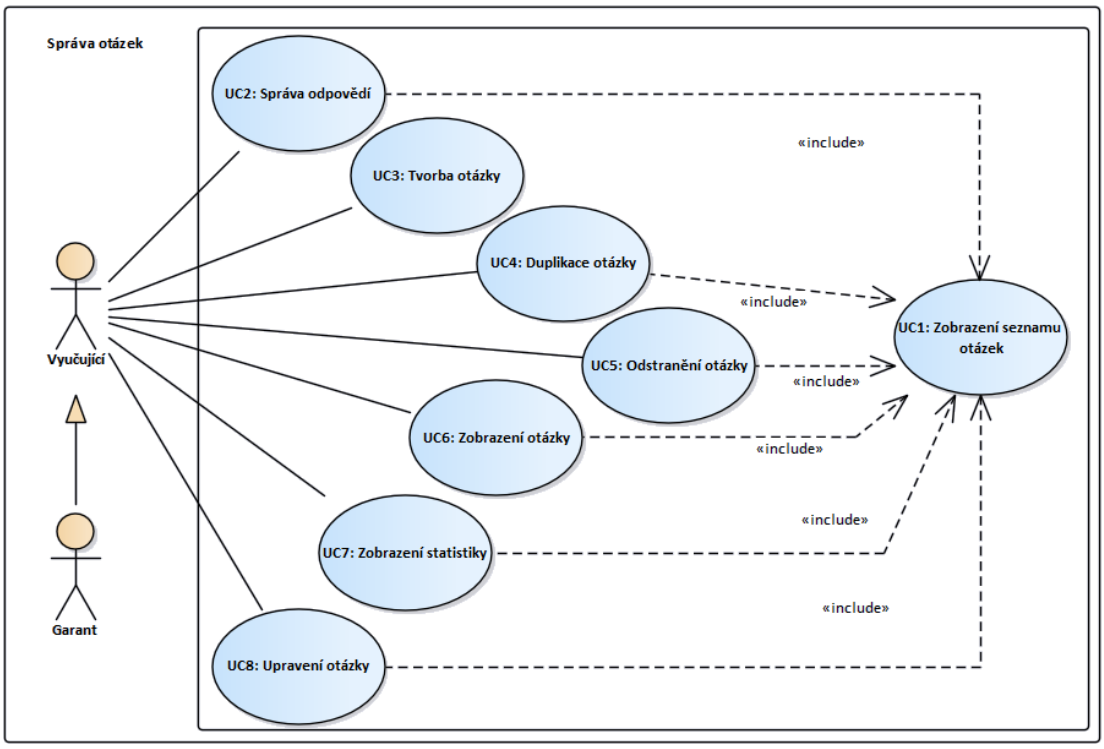
1. Případ užití začíná, když se uživatel rozhodne vytvořit nové zadání a zvolí v levém menu tlačítko „*Vytvořit zadání.*“
2. V menu se zobrazí typy zadání, které je uživatel schopen vytvořit.
3. Uživatel zvolí jeden z druhů zadání, na jehož základě se mu zobrazí odpovídající formulář.
4. Systém zobrazí formulář v závislosti na druhu zadání.
5. Uživatel vyplní formulář a zvolí „*Uložit*“ pro uložení zadání.
6. Systém provede žádanou operaci a zobrazí seznam zadání nebo chybu, pokud nastala.

Alternativní scénář: uživatel zvolil typ „Normalizace relačního schématu.“

1. Scénář začíná ve 3. kroku hlavního scénáře.
2. Systém zobrazí formulář s údaji: název, jazyk, množina atributů a množina funkčních závislostí.
3. Poté, co uživatel vyplní formulář, může kliknout na tlačítko „*Uložit*“ a systém uloží provedené změny nebo zvolit „*Ukázkové řešení.*“
4. Dále scénář pokračuje v 6. kroku hlavního scénáře

Alternativní scénář: uživatel klikl na tlačítko „Generovat zadání.“

1. Scénář začíná ve 2. kroku alternativního scénáře. Uživatel zvolil možnost automatického generování zadání.
2. Systém zobrazí formulář, kde lze nastavit množinu atributů a počet funkčních závislostí.
3. Uživatel vyplní formulář a zvolí „*Generovat zadání.*“
4. Systém vygeneruje zadání a scénář pokračuje ve 3. bodě alternativního scénáře.



Obrázek 2.3 Diagram případů užití: Správa otázek

2.2.3 Správa otázek

V této podkapitole jsou znázorněny a popsány případy užití pro současnou správa testových otázek (F2 2.4.1.2). Diagram případů užití lze nalézt na obrázku 2.3

2.2.3.1 UC1: Zobrazení seznamu otázek

Případ užití umožní uživateli zobrazit všechny již vytvořené otázky.
Aktéři: vyučující, garant.
Scénář:

1. Případ užití začíná, když se uživatel rozhodne podívat se na všechny otázky v systému.
2. Uživatel zvolí v levém menu tlačítko „*Zobrazit otázky.*“
3. Systém zobrazí tabulku, která obsahuje všechny vytvořené otázky.

2.2.3.2 UC2: Správa odpovědí

Systém umožňuje uživateli přidávat nebo odstraňovat odpovědi z otázky. Postup tvorby odpovědí se liší v závislosti na jejich druhu. Odpovědi můžou být následujících typů:„ SQL dotaz, dotaz v relační algebře, textová odpověď, diagram, normalizace relačního schématu, transformace a zaškrtávací odpovědi.“
Aktéři: vyučující, garant.
Scénář:

1. Uživatel projde případem užití „UC1: Zobrazení seznamu otázek.“

2. Ze seznamu vybere konkrétní otázku a zvolí tlačítko „*Správa odpovědí.*“
3. Systém zobrazí odpovídající formulář dle druhu odpovědi. Obecně tento formulář obsahuje tyto části: zadání, nastavení otázky, referenční odpověď, všechny odpovědi a možnost přidání nové odpovědi.
4. Uživatel vyplní textová pole nebo nakreslí diagram v závislosti na druhu odpovědi a klikne na „*Přidat odpověď.*“
5. Systém odpověď přidá a opětovně načte aktuální stránku, kde se tato odpověď objeví v sekci „Všechny odpovědi.“
6. Po dokončení přidávání odpovědí, může uživatel kliknout „*Zpět*“ a vrátit se k seznamu otázek.

2.2.3.3 UC3: Tvorba otázky

Uživatel může vytvářet nové otázky v portálu. Pokud otázka nemá přiřazené zadání a alespoň jednu referenční odpověď, pak ji systém zobrazuje jako nepoužitelnou.
Aktéři: vyučující, garant.
Scénář:

1. Případ užití začne, když se uživatel rozhodne vytvořit novou otázku.
2. Z levého menu vybere „Vytvořit otázku.“
3. Systém zobrazí formulář s údaji: název, úkol, obtížnost, kategorie, úroveň, jazyk, typ odpovědi a možnost zařadit otázku do demo testu.
4. Uživatel zadá všechny údaje a klikne na „*Uložit*“ nebo „*Uložit a spravovat zadání.*“
5. Pokud uživatel zvolil „*Uložit a spravovat zadání,*“ systém otázku uloží a zároveň přejde do vyplňování „UCX Přiřazování zadání,“ viz podkapitola 2.2.4

2.2.3.4 UC4: Duplikace otázky

Systém dovoluje uživateli vytvořit kopii konkrétní otázky.
Aktéři: vyučující, garant.
Počáteční podmínka: v systému existuje alespoň jedna otázka.
Scénář:

1. Uživatel projde případem užití „UC1: Zobrazení seznamu otázek.“
2. Uživatel v seznamu vyhledá otázku a klikne na „*Duplikovat.*“
3. Systém vytvoří novou otázku a následně zobrazí seznam otázek a odpovídající hlášku o úspěchu či neúspěchu.

2.2.3.5 UC5: Odstranění otázky

Případ užití umožňuje uživateli smazat otázku. Pokud je otázka použita v jakémkoli testu, tak nelze smazat a tlačítko „*Odstranit*“ bude neaktivní.
Aktéři: vyučující, garant.
Počáteční podmínka: otázka není použita v žádném testu.
Scénář:

1. Uživatel projde případem užití „UC1: Zobrazení seznamu otázek.“.
2. Uživatel vyhledá otázku a zvolí „*Odstranit.*“

3. Systém požádá o potvrzení.
4. Uživatel volbu potvrdí tlačítkem „*Ok*“ nebo odmítne tlačítkem „*Cancel.*“
5. Systém otázku odstraní a zobrazí seznam otázek.

2.2.3.6 UC6: Zobrazení otázky

Případ užití umožňuje uživateli zobrazit detailní informace o otázce.
Aktéři: vyučující, garant.
Počáteční podmínka: v systému existuje alespoň jedna otázka.
Scénář:

1. Uživatel projde případem užití „UC1: Zobrazení seznamu otázek.“.
2. Uživatel vyhledá otázku a zvolí „*Náhled.*“
3. Systém ukáže informace o otázce: popis, složitost, referenční odpovědi a zadání.

2.2.3.7 UC7: Zobrazení statistiky

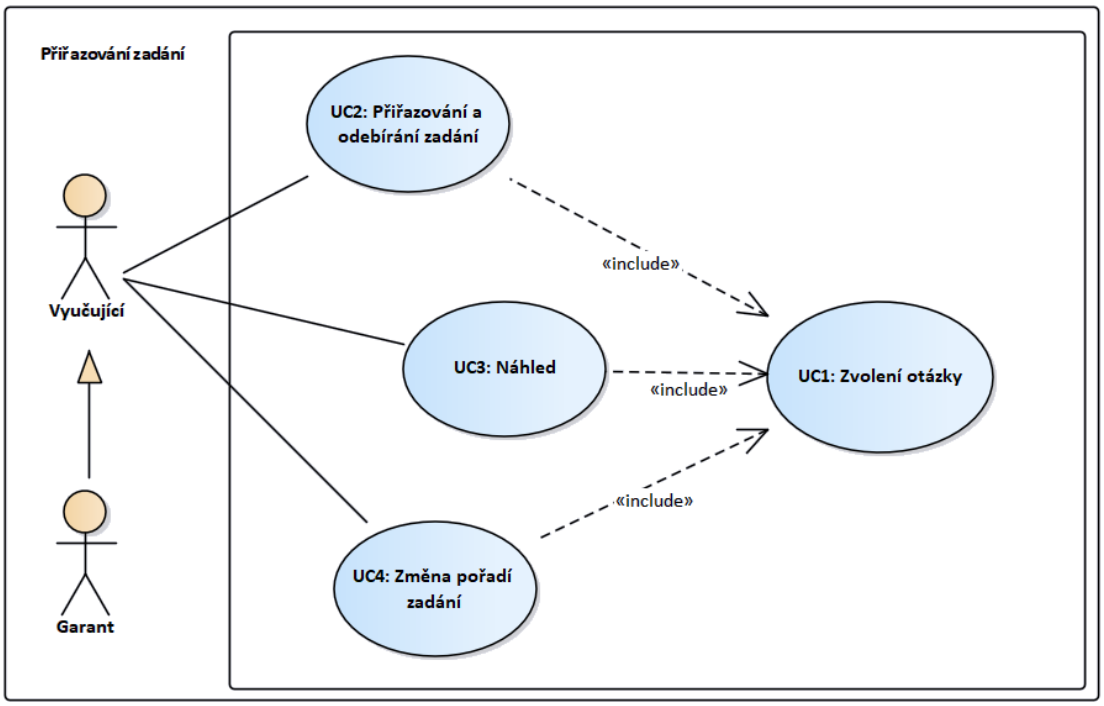
Uživatel je schopen monitorovat úspěšnost konkrétní otázky v testech. Statistika se sestaví ze správných a chybných odpovědí studentů ve všech testech, ve kterých byla otázka použita.
Aktéři: vyučující, garant.
Počáteční podmínka: v systému existuje alespoň jedna otázka.
Scénář:

1. Uživatel projde případem užití „UC1: Zobrazení seznamu otázek.“.
2. Uživatel vyhledá otázku a klikne na tlačítko s obrázkem statistiky.
3. Systém otevře modální okno, ve kterém se zobrazí graf úspěšnosti studentů u dané otázky.

2.2.3.8 UC8: Upravení otázky

Tento případ užití umožňuje měnit obecné údaje v otázce, např. název, popis apod. Uživatel nemůže měnit druh odpovědi, jestliže existuje alespoň jedna referenční odpověď.
Aktéři: vyučující, garant.
Počáteční podmínka: v systému existuje alespoň jedna otázka.
Scénář:

1. Uživatel projde případem užití „UC1: Zobrazení seznamu otázek.“.
2. Uživatel si zvolí otázku, kterou potřebuje modifikovat a zvolí tlačítko „*Upravit.*“
3. Systém zobrazí formulář umožňující změnit následující údaje: název, úkol, obtížnost, kategorie, úroveň, jazyk, typ odpovědi, pokud otázka dosud neměla žádnou přidanou odpověď a zařazení do demo testu.
4. Uživatel upraví potřebné údaje a klikne na „*Uložit.*“
5. Systém otázku uloží a zobrazí hlášku o úspěchu či chybě, potom zobrazí seznam všech otázek.
6. Pokud uživatel zvolil „*Uložit a spravovat zadání,*“ systém otázku uloží a přejde do případu užití „UC2: Přiřazování zadání,“ viz podkapitola 1.1.4. 2.2.4



Obrázek 2.4 Diagram případů užití: Přiřazování zadání

2.2.4 Přiřazování zadání

Tato podkapitola popisuje současný postup přiřazování zadání k otázce. Diagram případů užití je zachycen na obrázku 2.4

2.2.4.1 UC1: Zvolení otázky

Před přiřazováním či odebíráním zadání si uživatel musí vybrat u které otázky.
Aktéři: vyučující, garant.
Počáteční podmínka: v systému existuje alespoň jedna otázka.
Scénář:

1. Uživatel si zobrazí seznam otázek, vybere danou otázku a zvolí „*Správa zadání.*“
2. Systém zobrazí formulář, kde budou veškerá zadání a informace o zvolené otázce.

Alternativní scénář:

1. Uživatel klikne na „*Uložit a spravovat zadání*“ během tvorby otázky. (2.2.3.3)
2. Systém zobrazí formulář, kde budou veškerá zadání a informace o zvolené otázce.

2.2.4.2 UC2: Přiřazování a odebírání zadání

Případ užití dovoluje přidat nebo odebrat zadání z otázky.
Aktéři: vyučující, garant.
Počáteční podmínka: v systému existuje alespoň jedna otázka a jedno zadání.
Scénář:

1. Uživatel splní případ užití „UC1: Zvolení otázky.“
2. Uživatel vyhledá ze seznamu potřebné zadání a klikne „*Přidat zadání k otázce*“ nebo „*Odebrat zadání z otázky.*“
3. Systém zvolenou akci provede a hned zobrazí změny

2.2.4.3 UC3: Náhled

Systém umožňuje uživateli zobrazit informace o konkrétním zadání po kliknutí na tlačítko „*Náhled.*“

Aktéři: vyučující, garant.
Počáteční podmínka: v systému existuje alespoň jedna otázka a jedno zadání.

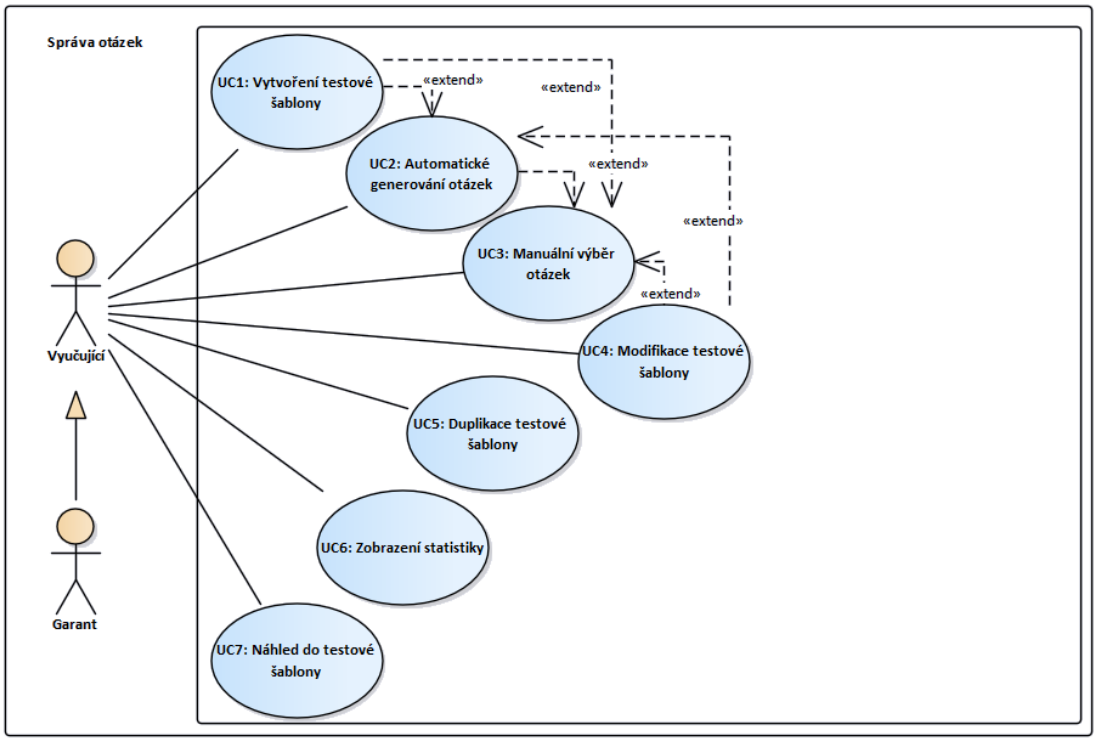
2.2.4.4 UC4: Změna pořadí zadání

Uživatel je schopen měnit pořadí zobrazování zadání studentům v testech pomocí šipek v seznamu zadání.

Aktéři: vyučující, garant.
Počáteční podmínka: v systému existuje alespoň jedna otázka a dvě zadání.

2.2.5 Správa testových šablon

V této podkapitole můžete vidět aktuální případy užití, které popisují postupy pro tvorbu a správu testových šablon (F5 2.4.1.5). Diagram případů užití je zobrazen na obrázku 2.5.



Obrázek 2.5 Diagram případů užití: Správa testových šablon

2.2.5.1 UC1: Vytvoření testové šablony

Případ užití umožní uživateli vytvořit novou testovou šablonu, kterou lze následně využít pro vytvoření testů.

Aktéři: vyučující, garant.

Scénář:

1. Případ užití začne, když uživatel chce vytvořit novou testovou šablonu a v levém menu klikne na „*Vytvořit testovou šablonu.*“
2. Systém zobrazí formulář s údaji: název, čas na vypracování, časové okno pro celý test, typ testu a maximální počet bodů.
3. Uživatel formulář vyplní a zvolí „*Automatické generování otázek*“ nebo „*Manuální výběr.*“
4. Systém uloží šablonu a přejde do „UC2: Automatické generování otázek“ nebo „UC3: Manuální výběr otázek“ v závislosti na volbě uživatele.

2.2.5.2 UC2: Automatické generování otázek

Uživatel může nastavit počty otázek v jednotlivých kategoriích, na jejichž základě systém vygeneruje otázky a přiřadí je k šabloně.

Aktéři: vyučující, garant.

Scénář:

1. Případ užití začne tím, že uživatel v předchozím případě užití zvolil „*Automatické generování otázek.*“
2. Systém zobrazí formulář, ve kterém lze nastavit počty otázek pro každou z kategorií. Pokud v systému není dostatečný počet otázek příslušné kategorie, je uživatel systémem upozorněn.
3. Uživatel nastaví počty otázek a klikne na „*Uložit.*“
4. Pokud druh testu byl demo test, systém uloží otázky a zobrazí seznam testových šablon. Jinak přejde do „UC3: Manuální výběr otázek.“

2.2.5.3 UC3: Manuální výběr otázek

Případ užití dovoluje přidávat nebo odebírat otázky z testu či šablony manuálně. Kromě toho může uživatel měnit pořadí otázek v testu.

Aktéři: vyučující, garant.

Scénář:

1. Případ užití je aktivován systémem po kliknutí uživatelem na tlačítko „*Manuální výběr*“ nebo po dokončení případu užití: „UC2: Automatické generování otázek,“ pokud typ testu nebyl demo test.
2. Systém zobrazí seznam otázek v šabloně a seznam všech dostupných otázek.
3. Uživatel může přidávat a odebírat otázky, měnit jejich pořadí nebo přejít do modifikace obsahu konkrétní otázky.
4. Po provedení potřebných změn uživatel klikne na tlačítko „*Uložit.*“
5. Systém uloží provedené změny a zobrazí seznam testových šablon.

2.2.5.4 UC4: Modifikace testové šablony

Případ užití umožňuje uživateli měnit parametry konkrétní testové šablony pouze v případě, pokud šablona nemá instanci testu (dosud nebyla použita v systému).

Aktéři: vyučující, garant.
Počáteční podmínka: neexistuje žádná instance testu, která by patřila zvolené šabloně.
Scénář:

1. Uživatel se rozhodne modifikovat šablonu, proto ze seznamu vybere tu potřebnou a zvolí tlačítko „*Upravit.*“
2. Systém zobrazí formulář, kde lze nalézt následující údaje: název, čas na vypracování, časové okno pro celý test, typ testu a maximální počet bodů.
3. Uživatel upraví hodnoty a klikne na „*Manuální výběr.*“
4. Systém uloží provedené změny a přejde do „UC3: Manuální výběr otázek.“

2.2.5.5 UC5: Duplikace testové šablony

Systém dovoluje uživateli duplikovat testovou šablonu pomocí tlačítka „*Duplikovat*“ v seznamu šablon. Šablona bude duplikována včetně všech přiřazených otázek.

Aktéři: vyučující, garant.
Počáteční podmínka: v systému existuje nejméně jedna validní testová šablona.

2.2.5.6 UC6: Zobrazení statistiky

Případ užití umožňuje uživateli podívat se na úspěšnost studentů v testech, které byly vytvořeny z dané šablony.

Aktéři: vyučující, garant.
Počáteční podmínka: v systému existuje nejméně jedna testová šablona.
Scénář:

1. Uživatel se rozhodne zobrazit si statistiku úspěšnosti studentů v rámci jedné šablony, proto ze seznamu šablon vybere tu, která ho zajímá a klikne na tlačítko s obrázkem statistiky.
2. Systém statistiku vypočítá a zobrazí graf úspěšnosti studentů.

2.2.5.7 UC7: Náhled do testové šablony

Uživatel má možnost si prohlédnout konkrétní testovou šablonu a zjistit, jaké otázky obsahuje testová šablona a měnit jejich pořadí zobrazení v testu.

Aktéři: vyučující, garant.
Počáteční podmínka: v systému existuje nejméně jedna testová šablona.
Scénář:

1. V seznamu testových šablon uživatel vyhledá šablonu a zvolí „*Náhled.*“
2. Systém zobrazí okno, které bude obsahovat seznam přiřazených otázek k šabloně.
3. Uživatel může pořadí otázek v testu libovolně měnit a také nastavovat opravujícího u konkrétní otázky. Provedená změna se hned projeví po kliknutí na jedno z tlačítek.

2.3 Nový backend

V této kapitole se budu zabývat změnami nového backendu, které je potřeba zavést do nového návrhu uživatelského rozhraní. Zaměřím se na novou funkcionalitu a změny oproti stávajícímu řešení. Zejména na ty části, které jsou pro vytvoření nového uživatelského rozhraní potřebné.

Bc. Andrii Plyskach se ve své bakalářské práci zabýval návrhem a implementací prototypu backendu testového modulu za cílem zjednodušit práci vývojářům a usnadnit další rozvoj nebo změny v portálu. Práce je tedy zaměřena především na návrh databáze, způsob práce s daty, provázání se zbytkem systému a jednoduchou rozšiřitelnost. Detaily backend části testového modulu se v této práci nebudu zabývat, pro pochopení této části doporučuji si přečíst zmíněnou práci. [2]

2.3.1 Zadání

2.3.1.1 Balíčky

Nově se budou používat balíčky zadání, která budou obsahovat dílčí zadání a tvořit jeden celek. To zjednoduší vyhledávání zadání, které se skládá z více částí. Například pro otázku typu *SQL* se použije textové zadání a zadání nového typu *Databáze*. Balíček, jako celek nahradí testové zadání v případech užití v podkapitole „Správa zadání“ 2.2.2.

2.3.1.2 Nový typ

Nově bude přidáno zadání typu *Databáze*, které bude obsahovat připojení k databázi, které se pak bude využívat pro vyhodnocení otázek, jejichž odpovědi jsou typu *RA* nebo *SQL*.

V současném řešení je připojení k databázi uložené u otázek typu *RA a SQL*. Při změně zadání a dané databáze je tedy potřeba provést změnu u všech otázek s daným zadáním, oproti novému řešení, kde stačí změnu udělat jen jednu.

2.3.2 Otázky

U otázek budou nově přidané štítky, které umožní rozdělování otázek na libovolně nastavené skupiny.

Na obtížnost, kategorii a úroveň stávajícího řešení se použijí štítky pro zachování již existujících otázek.

2.3.3 Štítky

Jak už jsem zmínil, budou přidané štítky, které umožní jednodušší práci s otázkami.

Štítky budou rozdělené do skupin, ty budou mít název, popis a disjunktnost. Disjunktivní skupina umožní přidat jenom jeden štítek k dané otázce. Disjunktnost má využití u skupin štítků jako je například obtížnost. Otázka nemůže být zároveň lehká i těžká.

2.3.3.1 Správa štítků

V této podkapitole jsou popsány operace prováděné se štítky a štítkovými skupinami. Případy užití jsou krátké, proto operace popisuji jen slovně.

Štítková skupina má název, popis a disjunktnost.

Štítek má název, popis a skupinu, ve které se nachází.

Při vytváření je potřeba jednotlivé vlastnosti vyplnit, následně po vytvoření půjdou zobrazit, upravit a smazat.

UC1: Vytvoření štítkové skupiny

UC2: Vytvoření štítku
UC3: Upravení štítkové skupiny
UC4: Upravení štítku
UC5: Odstranění štítkové skupiny
UC6: Odstranění štítku
UC7: Zobrazení štítkové skupiny
UC8: Zobrazení štítku
UC9: Přiřazení štítku k otázce - Uživatel může k otázce přiřadit několik štítků.
Aktéři: vyučující, garant.
Počáteční podmínka: v systému existuje nejméně jeden štítek.
Scénář:

1. Případ užití začíná při úpravě nebo vytváření otázky.
2. Uživatel vybere jednotlivé štítky, které chce k otázce přiřadit

2.3.4 Testové šablony

Z testové šablony půjdou nově generovat dva druhy testů, a to statický a dynamický test. Pro tvorbu statického testu budou použity přiřazené otázky a test dynamický se bude generovat za pomoci štítků.

K testovým šablonám tedy nově půjdou přiřadit jak otázky, tak štítky, a to který test se z dané šablony vygeneruje, bude na volbě učitele. Statická a dynamická část bude aktuálně oddělená z důvodu limitace backendu. Do budoucnosti je možné provést úpravu, která by umožnila vytvořit šablonu, která generuje test současně jak ze statických otázek, tak i z otázek vybraných pomocí štítků, tím se ale ve své bakalářské práci zabývat nebudu.

2.4 Seznam požadavků

V této kapitole uvedu všechny funkční a nefunkční požadavky, které jsou potřebné zavést do nového řešení frontendu. Tyto požadavky vycházejí z analýzy provedené v předchozích kapitolách a obsahují všechny potřebné funkce pro správnou funkčnost nového uživatelského rozhraní. Jedná se tedy o požadavky, které zahrnují jak funkčnosti ze současného systému tak i požadavky vytvořené novým backendem.

Požadavky z pohledu softwarového inženýrství jsou specifikací toho, co by mělo být implementováno, popisují veškeré chování systému nebo jeho části. Měly by jasně a jednoznačně definovat konkrétní omezení, hranice, kde by měl softwarový proces skončit. Nestanovení jasné hranice může způsobit problémy se zákazníkem, který nebude chtít platit za něco, co dle něj již bylo zaplaceno. Z tohoto důvodu se doporučuje jasně a jednoznačně vymezit to, co se udělat má a co se implementovat nebude.

Proto by každý funkční požadavek měl obsahovat:

- Název,
- Zkratka - ta usnadňuje odkazování na daný požadavek.
- Popis - nejdůležitější část požadavku (měl by být jednoznačný s konkrétní činností, mít vymezené podmínky a omezení, které se na požadavek kladou).
- Priorita,
- Složitost. [7]

Softwarové požadavky můžeme dělit dle následujících typů:

- Funkční - požadavky popisující funkce systému, jeho chování.
- Nefunkční - požadavky zabývající se obecnými vlastnostmi systému. Typicky: *přenositelnost, bezpečnost, udržovatelnost, spolehlivost, výkon, flexibilita.*
- Požadavky na doménu - liší se v závislosti na konkrétní doméně. [8]

2.4.1 Funkční požadavky

Zde vypisuji všechny funkční požadavky pro nový frontend. Jelikož se jedná o vytvoření nového uživatelského rozhraní, je většina funkčních požadavků převzata ze stávající aplikace. Nové požadavky jsou zavedeny novým backendem. Nové požadavky jsou konkrétně funkční požadavky F3 a F4 (2.4.1.3, 2.4.1.4). Ostatní funkční požadavky mají stejný účel jako v současné aplikaci, ale budou řešeny jiným způsobem s ohledem na nový backend, změny způsobené novým backendem jsou zavedeny v popisu jednotlivých požadavků.

2.4.1.1 F1: Správa zadání

Systém umožňuje uživateli vytvářet, editovat, případně mazat testová zadání (v novém řešení tzv. balíčky zadání viz. 2.3.1.1), která se budou skládat ze zadání těchto druhů: text, obrázek, diagram, normalizace, databáze. Poslední druh zadání v současném systému neexistoval. Tento druh zadání bude obsahovat připojení, které se pak bude využívat pro vyhodnocení otázek, jejichž odpovědi jsou typu RA nebo SQL.

Priorita: vysoká
Složitost: střední

2.4.1.2 F2: Správa otázek

Systém umožňuje uživateli vytvářet, editovat, případně mazat otázky. Otázky už nebudou obsahovat některé z atributů, např. připojení, kategorii a úroveň. Místo těchto vlastností budou k otázkám přiřazovány štítky, na základě kterých se bude provádět rozdělení do skupin, vyhledávání apod. Otázky budou rovnou při vytvoření přiřazeny k zadání(balíčku zadání). Součástí otázek jsou také jejich odpovědi, které bude možné vytvářet, editovat, měnit jejich pořadí a mazat. Otázka může mít pouze odpovědi jednoho typu. Typy odpovědí budou: *text, radio list, checkbox, diagram, RA, SQL, transformace a normalizace relačního schématu*

Priorita: vysoká
Složitost: vysoká

2.4.1.3 F3: Správa štítků a skupin štítků

Štítky umožní rozdělování otázek dle libovolného vymyšleného druhu. Uživatel bude schopen tyto štítky vytvářet, modifikovat a mazat. Každý štítek musí být ve štítkové skupině, která může být disjunktní nebo ne, což se pak bude využívat během generování otázek. Disjunktnost skupiny znamená možnost přiřazení pouze jednoho štítku z dané skupiny k otázce. Pokud skupina disjunktní není, můžeme přiřadit více štítků. Každý štítek nebo jejich skupina bude obsahovat název a popis, čímž poznáme, k čemu slouží.

Priorita: vysoká
Složitost: vysoká

2.4.1.4 F4: Přiřazení štítků

Systém umožní uživateli jednotlivé štítky k otázkám přiřazovat nebo odebírat. Libovolná otázka může mít více štítků, štítek může být přiřazen u mnoha otázek. Otázka, která nemá přiřazený žádný štítek je neplatnou. Následně tuto otázku nelze použít při tvorbě testové šablony či testu.

Priorita: vysoká
Složitost: střední

2.4.1.5 F5: Správa testových šablon

V systému bude uživatel schopen vytvářet, upravovat, duplikovat a mazat testové šablony. Šablona bude obsahovat *název, autora, dobu trvání obecného testu, dobu trvání studentského testu, celkový počet bodů, jazyk, otázky pro tvorbu statického testu a seznám štítků pro tvorbu dynamického testu.* Z testové šablony pak bude možné vyrobit jeden ze dvou obecných testů. Před vytvořením testu systém zkontroluje správnost testové šablony. Pokud se budeme snažit vytvořit statický test, šablona musí obsahovat alespoň jednu přiřazenou otázku. V případě dynamického testu, musí šablona obsahovat přiřazené štítky.

Priorita: vysoká
Složitost: vysoká

2.4.2 Nefunkční požadavky

2.4.2.1 N1: Volba technologií

Současný systém je z velké části naprogramován v jazyce PHP na základě frameworku Nette, s pomocí Latte šablon pro uživatelské rozhraní. V již refaktorovaných částech systému se však přechází na framework Vue.js. Z tohoto důvodu se bude používat framework Vue.js a jeho knihovny potřebné pro funkčnost systému. Vue.js je navíc zcela oddělen od datové vrstvy a s daty pracuje pomocí API, což zajišťuje maximální nezávislost komponent.

Priorita: vysoká
Složitost: střední

2.5 Uživatelské rozhraní

Uživatelské rozhraní je souhrn způsobů, jakými uživatel ovlivňuje chování systému. Hlavními aspekty pro hodnocení uživatelského rozhraní je jednoduchost a přehlednost.

Je to prostředník, který zprostředkovává komunikaci mezi člověkem a počítačem nebo informačním systémem. Jeho vzhled, funkčnost, přehlednost nebo chaos ovlivňují, zda se uživatel příště vrátí, nebo ne. Návrh, struktura a následná realizace jsou důležitým faktorem vzniku jakéhokoli informačního systému.

Uživatelské rozhraní tvoří následující 3 složky:

- Fyzická složka: vstupní, výstupní zařízení a nástroje zpětné vazby,
- konceptuální složka: mechanismy použity při komunikaci mezi člověkem a počítačem. Např. dotazovací jazyky,
- perceptuální složka: vnímání objektů na obrazovce, jejich uspořádání a použití barev. [9]

Dobře navržené uživatelské rozhraní by mělo být:

- Jednoznačné,
- stručné,

- responzivní,
- konzistentní,
- efektivní,
- pohodlné,
- lehce napravitelné.[10]

Jednou z hlavních vlastností uživatelského rozhraní je použitelnost. Použitelnost souvisí s ovládáním, srozumitelností a přehledností. Zda se uživatel orientuje snadno, rychle a bez námahy. Některé z principů použitelnosti:

- Pravidlo 10 minut - uživatel by měl být schopen osvojit si základy práce se systémem do 10minut.
- Pravidlo 3 kliknutí - pokud se nachází hledaná informace na stránce, měl by jí uživatel najít do 3 kliknutí.
- Pravidlo 2 sekund - doba odezvy systému by měla být v rozmezí 2 sekund. [9]

2.5.1 Hlavní problémy současného uživatelského rozhraní

Způsobu, jakými analyzovat uživatelské rozhraní je několik. Například pomocí rozhovorů s uživateli systému, pomocí dotazníku, pomocí uživatelských scénářů nebo při skupinové diskuzi. Pro účely této práce jsem zvolil kvalitativní výzkum s vedoucím projektu. Vedli jsme diskuze nad jednotlivými problémy a změnami v uživatelském rozhraní. První diskuze proběhla po krátké analýze současného řešení, pro kterou jsem si připravil první návrh, ve kterém ještě nebyla zavedena nová funkcionalita. Při diskuzi jsme prošli návrh a představil jsem svojí vizi, jak by mohl nový systém vypadat. Následně jsme vedli další diskuze nad každou další verzí návrhu a postupně jsem se dostal k řešení, s kterým jsem byl spokojený a začal naimplementovat. Diskuze byli různě dlouhé, občas se jednalo pouze o textovou konverzaci, při návrhu s více změnami potom probíhali hovory přes internet z důvodu pandemické situace.

V této kapitole se budu zabývat hlavními nedostatky uživatelského rozhraní, na které jsem přišel za celou dobu iterativních diskuzí nad jednotlivými návrhy. Kapitola bude rozdělena na jednotlivé části potřebné pro vytvoření testu (*zadání, otázky a testové šablony*). V každé části sepíšu problémy a v kapitole 3 se budu zabývat jejich řešením nebo odstraněním, popisuji v ní také detailněji postup schůzek, na kterých se problémy řešily.

Jedním velkým problémem na celé testové části systému jsou zbytečně časté přesměrování uživatele na jinou stránku. Načítání stránky nějakou dobu trvá a při vytváření celého testu (velké množství zadání, otázek, odpovědí atd.) čas strávený u načítání stránky není malý. V další řadě samotné podmenu pro testovou část systému je zbytečně velké, v současném stavu se zde nachází více jak 15 odkazů, což snižuje přehlednost.

2.5.1.1 Zadání

Před vytvářením zadání si musí uživatel dopředu rozmyslet, který typ zadání chce vytvořit a ten musí vybrat v rozbalovacím podmenu stránky.

Po vytvoření zadání je uživatel přesunut na seznam zadání. To neumožňuje rychlou tvorbu více zadání, nebo ihned vytvářet otázky k danému zadání.

2.5.1.2 Otázky

Při vytváření otázky nelze na stejné stránce přiřadit zadání a vytvořit odpovědi k otázce.

Po vytvoření otázky se může uživatel rozhodnout přejít na seznam otázek, nebo přidat zadání. Při volbě přidání zadání je však opět přesměrován na jinou stránku.

Možnost přidávání odpovědí je také na samostatné stránce, na kterou se uživatel musí proklikat ze seznamu otázek, nebo ze stránky, kde se přidávají k otázce zadání.

2.5.1.3 Testové šablony

Testové šablony v současném řešení představují v podstatě nespuštěný test. Po vytvoření instance testu už nejde šablona upravovat, pokud ji chce uživatel znovu použít a upravit tak si musí vytvořit kopii a editovat ji. Tento problém byl zmíněný i v práci Bc. Andriiho Plyskache a v rámci backendu byl vyřešen.

Pro přidávání otázek do šablony je opět použita jiná stránka.

Kapitola 3
Návrh

V této kapitole se zabývám návrhem nového uživatelského rozhraní testové části portálu DBS ze strany učitele. První částí návrhu GUI byl papírový model neboli wireframe. Wireframe definuje textový i grafický obsah, rozmístění funkčních prvků a také navigaci či tlačítka. Grafický obsah je tvořen pouze pomocí čar a nepoužívají se ani barvy. Papírový model následně umožnuje konzultovat návrh uživatelského rozhraní se zadavatelem před samotnou implementací, která je časově náročná na změny. Provádění změn v papírovém modelu je jednoduché a rychlé, zároveň po vytvoření a otestování modelu zadavatelem se může grafik plně soustředit na svoji práci a nemusí řešit rozmístění prvků a obsah stránky. [11]

Pro tvorbu papírového modelu jsem využil program Balsamiq Wireframes [12]. Jedná se o nástroj pro rychlou tvorbu prototypů GUI, bez nutnosti psaní programového kódu. Každý jednotlivý návrh jsem iterativně konzultoval (testoval se zadavatelem) a následně pozměnil. Celkem proběhlo zhruba pět iterací, ve kterých jsem našel chybu, nebo jsem byl upozorněn na nějaký nedostatek. Každou iterací se návrh zlepšoval a přibližoval řešení, které už bylo vhodné implementovat. Výsledné návrhy jsou vidět v kapitole 3.3. Některé iterace návrhů jsou uvedené v příloze A.

3.1 První verze návrhu

První návrh proběhl již v průběhu předmětu BI-SP2, kdy jsem se zabýval vývojem na jiné části systému. Hned poté co jsem si vybral toto téma bakalářské práce, ještě před analýzou nových funkčností jsem navrhnul jednodušší stránku pro vytváření zadání spolu s možností rovnou vytvořit otázky a k otázkám odpovědi.

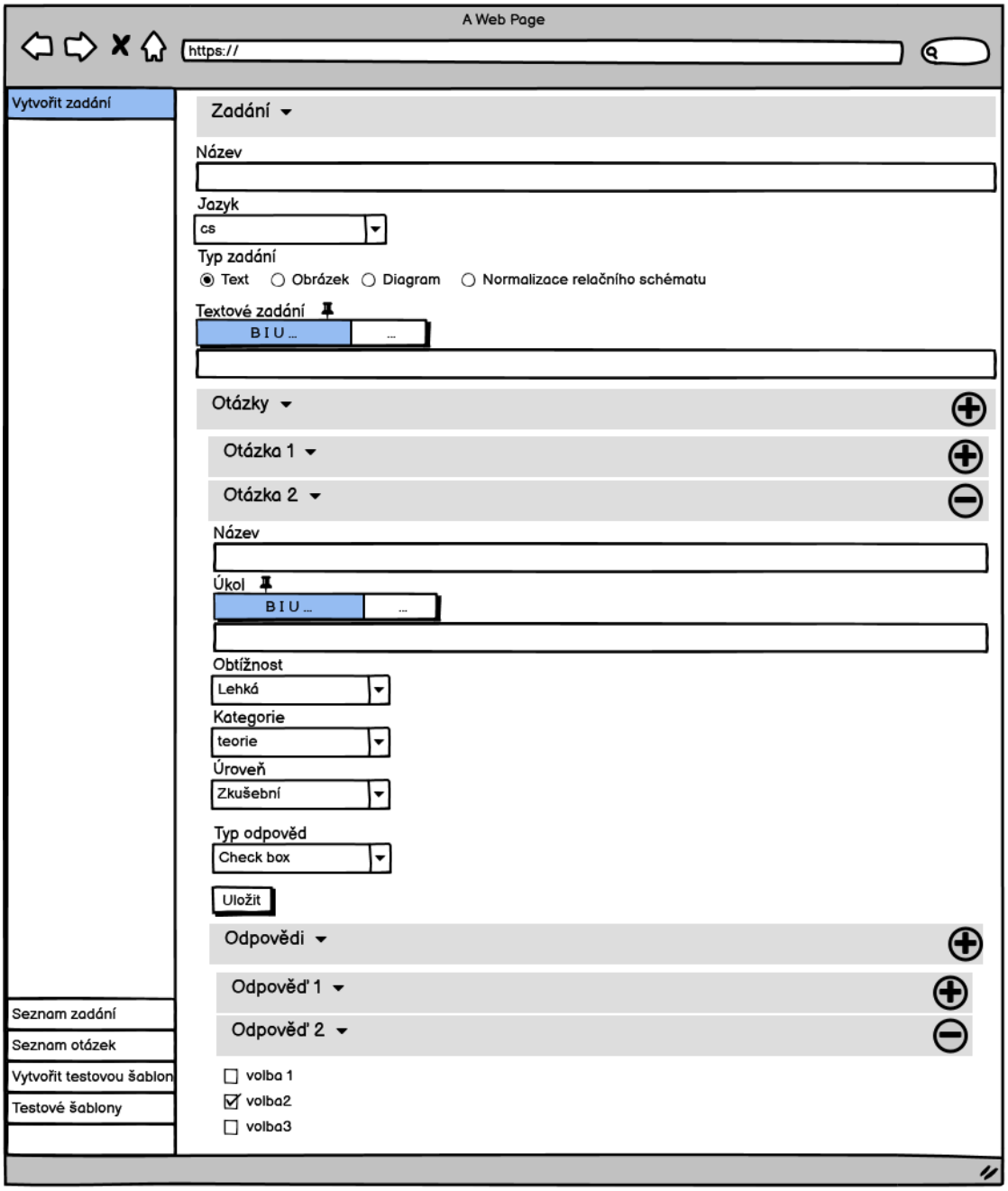
Pro jednotlivé komponenty jsem se rozhodl použít rozbalovací boxy, na kterých jsou vidět hlavní informace již před rozbalením a po rozbalení se zobrazí všechny informace s možností úprav. Použití rozbalovacích boxů umožnuje uživateli skrýt informace, které už dále nepotřebuje vidět. Výhodou rozdělení stránky na jednotlivé komponenty je také možnost jejich znovu použitelnosti.

Vytváření zadání, otázek a odpovědí na jedné stránce vyřešilo jeden z hlavních problémů, a to časté přesměrování uživatele na jiné stránky. Tímto způsobem se může učitel plně zaměřit na vymýšlení otázek k danému zadání, bez přerušení vyvolaného načítáním nové stránky.

Tato verze návrhu, která je vidět na obrázku 3.1 sloužila jako hlavní kostra pro vytvoření návrhu zbylých stránek a jako základ, na kterém jsem v průběhu analýzy prováděl změny, z důvodu zavedení nových funkcionalit systému.

Jak už jsem zmiňoval, první návrh jsem vytvářel před analýzou a zaměřil jsem se jenom na vytváření otázek, u kterých se vyplňovali informace na třech samotných stránkách, u ostatních

komponent testu jsem v té době velký problém neviděl. Návrhem zbylých stránek jsem se tedy zabýval až po důkladné analýze stávajícího řešení i nových funkcí nového backendu.

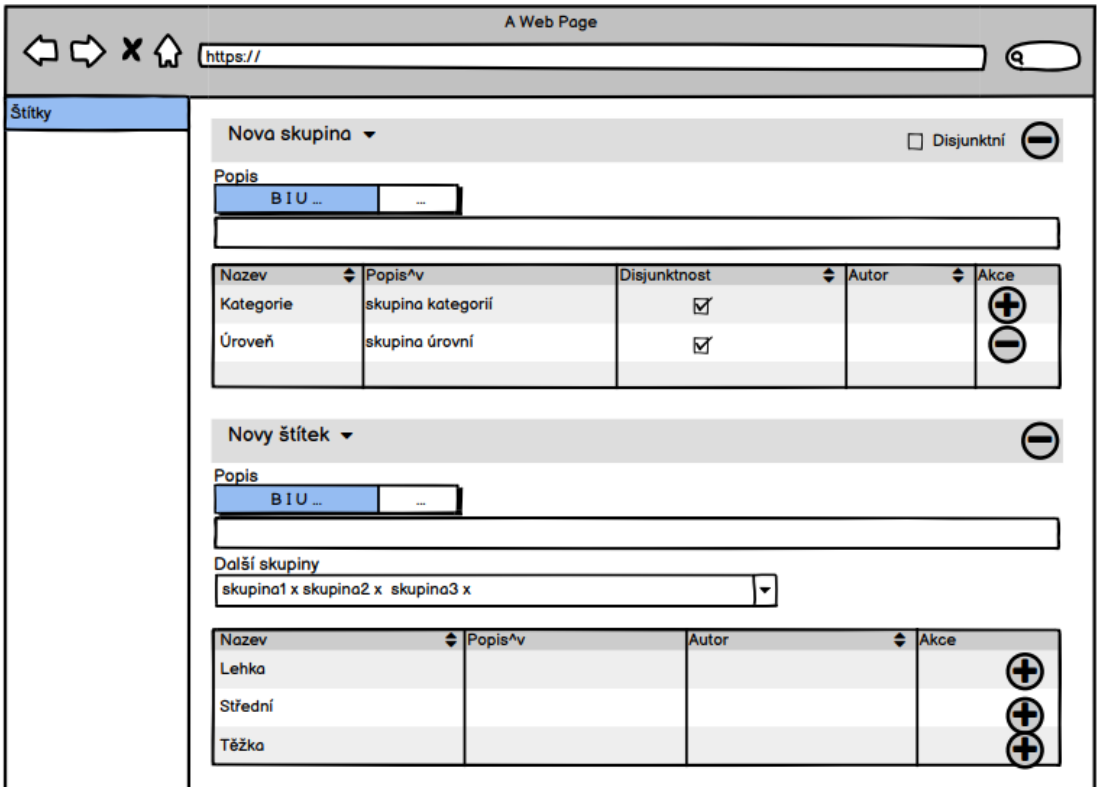


Obrázek 3.1 Wireframe: První verze návrhu stránky *Vytvořit zadání*

3.2 Následující průběh návrhů

Jak už jsem popisoval dříve, postup návrhu probíhal iterativně za pomocí diskuzí se zadavatelem. V této kapitole stručně popisuji jednotlivé změny po daných diskuzích. První návrh a následná diskuze byla rozsáhlá, popsal jsem ji tedy v sekci 3.1. Výsledné návrhy potom důkladně popisuji v dalších sekcích.

V druhé verzi jsem se zaměřil na přidání nových funkcionalit do již navržené stránky pro vytváření zadání, otázek a odpovědí, kterými byli zejména štítky a databázové zadání. Tato verze je uvedena v příloze A.1. Připravil jsem také stránku pro vytváření štítkových skupin a štítků samotných, která je vidět na obrázku 3.2.

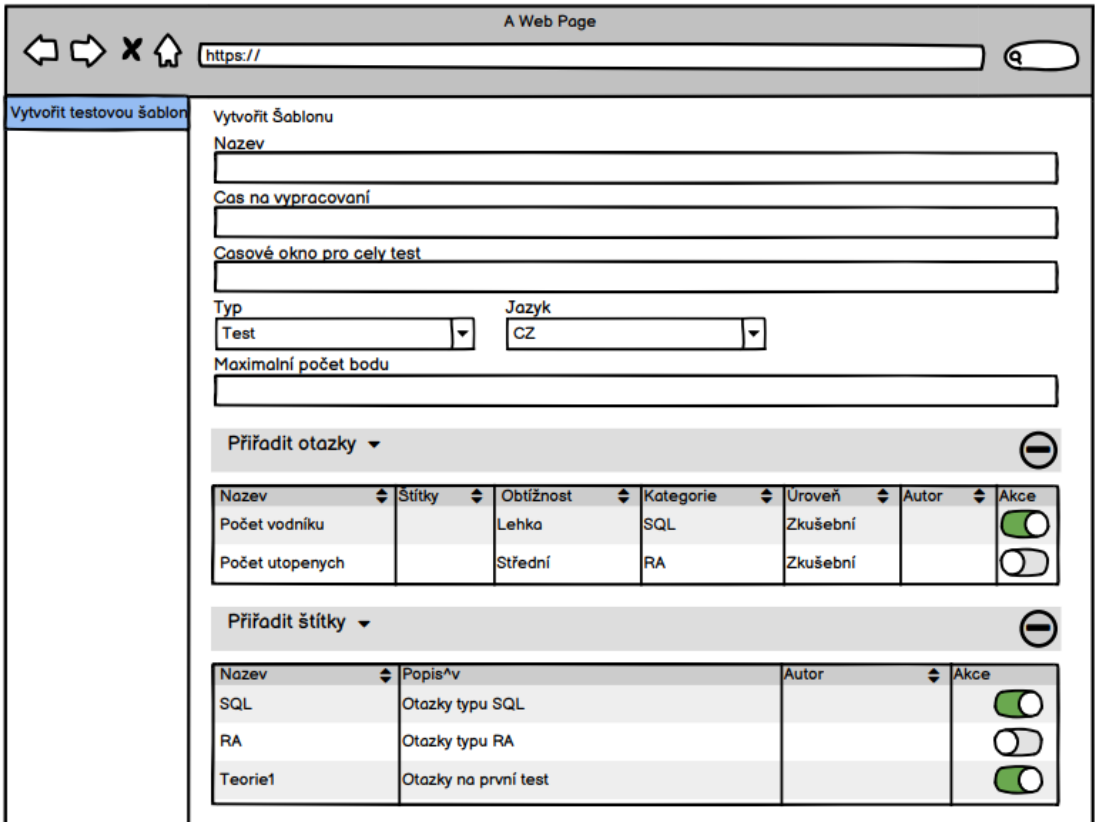


Obrázek 3.2 Wireframe: Stránka pro vytváření štítkových skupin a štítků

Na třetí diskuzi jsem si připravil stránku pro vytváření testových šablon, která je na obrázku 3.3. Měl jsem také připravenou novou verzi stránky pro vytváření zadání, ve které jsem se snažil umožnit rychlé zobrazení jakéhokoliv zadání s možností přiřadit ho k zobrazené otázce. Po důkladné diskuzi jsem tento směr návrhu ale nepoužil, jelikož začínal být nadměrně složitý a nejasný. Tento návrh je uvedený v příloze A.2.

Pro čtvrtou schůzku jsem doladil stránku pro vytváření testových šablon, která je popsaná ve výsledném návrhu 3.3.4. Doladil jsem také stránku pro vytváření zadání, otázek a odpovědí, kde oproti minulému návrhu proběhli znatelné změny, které blíže popisuji ve výsledném návrhu pro stránku vytvořit zadání 3.3.1.

Při páté diskuzi jsem se zadavatelem důkladně prošel všechny návrhy. V některých jsem provedl menší změny, které jsem na schůzce zmínil. V druhé části schůzky jsem se pak zaměřil na seznamy jak zadání a otázek tak i testových šablon. Pro stránky se seznamy jsem papírové návrhy nevytvářel, jenom jsem sepsal atributy a akce, které budou v seznamech zobrazeny. Více popisuji v další kapitole u daných seznamů 3.3.2 a 3.3.5.



Obrázek 3.3 Wireframe: Stránka pro vytváření testové šablony

3.3 Výsledný návrh

V této kapitole popisuji návrhy pro jednotlivé stránky, ve stavu, před tím, než se začalo implementovat. Jak už jsem zmiňoval, verzí papírových modelů jsem měl několik a po konzultacích a po nalezení nedostatků jsem je upravoval, až jsem se dostal k výslednému modelu, který už byl dostatečně upraven a teoreticky otestován. V průběhu implementace probíhali v návrhu ještě malé změny, které jsem již do papírových modelů zpětně nezaváděl. Podkapitoly se budou zabývat jednotlivými stránkami.

3.3.1 Vytvořit zadání

Tato stránka se zaměřuje na funkční požadavky F1, F2 a F4 (správa zadání, otázek a přiřazení štítků 2.4.1). První sbalovací box slouží k vytvoření a úpravě balíčku zadání. Nastavuje se název a jazyk, v kterém bude celý balíček. Dále jsou boxy s dílčími částmi zadání. Výběr druhu zadání je umožněn pomocí přepínačů, podle výběru se pak zobrazí pole pro potřebné informace k danému druhu. V hlavičce boxu je editovatelný volitelný název zadání a na pravé straně je tlačítko pro odstranění. K přidání dalších zadání slouží tlačítko „*Přidat zadání*“ pod vytvořenými zadáními.

Další částí stránky je rozbalovací box s otázkami. V boxu „Otázky“ jsou všechny otázky vytvořené k danému balíčku zadání. V hlavičce boxů je vidět název otázky, na pravé straně boxu je pak opět tlačítko pro odstranění. Po rozbalení boxu je možné všechny výše uvedené informace k otázce upravit.

V poslední části otázky jsou odpovědi, kde uživatel musí nejprve vybrat typ odpovědi a až poté může jednotlivé odpovědi přidat a vyplnit. Hlavičky odpovědí obsahují samotnou odpověď a tlačítka pro odstranění, posouvání pořadí odpovědí a dále možnost nastavení správnosti otázky a referenční odpověď (tyto volby záleží na typu odpovědi). V poslední řadě bude hlavička používat barvy pro přehlednost. Správné otázky budou zabarvené zeleně, špatné červeně a referenční odpověď modře. Výsledná verze je vidět na obrázku 3.4.

3.3.2 Seznam zadání

Tato stránka slouží k zobrazení všech vytvořených balíčků zadání s otázkami v systému. Jedná se o filtrovatelný seznam s možností řazení. Sloupce tabulky obsahují název, jazyk, autora a akce pro úpravu, duplikaci a odstranění balíčku. Tato stránka se téměř neliší od stránky seznamu zadání starého řešení, pouze byla předělána s použitím Vue.

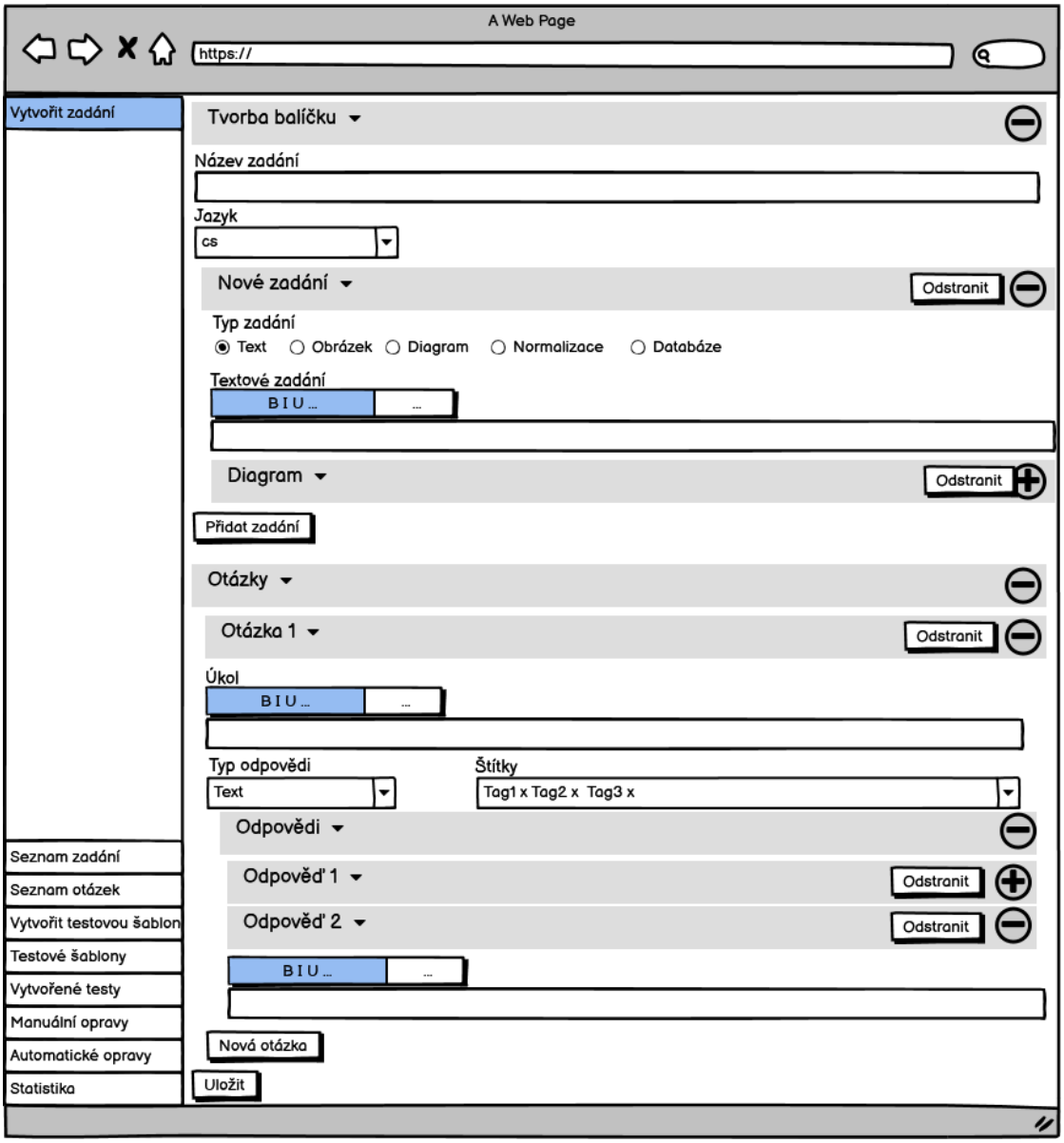
3.3.3 Štítky

Tato stránka se zaměřuje na funkční požadavek F3 (správa štítků a skupin štítků 2.4.1.3). V prvním boxu s názvem *skupiny* je další box s možností vytvořit novou štítkovou skupinu a pod tímto boxem je filtrovatelný seznam všech vytvořených skupin přímo s možností editace názvu, popisu a disjunktnosti. V akcích jsou pak tlačítka pro zobrazení štítků v dané skupině, spolu s tlačítkem na odstranění skupiny.

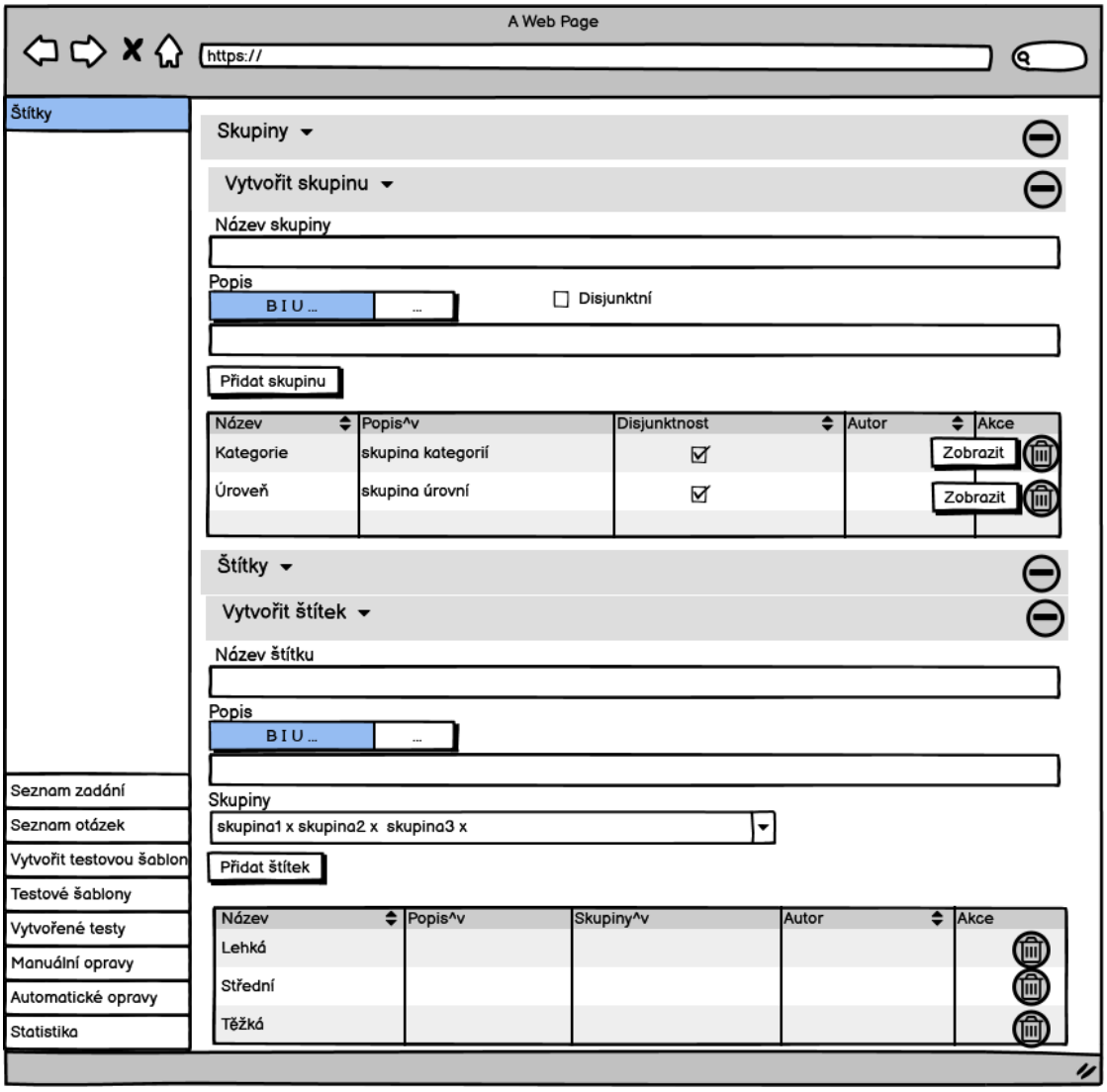
Další box se věnuje samotným štítkům. Opět je zde volba vytvoření nového štítku a seznam štítků. Seznam je opět editovatelný a filtrovatelný. Stisknutí tlačítka „Zobrazit štítky“ u některé ze skupin pouze vyvolá filtr pro danou skupinu. Výsledný návrh je na obrázku 3.5

3.3.4 Vytvořit testovou šablonu

Na této stránce probíhá vytváření testových šablon. Horní část vytváření testové šablony zůstala stejná. Uživatel musí vyplnit název, čas na vypracování, časové okno pro celý test, typ, jazyk a maximální počet bodů. Dále si může vybrat, zda chce vytvářet statický nebo dynamický test.



Obrázek 3.4 Výsledný návrh: stránka pro vytváření zadání, otázek a odpovědí

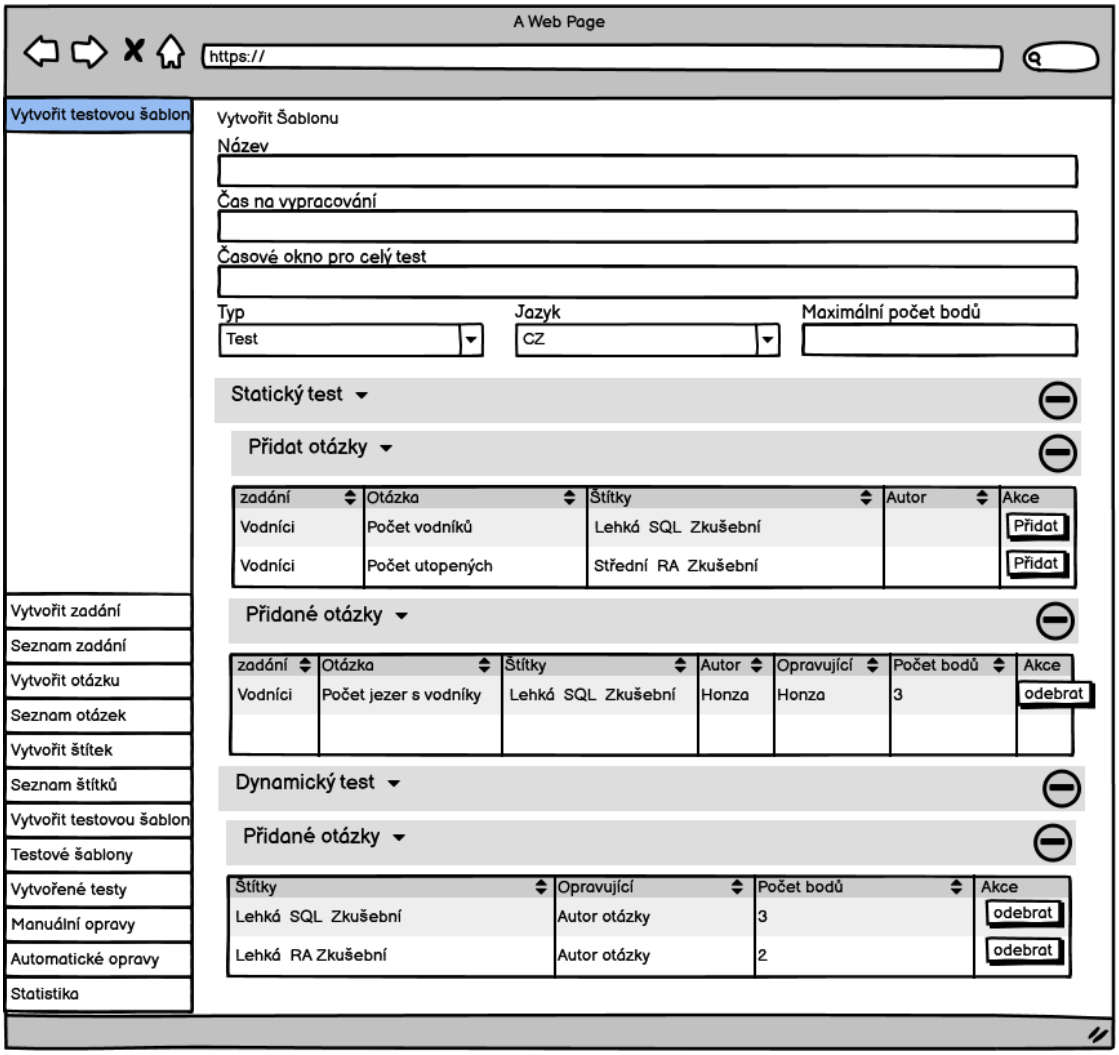


Obrázek 3.5 Výsledný návrh: Stránka pro správu štítkových skupin a štítků

Pro vytváření statického testu slouží 2 seznamy otázek. Všechny otázky v systému, které lze do šablony přidat a seznam již přidaných otázek, kde je potřeba nastavit opravujícího a počet bodů za danou otázku, je zde také možnost měnit pořadí otázek.

Pro vytvoření dynamického testu se používá automatické vybrání otázky pomocí zvolených štítků. Uživatel musí vybrat jeden nebo více štítků, které charakterizují danou otázku. Stejně jako u statického testu je zde potřeba vyplnit opravujícího a počet bodů za otázku, samozřejmě nechybí ani možnost změny pořadí otázek.

Výsledný návrh je na obrázku 3.6.



Obrázek 3.6 Výsledný návrh: tvorba testové šablony

3.3.5 Testové šablony

Tato stránka obsahuje seznam všech vytvořených testových šablon systému. Sloupce tabulky obsahují datum vytvoření, název, čas na vypracování, celkový čas, typ, počet bodů a autora. Uživatel zde má také možnost spustit statický nebo dynamický test z dané šablony, dále možnost duplikovat a odstranit šablonu.

Kapitola 4
Realizace

V této kapitole se budu věnovat postupu vývoje základního prototypu frontendu testového modulu na základě vytvořeného návrhu, který jsem popsal v předchozí kapitole.

Nejprve představím použité technologie, podívám se na instalaci a nastavení prostředí. Dále provedu potřebné aktualizace a integrace knihoven, které jsem se rozhodl použít. Upřesním postup integrace do portálu pro budoucí využití a popíši postup a metodiku vývoje, při které jsem měl tu šanci spolupracovat na vývoji s týmem z předmětu BI-SP1 jako vedoucí a dozorce.

4.1 Architektura systému

Vue.js je stejně jako současně používaný Framework Nette postavené na architektuře MVC. Architekturu MVC jsem již popisoval v kapitole *Aktuální struktura aplikace* 2.1.1. Pro komunikaci mezi současným řešením a novým testovým modulem se použijí Nette presentery pro vytvoření API. Nette se bude starat o data v databázi a Vue.js bude využito pro zobrazení uživatelského rozhraní.

4.2 Použité technologie

Klasická webová stránka se skládá z několika součástí. Základem je HTML, představující kostru stránky včetně obsahu, a CSS, starající se o vzhled a grafickou podobu. Často bývá přítomný JavaScript, ať už jen pro analytické nástroje, nebo pro dodání dynamiky uživatelského rozhraní. Moje řešení využívá framework Vue.js. Ten je psaný v jazyce JavaScript. V této sekci popisuji zmíněné technologie. Vue.js jsem zvolil ze dvou důvodů, a to jelikož se již využívá v některých částech portálu, které se předělávají, proto byl také vytvořen nefunkční požadavek N1 2.4.2.1 pro volbu Vue.js. Hlavním důvodem je analýza, kterou provedl Ing. Oldřich Malec ve své diplomové práci pro tvorbu frontendu skladového systému. Kde důkladně porovnal populární frontendové frameworky a knihovny. [1]

4.2.1 HTML a CSS

HTML je jazyk popisující strukturu webových stránek [13]. Umožňuje strukturovat dokument do nadpisů a odstavců, vkládat tabulky, obrázky a jiná multimédia přímo do dokumentu, vytvářet formuláře pro práci s daty a také vzájemně propojovat dokumenty pomocí hypertextových odkazů. Struktura stránky je popsána značkami, které představují jednotlivé elementy (např.

<table>, <img> nebo <button>). Objektově orientovaná reprezentace struktury dokumentu je označována jako rozhraní DOM.

Pro popis vzhledu stránky slouží jazyk CSS [13]. Jednotlivým elementům přiřazuje styly, které tvoří atributy jako barva, velikost, font a spousta dalších. Umí také základní animace při interakci uživatele (např. při najetí kurzorem myši na element). Dokáže navrhnout rozvržení stránky pro malá mobilní zařízení, velké desktopové obrazovky nebo tiskárny. Oddělení HTML od CSS umožňuje jednodušší správu stránek a také sdílení stylů mezi více stránkami.

4.2.2 Vue.js

Vue je progresivní framework používaný pro vytváření uživatelských rozhraní. Základní knihovna je zaměřena pouze na vrstvu zobrazení dat a je snadné ji použít samotnou, nebo dále integrovat s jinými knihovnami nebo projekty. Dále je Vue schopno také vytvářet jednostránkové aplikace při použití v kombinaci s moderními nástroji a knihovnami. [14]

4.3 Lokální instalace

Začal jsem instalací lokálního prostředí podle návodu na wiki systému DBS. Pro verzování systému se používá Git a pro vývojové prostředí se využívá virtualizace, konkrétně Vagrant v kombinaci s programem VirtualBox.

Git se používá ve většině softwarových firem, umožňuje sdílet kód mezi počítači a kooperovat v týmu více lidí. Kdykoliv se dá vrátit k původní stabilní verzi a vrátit nepovedené změny bez toho, aniž byste o svůj kód přišli. [15]

Vagrant je nástroj, který vytváří kompletní přenosné vývojové prostředí pomocí virtualizace. V podstatě vytvoří na našem stroji další virtuální počítač, který muže být konfigurován například stejně jako produkční linuxový server [16]. VirtualBox je software od společnosti Oracle, úkolem tohoto softwaru je vytvořit virtuální počítač v jiném počítači. Může sloužit například k vyzkoušení jiného operačního systému [17]. Nebo v mém případě k simulaci produkčního prostředí serveru.

Samotná instalace nebyla složitá, ale při stažení nového backendu jsem narazil na chybovou hlášku „Call to undefined function apcu_fetch(), did you mean oci_fetch()?.“ Problémem byla chybějící knihovna. Při její instalaci jsem však narazil na další problém.

Knihovnu jsem instaloval pomocí APT. APT je balíčkový systém používaný v Linuxu a jeho derivátech a usnadňuje správu softwaru [18]. Uložené archivy odkud se knihovny stahují, byly zastaralé a jednoduché apt-get update také nepomohlo. Nakonec jsem se po hledání na internetu [19] a radě z DBS chatu dopracoval k výsledným sedmi příkazům, které aktualizovali archivy a knihovnu nainstalovali.

Výpis kódu 4.1 Použité příkazy

```
sudo apt-get clean
sudo mv /var/lib/apt/lists /var/lib/apt/lists.old
sudo mkdir -p /var/lib/apt/lists/partial
sudo apt-get clean
sudo apt-get update
apt install php7.2-apcu
systemctl restart php7.2-fpm
```

V poslední řadě bylo potřeba vygenerovat nové tabulky do databáze pomocí jednoduchého příkazu *orm:schema-tool:update*. To však také nefungovalo, jak by mělo, příčinou byly některé sloupce typu *datum*, které byly v kódu definovány jako typ „timestamp,“ ten však nefungoval a tak jsem jej prozatím nahradil typem *datetime,* jelikož se samotnou databází pracovat

nebudu. Po přepsání tohoto typu v pěti souborech, konkrétně *studentTest.php, course.php, generaltest.php, user.php a exam.php,* již příkaz *orm:schema-tool:update* proběhl tak jak měl a nové tabulky vygeneroval.

4.3.1 Vývojové prostředí

Pro implementaci prototypu jsem používal program PhpStorm. PhpStorm je chytrý editor kódu, který poskytuje vynikající podporu pro PHP (včetně nejnovějších verzí a rámců jazyka), HTML, JavaScript, CSS a mnoho jiných jazyků. Pomáhá s kódováním za pomoci kontextového doplňování kódu, detekcí chyb, kontrolou a opravami kódu za chodu. [20]

Dalším, velmi užitečným nástrojem je Webpack. Je to software, který zpracovává části webových aplikací a tvoří z nich balíčky vhodné pro webové prohlížeče. Primárně je zaměřen na JavaScript, ale zvládne zpracovat i mnoho dalších formátů. Hlavním důvodem využití je možnost rozdělovat kód do více souborů, což je při programování žádoucí. Výstupem jsou poté jenom čtyři soubory, které zpracuje prohlížeč. [21]

Na portálu DBS již byl webpack zaveden a nakonfigurován. Moje práce s webpackem spočívala pouze v přidání jednotlivých záznamů o stránkách, které jsem do aplikace přidal a jeho spouštění pro aktualizaci generovaných souborů.

4.3.2 Aktualizace a knihovny

4.3.2.1 Vue 3

Po instalaci je čas aktualizovat knihovny na nejnovější podporované verze. První aktualizací byla aktualizace samotného Vue.js z verze 2 na verzi 3. Návod pro migraci lze najít na oficiálních stránkách Vue.js [22]. V době aktualizace byly ještě některé části kódu zpětně nekompatibilní, v rámci DBS portálu se však jednalo jen o jednu knihovnu, kterou při implementaci používat nebudu. Knihovnu jsem tedy prozatím odstranil a při integraci prototypu na produkční úroveň předpokládám, že už daná knihovna bude aktualizovaná. Další nekompatibilní kód bylo potřeba změnit v již vytvořených zhruba dvanácti souborech podle návodu migrace. Převážně se jednalo o malou změnu syntaxe. Všechny potřebné provedené změny lze najít na oficiálních stránkách Vue jak už jsem zmiňoval.

Výpis kódu 4.2 Příklady změn

```
import Vue from 'vue'; -> import { createApp } from 'vue';

$root.$on -> $emit

Vue.use(l10nPlugin);
new Vue({
    store,
    render: h => h(App)
}).$mount('#app');
        ->
createApp(App)
    .use(store)
    .use(l10nPlugin)
    .mount('#app')
```

4.3.2.2 Multiselect

Dalším krokem bylo vybrat knihovnu, která se bude používat zejména pro štítky. Jako vhodná knihovna se ukázala knihovna multiselect [23]. Samotná instalace není nic složitého, vše je po-

psané na zmíněné stránce knihovny. Důvodem výběru byla přímo funkce označování, což je přesně to co od štítků požadujeme. Umožnuje také rozdělování do skupin a jednoduše lze nastavit i potřebná disjunktnost skupin.

Následující kód je ukázka jak lze docílit disjunktivních skupin. Potřebné nastavení komponenty multiselect, vstupní data a funkce pro schování použitých skupin. Způsobů jak lze tuto funkcionalitu naimplementovat bude více, zde uvádím způsob, kterým jsem to udělal ve své práci.

Výpis kódu 4.3 Nastavení komponenty multiselect

```
<multiselect id="tags" v-model="question.tags" label="name" track-by="id"
   :hide-selected="true" :show-labels="false" :placeholder="Add tag"
   group-label="name" group-values="tags" :group-select="false"
   :options="availableGroups" :multiple="true" :disabled="!isEditable">
```

Výpis kódu 4.4 Funkce pro schování použitých skupin

```
availableGroups() {
 return this.tagGroups.filter(group => group.disjunctive === true &&
 this.question.tags.some(tag => group.tags.map(t => t.id).includes(tag.id)))
;}
```

Výpis kódu 4.5 Vstupní data s disjunktivními skupinami

```
tagGroups: [
  { id: 0,
    name: 'žObtínost',
    disjunctive: true,
    tags: [
        { id: 0, name: 'Lehká' },
        { id: 5, name: 'ěžTká' } ] },
  { id: 1,
    name: 'Kategorie',
    disjunctive: true,
    tags: [
        { id: 2, name: 'SQL' }, { id: 3, name: 'RA' },
        { id: 6, name: 'Teorie' } ] } ]
```

4.4 Implementace

Úkolem práce bylo naimplementovat prototyp části frontendu testového modulu pro portál DBS. Implementace spočívala ve vytvoření jednotlivých částí podle funkčních požadavků 2.4.1 a návrhů 3.3. V aplikaci již byla část implementovaná ve Vue, což nepatrně usnadnilo práci a po aktualizaci knihoven tedy stačilo jenom přidat část pro testový modul.

Po řádně zpracovaném návrhu, jsem měl na výběr, jestli prototyp budu implementovat sám, nebo si vyzkouším roli manažera vývojového týmu, který si v rámci BI-SP1 vybral, že by chtěli pracovat na vývoji frontend části nového testového modulu. Vybral jsem si roli manažera při spolupráci s týmem z BI-SP1.

V této kapitole se podrobně věnuji postupu, kterým se prototyp vyvíjel.

4.4.1 Spolupráce s týmem BI-SP1

V rámci předmětu BI-SP1 si studenti zvolí projekt, na kterém se chtějí podílet a pokud je studentů na dané téma dostatek utvoří tým a podílejí se na vývoji na daném projektu. Hlavní

částí předmětu je analýza v podobě modelování různých diagramů, sepsání požadavků a případů užití, analýza domény a následný návrh. Další částí je potom implementace a testování.

Tento rok se na realizaci DBS portálu přihlásilo devatenáct studentů, kteří byli nakonec rozděleni do třech týmů. Se mnou spolupracoval tým 1, v kterém bylo 7 studentů. Analýzu si tým provedl samostatně pro potřebu předmětu a spojili se se mnou až po jejím dokončení. Na první schůzce jsem tým seznámil s vytvořenými návrhy stránek a spolupráce tedy spočívala převážně v implementaci.

4.4.1.1 Použité nástroje

Pro komunikaci jsme používali Discord [24]. Jedná se o webovou, desktopovou i mobilní aplikaci, která umožňuje vytvořit si svůj prostor, který je dostupný jen pro pozvané. V tomto prostoru jdou pak vytvořit různé textové a hlasové kanály. Discord jsme používali především pro schůzky, jelikož jsme se nemohli scházet osobně z důvodu pandemické situace. Dále jsme komunikovali v rámci různých kanálů například za účelem pomoci s problémem nebo sdílení informací a souborů.

Dále jsme používali aplikaci Redmine [25], je to aplikace pro řízení projektu, která se aktuálně používá pro vývoj na DBS portálu. Redmine se při vývoji DBS používá pro sepisování úkolů, sledování odvedené práce na daném úkolu, spolu se stráveným časem při jeho řešení a v neposlední řadě je v aplikaci i Wiki – seznam návodů a dokumentací.

4.4.2 Metodika vývoje

Vývoj probíhal standartním způsobem, který je zavedený pro DBS projekt. Jedná se o agilní metodiku na pomezí Extrémního programování a Feature driven development. Z Extrémního programování je převzatá zhruba polovina základních postupů, konkrétně se jedná o neustálé revize kódu, krátké iterace, malé verze, karty zadání, testování, nepřetržitá integrace a zákazník na pracovišti. Z Feature driven development je přebrána role hlavního programátora a rozdělení na jednotlivé vlastnosti. Pokud se chcete dozvědět více o tom, proč se zavedl tento způsob vývoje pro DBS projekt, doporučuji se podívat na bakalářskou práci Oldřicha Malce [1]. V této sekci zkráceně popíši tento konkrétní postup, jak probíhá v reálném prostředí.

V aplikaci Redmine jsou nejprve vytvořeny úkoly, které obsahují popis, prioritu, stav, strávený čas atd. Úkol může vytvořit kdokoliv z vývojářů v systému, například v rámci týmů vytváří vedoucí úkoly pro jednotlivé členy. Úkoly jsou následně přiřazeny na studenty, nebo si student vybere úkol sám. Studenti na úkolu pracují a každý týden probíhá schůzka, na které vedoucí týmů, nebo samotný student co na úkolu pracoval, sdělí jejich postup za celý týden projektovému manažerovi, kterým je Oldřich Malec. Na schůzce je vždy také přítomen Jiří Hunka, který z pohledu Extrémního programování zastává roli zákazníka. Je jedním z hlavních uživatelů portálu DBS, určuje tedy priority a nové požadavky, případně také hlásí chyby a testuje nové verze.

4.4.2.1 Extrémní programování

XP extremizuje dosavadní principy. Způsob extremizace spočívá v tom, zda se něco osvědčí, bude se to používat neustále. Pracuje se na co nejjednodušší verzi, která splňuje požadovanou funkcionalitu a funguje korektně. Pracuje se ve velmi krátkých iteracích, které tak dodávají i ty nejmenší funkcionality. Integruje se ihned po řádném otestování. [26]

4.4.2.2 Feature driven developement

Metodika FDD je založena na iterativním vývoji a soustředí se především na vlastnosti produktu. Začíná se vytvořením základního modelu software, který se zaměřuje na jednotlivé vlastnosti. Zpravidla se potom pokračuje vývojem v dvoutýdenních iteracích, jejichž výstupem je fungující software, který lze představit zákazníkovi. [27]

4.4.3 Postup vývoje

V této kapitole popisuji způsob a zejména průběh vývoje prototypu, na kterém jsem pracoval spolu s týmem 1 v roli manažera. Celkově jsme měli 20 schůzek v rámci předmětu BI-SP1, které jsou zaznamenané na Wiki v aplikaci Redmine. Schůzky byly také nahrávané, avšak všichni členové týmu preferovali, aby nebyly schůzky veřejně dostupné, ale umožnují zobrazení nahrávek na vyžádání. Následoval i vývoj v rámci předmětu BI-SP2, kde už jsem se neúčastnil tak aktivně jako v rámci BI-SP1.

Jednotlivé schůzky probíhaly téměř každé úterý a pátek. Úterní schůzky byly interní v rámci týmu 1, na kterých jsem kontroloval výstupy, které studenti zvládli připravit za celý týden, zodpověděl případné otázky, které si pro mě připravili, nebo se pokusil poradit na problém, na který narazili. Následně jsem sloučil kódy do jedné větve a vytvořil nové úkoly na další týden, které si studenti mezi sebou rozdělili podle preferencí. Schůzky v pátek byly schůzky všech týmů, spolu s Oldřichem Malcem a Jiřím Hunkou. Obsah této schůzky byl podobný jako schůzka v rámci týmu 1, jednalo se akorát o schůzku v poměrně větším obsazení.

Mimo schůzky jsem se věnoval problémům, které se v týmu objevili a snažil se co nejvíce pomoci je vyřešit. Ne vždy se tak podařilo, jelikož jsem s Vue také ještě nikdy před prací na tomto projektu nepracoval. Když nebyly žádné aktuální problémy tak jsem pomáhal s implementací většinou studentovi, který se na něčem zasekl.

4.4.3.1 Průběh

Na první schůzce jsem studenty seznámil s vytvořenými návrhy, které si studenti důkladně prošli a rozmysleli si, kdo bude pracovat na které části. V tomto týdnu se ale studenti věnovali převážně studování práce ve Vue.js, jelikož s ním ještě nepracovali. Já jsem mezitím připravil soubory pro jednotlivé stránky a základní strukturu pro nový testový modul. Další týden si studenti rozebrali jednotlivé části, na kterých následně pracovali, úkolem tohoto týdne bylo vytvoření především kostry stránek.

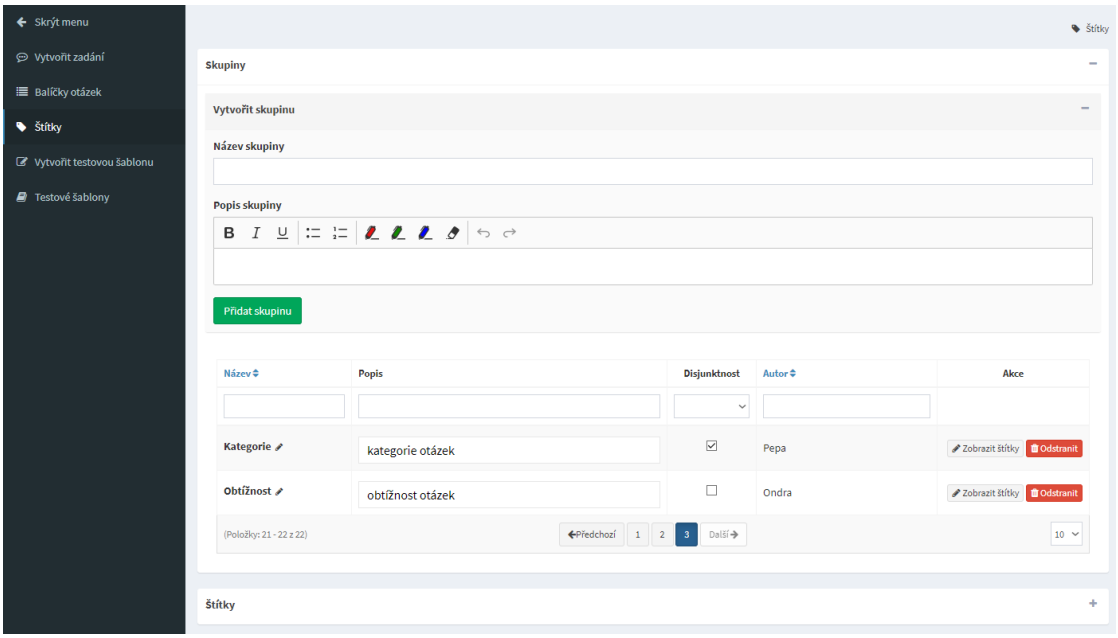
V následujících týdnech se pokračovalo především v přidávání prvků z Vue, které přidali funkčnost ještě nefunkčním prvků, které byly připraveny v rámci kostry. Narazilo se také na problémy, které se řešili několik týdnů, jednalo se hlavně o spárování kreslítka, tralexu a speciálních editorů s Vue komponentami, aby se dalo přistupovat k jejich datům. Nad tímto problémem jsme konzultovali i na společných schůzkách a nakonec se podařil vyřešit.

Tralex je speciální editor s tlačítky pro vkládání znaků potřebných pro transformace (podmnožina, sjednocení atd.). Kreslítko je speciální komponenta pro práci s diagramy (vytváření, import, uložení atd.). U zmíněných editorů se jedná o editor s vlastním formátováním a editor pro psaní dotazů v SQL a RA, spolu s připojením k databázi (je zde vidět struktura databáze – tabulky a jejich atributy, které se dají jednoduše kliknutím vložit do editoru).

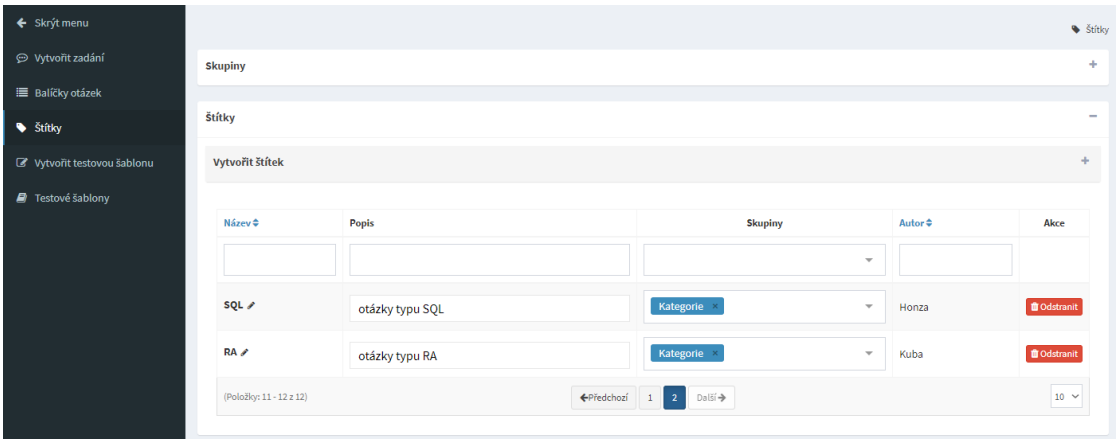
4.4.3.2 Prototyp po BI-SP1

V této sekci jde vidět výstup práce v rámci spolupráce s týmem z BI-SP1. Jedná se o funkční prototyp nového testového modulu, bez práce s API, které ještě neexistovalo. Data jsou vnitřně namockovaná a data mezi stránkami nejsou propojená. Z tohoto důvodů by uživatelské testování nebylo pro uživatele pohodlné. Pro účely uživatelského testování jsem si tedy mock připravil sám, průběh popisuji v další podkapitole 4.5.

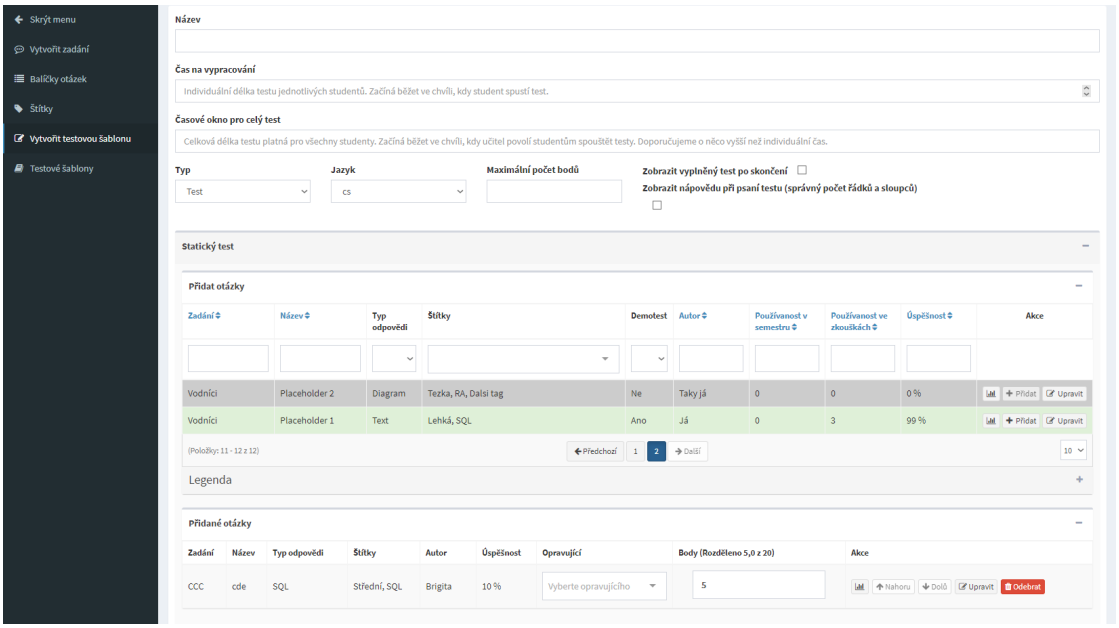
Design a styly jednotlivých komponent byly zvoleny tak, aby byli konzistentní se zbytkem systému. Výsledné stránky prototypu jsou vidět na následujících obrázcích (4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7).



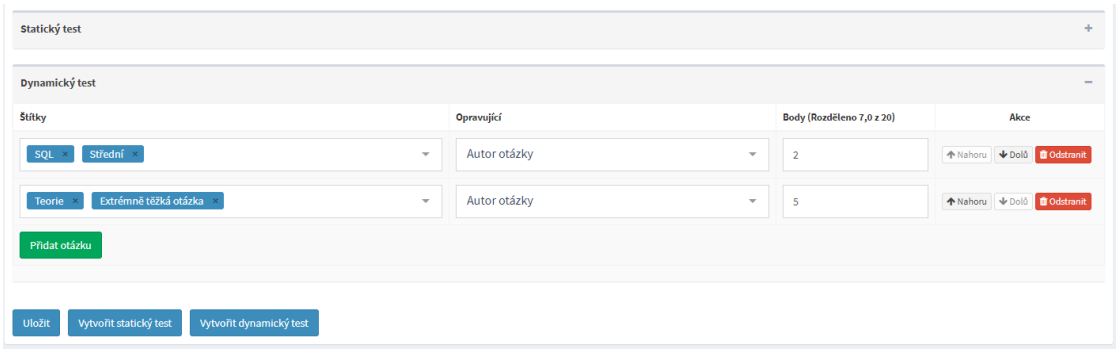
Obrázek 4.1 Výsledný prototyp: komponenta pro správu štítkových skupin



Obrázek 4.2 Výsledný prototyp: komponenta pro správu štítků

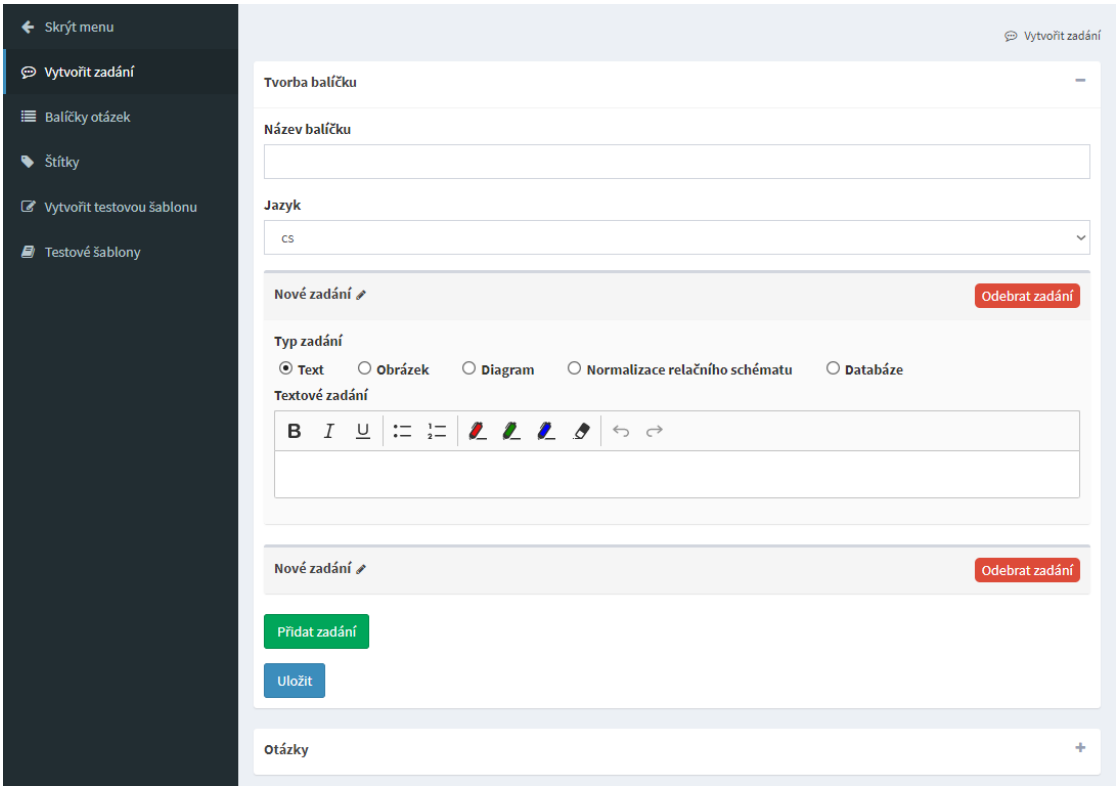


Obrázek 4.3 Výsledný prototyp: tvorba testové šablony, část statický test



Obrázek 4.4 Výsledný prototyp: tvorba testové šablony, část dynamický test

Obrázek 4.5 Výsledný prototyp: seznam testových šablon



Obrázek 4.6 Výsledný prototyp: komponenta pro vytváření balíčku zadání

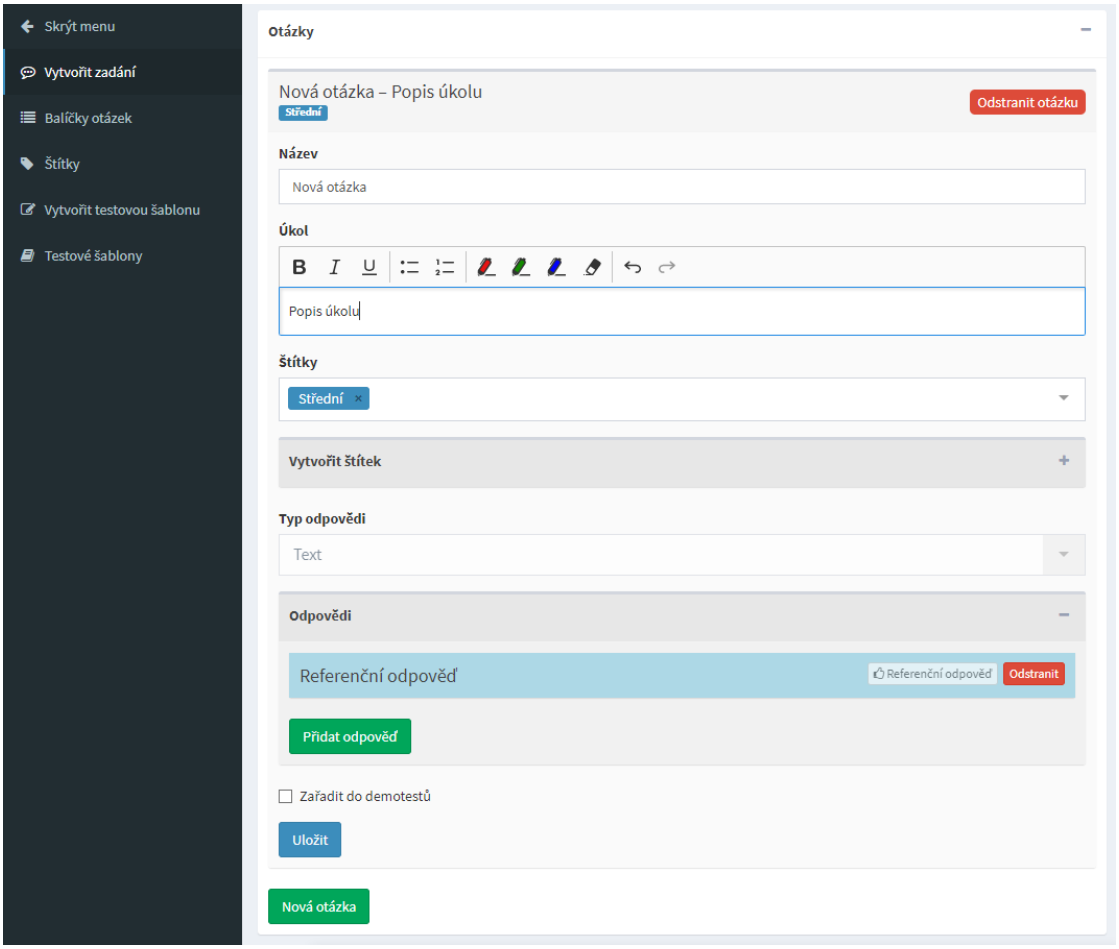
4.5 Mock pro testování

V rámci mé práce na novém testovém modulu byly zkušební data napsaná „natvrdo“ do kódu a neumožňovali zachování dat mezi stránkami. Uživatelské testování by tak bylo velice nepříjemné pro uživatele a proto jsem vytvořil mock, který ukládání mezi stránkami umožnil. Pro jeho realizaci jsem využil knihovnu Vuex [28] a vuex-persist [29]. Mock jsem vytvořil v nové větvi „add-vuex-mock“ na gitu a na té jsem potom prováděl uživatelské testování. Mock jsem vytvořil tímto způsobem, jelikož jsem v době před testováním věděl, že se zase mění backend.

4.6 Integrace do systému

Jedním z bodů zadání práce je integrace odladěného prototypu do systému. Plně funkční integrace do systému není možná, důvodem je neexistující API a opětovné předělávání backend části modulu. Nové uživatelské rozhraní je však funkční a může být využito pro další vývoj. Uživatelské rozhraní jsem otestoval, jak popisuji v kapitole 5, a ověřil tím jeho funkčnost a použitelnost. Problémy, na které jsem při testování byl upozorněn a poznámky, které mám k této práci z průběhu vývoje, jsou zavedené v aplikaci Redmine, což umožní jednoduchý následující vývoj.

Implementace celého prototypu nového modulu je vytvořená na samostatné stránce. Stránka se zbytkem systému bude komunikovat pouze pomocí API, které zatím neexistuje. Pro nasazení až bude funkční backend část modulu bude tedy potřeba sjednotit posílaná data mezi backendem a frontendem a vytvořit API.



Obrázek 4.7 Výsledný prototyp: komponenta pro vytváření otázek

Implementace probíhala přímo v repositáři projektu v samostatné větvi, kde je veškerý kód, který jsem v rámci bakalářské práce vytvořil nebo kontroloval. Kód se nachází na gitlabu projektu newDBS ve větvi bw_newtest_frontend.

V současné době už proběhl další vývoj na testovém modulu v rámci předmětů BI-SP1,2. V novém prototypu se připravují potřebné metody na spojení s API. Jsou zde i změny uživatelského rozhraní způsobené změnou backendu. U částí, které jsem měl součástí práce je například po změně backendu upravena tvorba testové šablony, kde se odstranilo rozdělení na statický a dynamický test, místo toho lze vytvořit test který má otázky namíchané jak ze statických otázek tak i z otázek generovaných pomocí štítků. Nově jdou dále štítky přidávat i k testovým šablonám například pro rozlišení zda se jedná o demo-test, test v semestru nebo zkoušku. Dále byli přidané ostatní části testového modulu jako je oprava a spouštění testů. Aktuální větev, na které se vyvýjí je SP-DBS-2022-frontend.

Aktuálně Bc. Andrii Plyskach pracuje na diplomové práci, ve které převádí celý systém na architekturu mikroslužeb. Po převedení systému na mikroslužby bude práce s daty a následná komunikace s Vue ještě jednodušší. Mikroslužby komunikují pomocí technologicky nezávislých protokolů, jsou na sobě nezávislé, mohou být implementovány pomocí různých programovacích jazyků a jsou malé.

Aktuální stav části, kterou se zabývala tato práce je stále ve fázi prototypu, kdy je ještě potřeba například dopsat persistence dat, ale většina funkcí, které budou potřeba pro komunikaci s API jsou už připravené a uživatelské rozhraní na jednotlivých stránkách je funkční. Kód pro komunikaci s API bude samozřejmě potřeba refaktorovat, ale zbývající kód je reálně použitelný. Dále je potřeba opravit chyby, které byli nalezeny při testování. V tomto semestru byl vývoj v rámci předmětu BI-SP1 zaměřený spíše na ostatní části testového modulu, na části pro vytváření testů se tedy nepracovalo. Do produkční verze bude řádně tato práce integrována, až poté co bude nový backend založený na mikroslužbách funkční.

4.7 Práce v BI-SP2

V rámci dalšího semestru tým 1 dále pokračoval ve vyvíjení nového testového modulu, kde dokončili věci, které nestihli v rámci BI-SP1 a následně připravovali nový prototyp pro připojení k API a jeho specifikaci. Já jsem se tohoto dalšího vývoje zúčastnil jenom okrajově, nemohu tedy napsat, že jimi provedené úpravy jsou součástí mojí práce, ale tuto informaci zde uvádím pro případné čtenáře, kteří budou dále navazovat na tuto práci a práci týmu 1. Práce, kterou jsem kontroloval a společně s týmem implementoval je na Gitlabu uložena ve větvi bw_newtest_frontend. Aktuální stav nového testového modulu je možné nalézt na Wiki v aplikaci Redmine u projektu „SP2-DBS-2021-team-I-tvorba testů učitel FE.“

Kapitola 5
Testování

Jedním z bodů zadání práce je otestování vytvořeného prototypu. Jedná se o proces kontroly kvality softwaru. Jeho podstata spočívá v ověření, že požadované funkce nastanou a nechtěné nenastanou. [30]

V případě této práce jde především o otestování použitelnosti uživatelského rozhraní. Použitelnost vyjadřuje zejména snadnost použití a efektivnost uživatele. Více o použitelnosti uživatelského rozhraní jsem popsal v kapitole 2.5. Výsledný prototyp používá v řadě věcí stejné komponenty jako současné řešení. Byly přidány nebo změněny některé funkce (štítky, statický a dynamický test atd.), u kterých je potřeba se přesvědčit, že půjdou intuitivně využít koncovými uživateli.

Z těchto důvodů proběhne uživatelský test. V této kapitole nejprve popíši teorii uživatelského testování. Dále zde bude popsaná příprava, průběh a zhodnocení testování prototypu frontendu pro vytváření testů.

5.1 Uživatelské testování

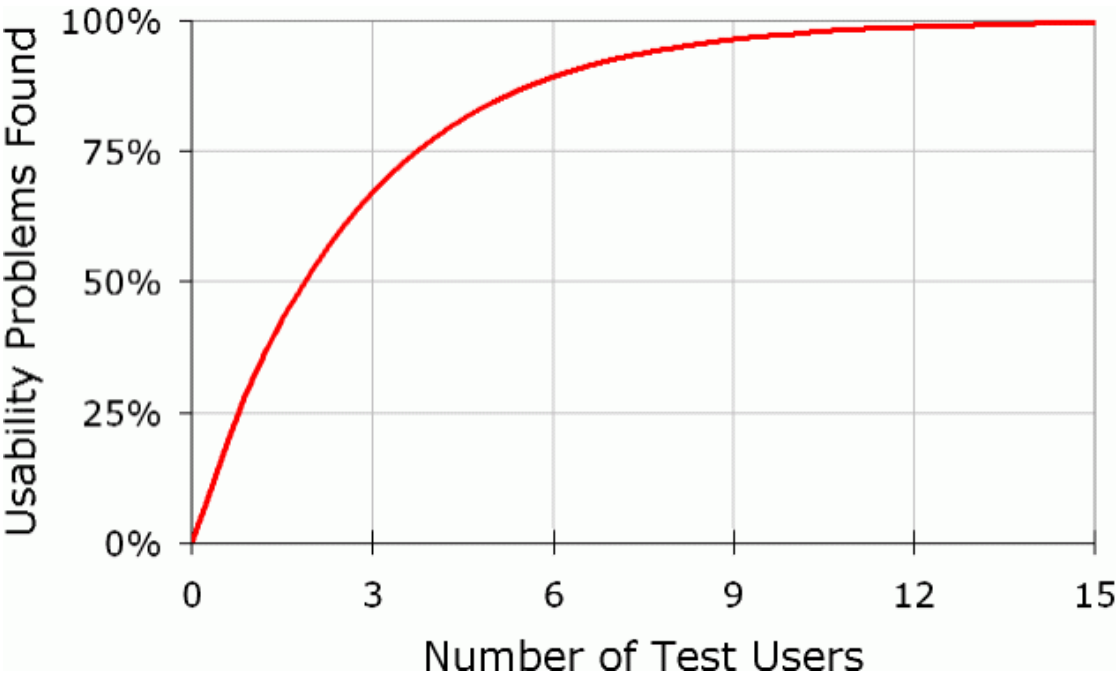
Uživatelské testování je jednou z nejčastějších metod testování použitelnosti aplikace. Tato metoda se zaměřuje na samotné chování potenciálních uživatelů, čímž lze jednoduše odhalit chyby co by vývojáře, které svůj vyvíjený systém znají, nenapadli. Uživatelské testování se dá aplikovat jak při fázi návrhu například nad jednotlivými papírovými modely, tak i v rámci samotného vývoje, při kterém může odhalit chyby zavčas a sníží tím počet problémů, které by se dali vytvořit při další práci s chybnou částí kódu. Nejčastější nalezené problémy se týkají například nestandartního chování prvků, špatně nazvaných prvků nebo špatně dohledatelné informace. [31]

5.1.1 Respondenti

K uživatelskému testování nesmí chybět samotní respondenti – potenciální uživatelé naší aplikace. Vybraná skupina uživatelů by měla být rozmanitá – různý věk, pohlaví, technická úroveň atd. V případě této práce se jedná o testování vytváření testů. Testy vytvářejí učitelé předmětu DBS, rozmanitá skupina tedy moc nebude, ale pro účely testování jsem si vybral dva učitele, kteří se současným vytvářením testů jsou velice dobře seznámeni a dva, kteří si současné vytváření testů už moc nepamatují.

5.1.2 Počet účastníků

Dle Jakoba Nielsena [32] odhalíme 60-80% chyb při testování s 3-5 uživateli. Mohlo by se zdát, že čím více testů provedeme, tím více problémů odhalíme, ve skutečnosti se nalezené problémy budou duplikovat. Některé problémy zjištěné při prvním testování se budou překrývat s těmi, které nalezne uživatel v druhém testování atd. Graf vztahu množství nalezených chyb a počtu respondentů je na obrázku 5.1. Pro uživatelské testování vytváření testů jsem si zvolil skupinu o 4 respondentech, kteří podle zmíněného grafu odhalí zhruba 75% problémů.



Obrázek 5.1 Procento nalezených chyb v závislosti na počtu respondentů. [32]

5.1.3 Scénáře a úkoly

Před začátkem testování je potřeba připravit si scénář testování, který obsahuje popis nějaké realistické situace s cílem a úkoly pro respondenty, které vedou k naplnění cíle a ověření použitelnosti testované části aplikace daným scénářem. Vytváření úkolů by se mělo držet několika pravidel.

- Úkoly nesmí být rozděleny na jednotlivé části (například úkol: nakupte černé triko nerozdělovat na: vyberte produkt, vložte do košíku, otevřete košík, atd.)
- Úkoly nesmí navádět jak samotný úkol vyřešit. [33]

V rámci testování vytváření testů v novém modulu jsem si připravil 2 scénáře testování. Scénář 1 se týká vytvoření testové šablony, ze které by následně učitel spouštěl test pro své studenty na cvičení. Jednotlivé úkoly jsou zaměřeny na tvorbu otázek pro tento test. Všechny potřebné materiály pro přípravu otázek jsou pro respondenta připravené v rámci jednotlivých úkolů. Jedná se o zadání, samotný úkol otázky a odpověď, ale za účelem zachování druhého pravidla, o kterém jsem psal výše, nejsou jednotlivé části popsány těmito názvy. Uživatel si tedy musí rozmyslet, jakým způsobem připravené otázky vloží do systému DBS.

Scénář 2 se týká otestování nově přidané možnosti vytvářet dynamický test pomocí štítků. Úkolem scénáře je vytvořit testovou šablonu pro demo test, který obsahuje jednu náhodnou zkušební otázku každého typu.

Celé scénáře, spolu se všemi úkoly a připravenými daty pro otázky naleznete v příloze B.

5.2 Průběh

Před začátkem samotného testování je dobré uživatele seznámit s obecným průběhem testování. Ujistit je, že pokud se zaseknou u některého úkolu, není to jejich chyba, ale jedná se o chybu v návrhu uživatelského rozhraní. Domluvit se na způsobu záznamu z testu (video nebo jen audio), zda souhlasí s případným zveřejněním záznamu (například do bakalářské práce). Dále je dobré požádat uživatele, aby „přemýšleli nahlas“ – vyjadřoval své myšlenky při řešení úkolů nahlas a v neposlední řadě, že by se neměl s testerem radit v průběhu testu. [34]

Uživatelské testování jsem prováděl s uživateli online formou, za pomocí aplikace AnyDesk (aplikace pro přístup a získání ovládání k vzdáleným zařízením [35]) a pro komunikaci a vytvoření záznamu z testování byla použita aplikace Microsoft Teams [36]. Před začátkem jsem uživatele seznámil s průběhem testování a poslal jim scénáře. Při plnění úkolů jsem získával zpětnou vazbu z jejich „hlasitého myšlení“ a po dokončení scénářů mi sdělili celkový dojem a případné další nedostatky nebo návrh na které si vzpomněli po dokončení testování. Zároveň si prošli veškeré stránky ještě jednou, aby si jednodušeji vzpomenuli, jestli se jim něco líbílo nebo nelíbilo.

Záznamy z testování s respondenty naleznete na přiloženém médiu. Z jednotlivých testů v následujících podkapitolách krátce popisuji nejdůležitější poznatky z testování. Celkový zápis nalezených problémů a případných návrhů na zlepšení jsem zpracoval do seznamu, který je dostupný v příloze C.

5.2.1 První testování

Při prvním pokusu o testování, které proběhlo 21. 6. 2021 s Ing. Jiřím Hunkou jsem narazil na řadu nepříjemností, které uživatelské testování znemožnily (nebyl jsem dostatečně připraven, jednalo se zároveň o jiný scénář, než ten co jsem tu popisoval, ale uvádět ho zde nebudu, jelikož se nevyužil). Jednalo se o neukládání dat mezi stránkami, všechny stránky se každých 5 minut obnovovali a tím způsobili ztrátu dat i na aktuální stránce. Špatně se načítaly překládané texty, někdy se načetly a někdy ne. Nakonec si při tomto testování uživatel byl schopen jenom prohlédnout jednotlivé prvky uživatelského rozhraní, které z výše uvedených důvodů nebylo možné důkladně otestovat. Důkladné uživatelské testování jsem tedy odložil, do té doby, než tyto problémy vyřeším a dostatečně si testování připravím a vyzkouším sám.

5.2.2 Respondent 1

Cvičící DBS, s praxí v portálu, ale test vytvářel před několika lety

S respondentem 1 se jednalo o první test, na který jsem byl řádně připraven. Pro testování se uživatel musel prvně seznámit se stránkou v menu uvedenou jako „Vytvořit zadání,“ na které se vytváří zadání, otázky i jejich odpovědi. Po otevření stránky uživatele rozhodilo pojmenování boxu „Vytvořit balíček.“ Myslel si, že bude vytvářet jeden balíček, do kterého přidá zadání všech otázek. Poté co se dostal k vytváření otázky nevěděl jak vybrat zadání, které k otázce chce přidat.

Vysvětlil jsem tedy terminologii co je to balíček zadání. Po vysvětlení už splnění prvního úkolu proběhlo bez problému. Splnění zbývajících problémů pak proběhlo bez větších zádrhelů. Při posledním úkolu (vytváření testové šablony) uživateli chvíli trvalo najít, kde se zobrazují přidané otázky k šabloně. Otázky se přidávájí do seznamu, který je v boxu s názvem „Přidané otázky“ box byl však sbalený a seznam tedy nebyl vidět. Uživatel navrhl mít boxy při otevření stránky na vytváření testových šablon rovnou rozbalené. S tímto návrhem jsem ihned souhlasil a pro následující uživatelské testy rovnou změnil v implementaci u všech boxů, u kterých to dávalo smysl. Zároveň jsem změnil terminologii balíčku zadání, přejmenoval jsem balíček na „zadání“ a dílčí zadání balíčku na „části zadání.“

5.2.3 Respondent 2

Cvičící DBS, s velkou praxí v portálu, jeden z hlavních uživatelů

Druhý uživatel již okrajově nový prototyp vytváření testů viděl. Seznámení se stránkou pro vytváření zadání, otázek a odpovědí tedy proběhlo velmi rychle a úkoly následně řešil bez větších problémů. Problémem, který se objevil v rámci testování při plnění více než jednoho úkolu, byla tlačítka pro přidání zadání, otázky a odpovědi. Jedná se o zelená tlačítka, která přidávají další zadání, otázku nebo odpověď. Uživatel si tlačítka několikrát spletl za tlačítka pro uložení. Po provedení tohoto testu jsem do tlačítek přidal slovo „další,“ tedy tlačítka jsou teď označeny slovy přidat další zadání, otázku a odpověď. V menu byly také po návrhu od uživatele přejmenovány 2 položky, konkrétně *Seznam zadání* na *Seznam zadání s otázkami* a *Vytvořit zadání* na *Vytvořit zadání a otázky*.

5.2.4 Respondent 3

Student a zároveň cvičící DBS, bez praxe v portálu, podílí se na jeho vývoji

Respondent 3 byl trochu zmatený připravenými daty, jelikož na portálu DBS už delší dobu test nevytvářel. Je možné, že jenom četl ve scénáři moc rychle a než si rozmyslel, co je zadání, co otázka a co odpověď už kopíroval do aplikace. Následně si přečetl, co vložil a musel vložená data přesunout na jiné místo. S novým rozhraním se ale seznámil rychle. Chyběli mu tlačítka pro uložení jednotlivých částí při vytváření částí zadání, otázek a odpovědí. Tlačítko uložit na stránce *vytvořit zadání a otázky* je jen jedno a ukládá celé zadání se vším, k němu přidaném. Jelikož se o tomto problému nezmínili předchozí respondenti, rozhodl jsem se nejprve provést poslední testování a rozmyslet si tlačítka pro ukládání až po něm.

5.2.5 Respondent 4

Cvičící DBS, s velkou praxí v portálu, podílí se na jeho vývoji

Čtvrtý uživatel bral testování velice důkladně a do detailů. S první stránkou se seznámil rychle. Testování proběhlo bez velkého problému, ale uživatel našel nejvíce nedostatků ze všech předchozích testů, dá se říci, že našel většinu nedostatků, které se našli ve všech předchozích testech a ještě některé navíc. Opět tomuto respondentovi chyběli tlačítka pro ukládání jednotlivých komponent. Jejich zavedení tedy určitě přiřadím nějakou prioritu pro další úpravy. Celkem větší nalezená chyba se týká znovupoužitelnosti obecných zadání, na kterou jsem byl při tomto testování upozorněn. Jde o případ, kdy popis s obecnými pokyny, například jak správně formátovat odpověď na transformaci, aby se otázka opravila automatem. Ve vytvořeném prototypu by se tyto pokyny museli kopírovat do každého zadání s transformací. Tuto chybu tedy bude potřeba opravit s vysokou prioritou.

5.3 Vyhodnocení

Uživatelské testování prototypu nového testového modulu, části pro tvorbu testů jsem provedl se skupinou čtyř respondentů. Respondent 1 potřeboval vysvětlit co to je balíček, než se podařilo splnit všechny problémy. Pro další testy jsem tento problém rovnou vyřešil a následující testy proběhly bez větších problémů. Velká většina nalezených nedostatků nebude na opravu nebo úpravu těžká a půjde opravit rychle. Nejtěžší problém k vyřešení bude problém zadání s obecnými pokyny, který našel respondent 4. Způsob, kterým bych problém řešil by spočíval v přidání nového přepínacího tlačítka do komponenty části zadání, které by po přepnutí umožnilo spojit zadání s částí jiného zadání, nebo s celým zadáním. Toto zadání, nebo část, by obsahovala obecný popis, který bude v mnoha zadáních.

Nalezené nedostatky jsem neopravoval z důvodu nedostatku času a navíc v době testování již proběhl na testové části vývoj, kterého jsem nebyl součástí, a mohli být některé věci změněny. Veškeré nalezené nedostatky jsem zpracoval do seznamu, který je dostupný v příloze C. Dále jsem tento seznam, spolu s návrhem řešení výše zmíněného problému zavedl také do aplikace Redmine pro poučení, v případě že by se uživatelské rozhraní předělávalo, ale zejména za účelem opravy chyb, které na aktuálně vyvíjené verzi zůstali stejné.

Kapitola 6

Závěr

Mezi hlavní cíle této práce patřila analýza a návrh frontendu nového testového modulu v DBS portálu. Tento návrh měl za úkol zavést novou funkcionalitu a následný důraz na jednoduchost použití. Na základě tohoto návrhu byl dále implementován prototyp frontendu a realizováno jeho základní testování. Tyto cíle byly splněny.

Nejprve byla provedena podrobná analýza současného systému pomocí případů užití, pro zjištění jeho hlavních funkcí a analýza nového backendu a jeho nových funkcionalit. Výsledkem celkové analýzy byl seznam funkčních a nefunkčních požadavků.

Dále byly zmíněny veškeré nedostatky, problémy a změny týkající se uživatelského rozhraní, na jejichž základě byl proveden návrh jednotlivých stránek. Některé věci byly nalezené při analýze, další problémy se potom projevily při diskuzích nad jednotlivými verzemi návrhu. Návrh probíhal v několika iteracích, postupným zlepšováním a opravováním chyb než se došlo k výslednému návrhu.

Následně proběhla implementace prototypu. Vytvořil jsem základní strukturu nového modulu včetně hlavních komponent a stránek. Vyzkoušel jsem si roli manažera vývojového týmu 1, který se podílel na vývoji projektu DBS v rámci předmětu BI-SP1. Výsledný prototyp, jehož vývoj jsem řídil, nebo se na jeho vývoji podílel je z pohledu uživatelského rozhraní funkční a je připraven pro další vývoj, kterým bude zejména připojení na backend část modulu, která se v současnosti mění, pomocí API, které bylo připravené v dalším vývoji v rámci předmětu BI-SP2.

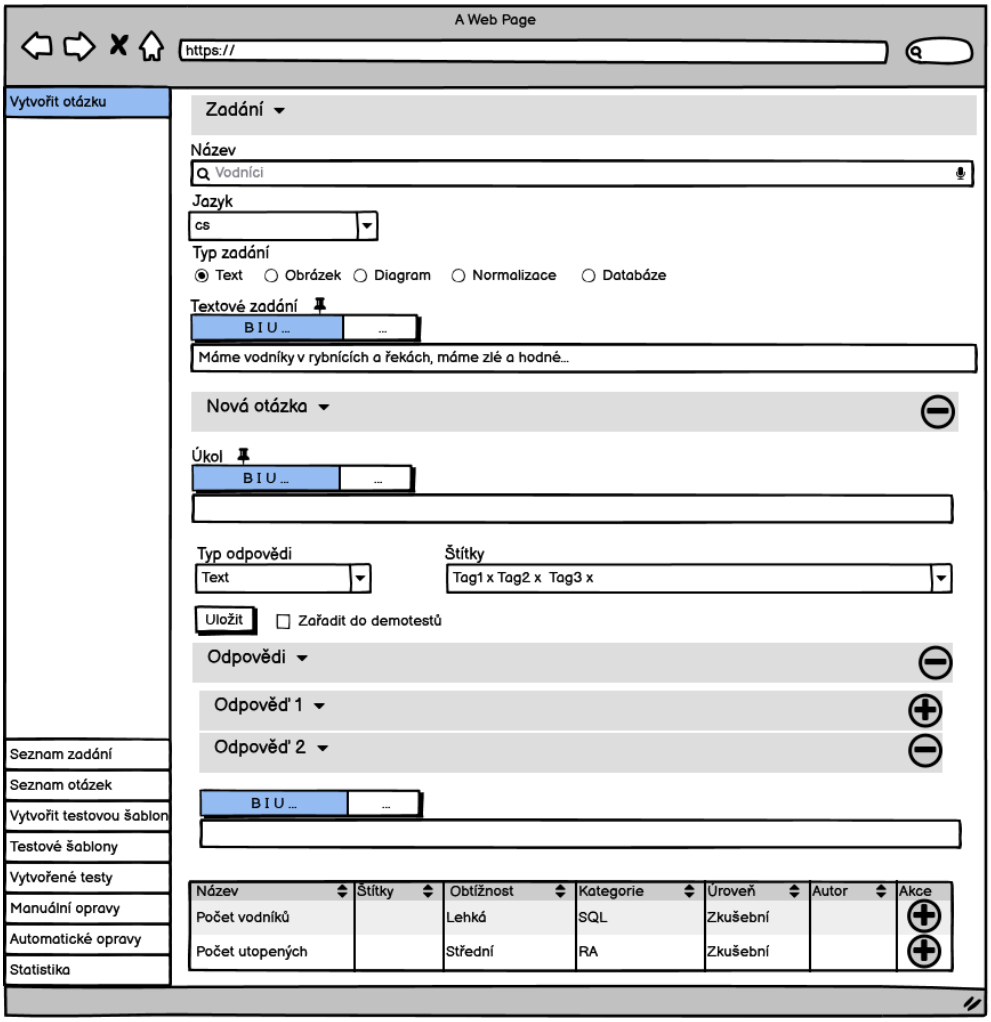
Prototyp jsem dále důkladně otestoval se čtyřmi uživateli. Z tohoto testování jsem obdržel zpětnou vazbu na uživatelské rozhraní, kterou jsem vyhodnotil a následně navrhl nebo opravil nalezené problémy. Výstup z testování jsem zapsal do aplikace Redmine, odkud bude vycházet další tým nebo člověk co bude na tuto práci navazovat.

Bohužel v současnosti ještě neexistuje implementace API pro nový testový modul v DBS portálu. Navíc je opět předělávaná backend část testového modulu, tentokrát na základě mikroslužeb. Implementace této části probíhá v rámci diplomové práce Bc. Andriiho Plyskache. Z tohoto důvodu není možné plně integrovat prototyp do portálu DBS. Nově navržené uživatelské rozhraní je však funkční a dá se plně a jednoduše využít pro další rozvoj. Je nahrané v gitu na samostatné větvi.

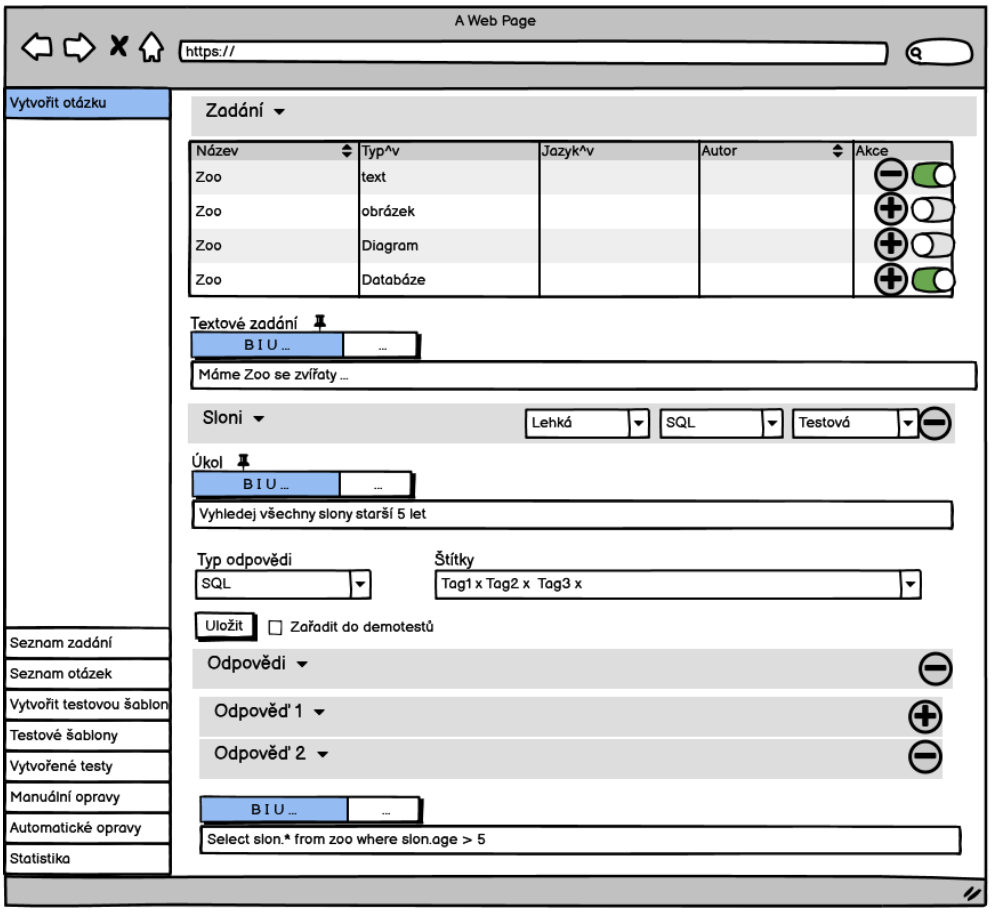
V současné době již proběhl další vývoj, který mimo jiné sloučil všechny nově tvořené prvky testového modulu v rámci bakalářských prací a práce odvedené v rámci předmětů BI-SP1 a BI-SP2 všemi týmy. Připravuje se nový backend a sjednocuje se způsob předávání dat za pomocí API mezi frontend a backend částí. Než bude nově vytvořený modul nasazen na produkční verzi a využíván jak učiteli, tak studenty, ještě nějakou dobu potrvá, ale vývoj směřuje správným směrem a celkový výsledek bude jistě na vysoké úrovni.

Příloha A

Wireframe návrhy



Obrázek A.1 Wireframe: Druhá verze návrhu stránky *Vytvořit zadání*



Obrázek A.2 Wireframe: Nepoužitá verze návrhu stránky *vytváření zadání*

Příloha B

Celý testovací scénář spolu s úkoly

Scénář 1

Aktuálně si chystáte test pro vaše cvičení. Máte už připravené otázky, a potřebujete jej vytvořit na portálu DBS.

1. Otázka na vytvoření diagramu

Připravený diagram máte uložený v D:\Downloads\BP testování\Zoo.json

Připravená otázka:

Vytvořte diagram pro dané schéma.

Databáze zoo:
Navstevnik - je identifikován umělým atributem **id_navstevnik**, dále povinně evidujeme **vek**, a volitelně **jmeno** a **prijmeni**.
Sekce - je identifikována umělým atributem **id_sekce**, dále evidujeme **nazev**.
Vstupenka - je identifikačně závislá na Návštěvníkovi a Sekci. Dále evidujeme **datum_platnosti** a **typ**. Datum_platnosti vyjadřuje den, kdy je vstupenka platná.
Urednik - je identifikován atributem **rodne_cislo**, dále evidujeme **jmeno**, **prijmeni** a **pozice**.
Každou vstupenku vlastní právě jeden návštěvník. Návštěvník za svou historii může vlastnit více vstupenek.
Pro každou sekci má návštěvník jinou vstupenku, nicméně platí, že v daný den může konkrétní Návštěvník vlastnit pro danou Sekci Vstupenku jen jednu.
Vstupenky prodává úředník, platí že jeden úředník může prodat libovolné množství vstupenek. Vstupenku musel prodat jeden konkrétní úředník
Dále evidujeme vztah sponzoruje. Návštěvník může sponzorovat libovolné množství Sekcí, Sekce může být sponzorována libovolným množstvím Návštěvníků.
Vazby nepopisujte (není třeba psát textový popis k vazbám).

2.Otázka na transformaci

Připravený diagram máte uložený v D:\Downloads\BP testování\Kolej.json

Připravená transformace:
kolej(cislo_popisne, *ulice)
navsteva(cislo_pokoje, cislo_popisne, rodne_cislo)
pokoj(cislo_pokoje, cislo_popisne)
student(rodne_cislo, *jmeno, *prijmeni, cislo_pokoje, cislo_popisne)
vybaveni(inventarni_cislo, *cislo_pokoje, *cislo_popisne, *nazev, *rok_evidence)
navsteva[rodne_cislo] ⊆ student[rodne_cislo]
navsteva[cislo_pokoje, cislo_popisne] ⊆ pokoj[cislo_pokoje, cislo_popisne]
pokoj[cislo_popisne] ⊆ kolej[cislo_popisne]
student[cislo_pokoje, cislo_popisne] ⊆ pokoj[cislo_pokoje, cislo_popisne]
vybaveni[cislo_pokoje, cislo_popisne] ⊆ pokoj[cislo_pokoje, cislo_popisne]

Připravené detaily k otázce upřesňující co od studenta chcete:
Transformujte dále uvedené ER schéma do relačního schématu. Použijte zjednodušený textový relační zápis.
Klíče podtrhávejte, povinnost atributů musíte vyznačovat pomocí prefixu *. Podtržené klíče se berou jako povinné automaticky – tam hvězdičku nepište.
Příklad správného použití:
relace1(atribut4, atribut5, *atribut6)
relace2(atribut1, atribut2, atribut3)
relace3(atributx, *atribut4)
relace2[atribut1]⊆relace1[atribut4]
relace3[atribut4]⊆relace1[atribut4]

Doporučení:
Podtrhávejte atributy tak, aby byl podtržen pouze daný atribut a ne závorky, či čárka za posledním atributem, který je součástí klíče – vizte ukázku výše!
Integritní omezení check netřeba psát, unique u atributu neřešte.
Korektnost zápisu kontroluje automat ihned při psaní. V případě komplikací se prosím přihlaste.
Složené klíče podtrhávejte, pokud je třeba, dohromady, jinak budou brány jako samostatné klíče!
Případné relace vzniklé dekompozicí pojmenovávejte dle popisu vazby v diagramu. Pokud vzniklé entitě budete dávat umělý klíč, využijte spojení id_pojmenovanientity.
Pokud to není nutné, atributy nepřejmenovávejte a držte se striktně podkladu z konceptuálního schématu. Názvy relací i atributů pište pouze malými písmeny.
Pokud musíte přejmenovávat atribut, můžete použít předpis: id_nazevrelace (relace na tabulku ze které je daný cizí klíč) nebo nazevrelace (relace na tabulku ze které je daný cizí klíč). Pokud nemusíte, nepřejmenovávejte.

Otázky 3 až 5 na Relační algebru a SQL

Připravená databáze: Máte již připravenou na portále pod názvem “Databáze 2”

Připravené relační schéma pro otázky:
Mějme fragment relačního schématu:
Osetrovatel (id_osetrovatel, jmeno_osetrovatel, adresa, plat)
Krmeni (id_osetrovatel, id_zvire, datum, cas, typ_krmiva, mnozstvi)
Zvire (id_zvire, jmeno_zvire, druh, datum_narozeni, id_kmotr)
IO: Krmeni[id_osetrovatel] ⊆ Osetrovatel[id_osetrovatel]
Krmeni[id_zvire] ⊆ Zvire[id_zvire]
Zvire[id_kmotr] ⊆ Osetrovatel[id_osetrovatel]
Vztah mezi ošetřovatelem a zvířetem (Zvire[id_kmotr] ⊆ Osetrovatel[id_osetrovatel]) nazývejme kmotrovství. Tedy ošetřovatel může dělat kmotra některému zvířeti. Tento vztah nesouvisí s krmením.

Cíl 3. otázky na RA:

Pomocí relační algebry vyberte **jména zvířat, která byla** již někdy **nakrmena**. (mají záznam o krmení).
Vzorová odpověď na 3. otázku:
{Zvire*krmeni}[jmeno_zvire]

Cíl 4. otázky na RA:
Pomocí relační algebry vyberte **ošetřovatele (id_osetrovatel, jmeno_osetrovatel, adresa, plat)**, kteří **krmili opice** (na jméně nezáleží) **POUZE 1. dubna 2015** (v dotazu zadejte '1.4.2015')
Vzorová odpověď na 4. otázku:
{zvire(druh='opice') * krmeni(datum='1.4.2015') * osetrovatel}[id_osetrovatel,jmeno_osetrovatel,adresa,plat] \ {zvire(druh='opice') * krmeni(datum!='1.4.2015') * osetrovatel}[id_osetrovatel,jmeno_osetrovatel,adresa,plat]

Cíl 5. otázky na SQL:
Pomocí SQL vyberte **zvire**(id_zvire, jmeno_zvire, druh, datum_narozeni, id_kmotr)**, které nikdy nikdo nekrmil.**
Vzorová odpověď na 5. otázku:
select * from zvire z where not exists (select 1 from krmeni k where k.id_zvire=z.id_zvire);

Otázky 6 až 8 na teorie

Otázka 6:
Které z uvedených příkazů lze v SQL použít pro spojení dvou tabulek?
Připravené možnost (poslední dvě jsou správné):
JOIN WHEN
NULL JOIN
JOIN EXCEPT
LEFT OUTER JOIN
INNER JOIN

Otázka 7:
Pokud na dotaz:
SELECT COUNT(*) FROM tabulka_A CROSS JOIN tabulka_B;
Odpověděl databázový stroj: 256
Co odpoví databázový stroj na následující dotaz, pokud se data mezitím nijak nezměnila?
SELECT COUNT(*) FROM tabulka_C;
Připravené možnosti (poslední je správná):
16
128
256
Nelze určit, protože nevíme nic o tabulce C.

Otázka 8:
Mějme relaci R71(a, b, c, d, e)
Je relace R71 ve třetí normální formě (3NF)? Proč ano, proč ne?
Vzorové odpovědi na otázku:
Relace R71 je v 3NF, protože vedle určení klíče nemá přidané žádné funkční závislosti.

Relace R71 je ve třetí normální formě, protože vedle určení klíče nemá přidané žádné funkční závislosti.

Pokračování scénáře:
Nyní z vytvořených otázek připravte testovou šablonu, z které byste později spouštěl test pro své studenty. U testu nastavte pro své studenty časový limit na 60 minut. Celkový prostor pro test (začátek a konec) máte v kosu v jednorázové akci nastaven od 12 do 16 hodin, přizpůsobte tomu dané nastavení. Maximum z testu i otázek nechám na vás.

Scénář 2

Chcete studenty seznámit s vyplňováním testů. Proto jste se rozhodl pro ně připravit demo test, který bude obsahovat jednu náhodnou **zkušební** otázku ke každého typu (**RA, SQL, transformace** atd.)
V systému už jsou všechny potřebné otázky vytvořené.

Příloha C

Zpracovaný výstup z uživatelského testování

Zadání

vysoká priorita: znovupoužitelnost obecných zadání (zadání s obecnými pokyny aby šlo vkládat i do jiných zadání) - možný způsob řešení: přidání nového přepínacího tlačítka do komponenty části zadání, které by po přepnutí umožnilo spojit zadání s částí jiného zadání, nebo s celým zadáním, které obsahuje obecný popis)
špatné pojmenování balíček - přejmenovat na zadání a části zadání
editace názvu části zadání - kliknutí na tužku nezačne editaci
vyřešit otázky bez zadání - třeba přidat volbu “bez zadání”
vymyslet jak na otázky, které nepotřebují zadání? - např zadání typu “prázdné”, nebo jenom legendu, že jde prázdné zadání nechat)

Seznam zadání

přidat počet otázek u zadání
přidat nějakou agregaci štítků z otázek

Otázky

vytváření štítku by u otázky být nemuselo - většina štítků se vytvoří při nasazení nového testového modulu
při vytvoření nové otázky použít jenom placeholder text místo samotného názvu “Nová otázka”
štítek by šel automaticky přiřadit podle typu odpovědi
v boxu přidané otázky napsat v headeru kolik je otázek vytvořených

Odpovědi

u odpovědí nezkracovat ten preview tolik - použít celý volný prostor
u RA, SQL dát najevo, že výsledek funguje
sbalit odpověď po napsání
u transformace jsou malé tlačítka u tralexu
checkbox a radio odpovědi zabírají moc místa i po sbalení,
návrh - šel by vylepšit i způsob zadávání checkbox a radio odpovědí, aby šli vytvářet rychleji, jelikož jsou tyto odpovědi většinou maximálně na řádek šlo by přidat další odpověď přidat např. stisknutím tlačítka enter
návrh - u radio odpovědí validovat, že je jen 1 správná odpověď až dýl? aktuálně by to mohlo vypadat, že to nutí mít první správně
návrh - u odpovědí text, checkbox a radio zvážit editor s formátováním

Testové šablony

opravující u otázek v testové šabloně defaultně nastavit na autora otázky
chybí popisy, že je čas v minutách
chybí rozlišení typu testu (demo test, zkouška, v semestru) - po práci s SP1 to v tu bylo
v seznamu přidat otázky a přidané otázky místo sloupec název přejmenovat na Otázka
možnost po spuštění testu náhodně seřadit otázky v něm (je možné, že se toto řeší až při spuštění testu a ne v testové šabloně.
v seznamu přidat otázky zvolit lepší využití barev (zelená vypadá, jako by otázka byla v šabloně už přidaná)

návrh - umožnit psát např 4h 5h a převést na minuty
návrh - automaticky rozdělit body do testů třeba přes tlačítko - šlo by dále vyřešit podle štítků (lehká míň bodů než střední atd.)
návrh - při vybírání otázky pomocí štítků umožnit vygenerovat více otázek stejnou konfigurací štítků (šlo by přidáním sloupce počet, a body by se potom rozpočítali mezi daný počet otázek)
bug - ošetřit mínusové body (po nastavení vysokého počtu bodů, vyplnění bodů u otázek a následném snížení maximálního počtu se po úpravě bodů v jedné z otázek nastaví mínusové body)

Štítky a štítkové skupiny

štítky řadit abecedně
lépe indikovat, kde na stránce se dá vytvořit skupina nebo štítek (zelená barva, nebo tlačítko a komponentu na přidání zobrazit až po kliknutí atd.)

Obecné

bug - chybí titulky v tabech (v záložce v prohlížeči)
zvážit jednotnost designu zadání a otázky - jeden upravuje název v boxu, druhý uvnitř
u částí zadání, otázek a odpovědí chybí identifikace, zda je box rozbalený nebo ne (+/-)
všechny boxy zbytečně defaultně zapouzdřené
při kliknutí odstranit zadání/otázku se zobrazuje “Odstranit předmět” - buď zvýraznit detaily v boxu, kde se se píše co se teda odstraňuje, nebo nahradit slovo předmět
u odebrání prázdné otázky/zadání se neptat na odstranění
tlačítka přidat zadání, otázku a odpověď - přidat slovo “další” do tlačítek
v menu přejmenovat seznam zadání na seznam zadání s otázkami
v menu přejmenovat vytvořit zadání na vytvořit zadání a otázky
je potřeba název otázky a název části zadání? zamyslet se nad tím, ale nejspíše potřeba budou - například při přidávání do testové šablony se používá název otázky, co by tam šlo jiného?
Při kliku na uložit pokud není povinný atribut vyplněný přesunout uživatele aby viděl co je špatně - například u zadání pokud není vyplněný název a uživatel klikne uložit poté co už přidal několik otázek a nemá je sbalené si je neví co je špatně
sjednotit co se stane po kliknutí na tlačítka uložit celé stránky (u šablon se na stránce zůstává, u zadání se přechází k vytvoření dalšího zadání) - šlo by například přidat tlačítka “uložit a vytvořit další”
přidat tlačítka uložit k jednotlivým komponentám (otázka, odpověď, části zadání), které budou zároveň odesílat data pro API
stylování vnořených boxů - není moc poznat co je vnořené
promyslet spojení položek v menu (zadání + seznam spolu, to samé u šablon) nebo například část vytváření spolu, část spouštění a opravování spolu)

Literatura

[1] Malec Oldřich. *Řízení projektu a infrastruktury portálu pro podporu výuky předmětu BI-DBS*. České vysoké učení technické v Praze, Fakulta informačních technologií, Praha, 2017. Bakalářská práce.

[2] Plyskach Andrii. *Refaktoring testové části backendu portálu dbs.fit.cvut.cz*. České vysoké učení technické v Praze, Fakulta informačních technologií, Praha, 2020. Bakalářská práce.

[3] Nette Foundation. Nette framework. [online] 2008 [Cit. 2021-04-15]. URL: `https://nette.org/cs/`.

[4] Špaček Petr. Architecture and design patterns. [online] 2018 [Cit. 2022-04-15]. URL: `https://gitlab.fit.cvut.cz/NI-ADP/ni-adp/blob/B191/media/lectures/2018/miadp-2018-lecture03-04-mvcgame-patterns.pdf`.

[5] Nette Foundation. Presenters. [online] 2008 [Cit. 2021-04-15]. URL: `https://doc.nette.org/cs/application/presenters`.

[6] Čapka David. Lekce 2 - uml - use case diagram. *Ajťácká sociální síť a materiálová základna pro C, Java, PHP, HTML, CSS, JavaScript a další*. [online] [Cit. 2021-02-25]. URL: `https://www.itnetwork.cz/navrh/uml/uml-use-case-diagram`.

[7] Mlejnek Jiří. Přednáška 3. *Analýza a sběr požadavků - případy užití*. [online] 2020 [Cit. 2021-02-25] [Potřeba přihlášení do sítě ČVUT - kopie souboru uložena na přiloženém médiu]. URL: `https://moodle-vyuka.cvut.cz/pluginfile.php/308910/course/section/46035/3_RequirementsEngineering.pdf`.

[8] Software engineering. *Classification of Software Requirements*. [online] 2018 [Cit. 2021-02-25]. URL: `https://www.geeksforgeeks.org/software-engineering-classification-of-software-requirements`.

[9] Obstova Barbora. Uživatelské rozhraní. [online] 2013 [Cit. 2021-05-01]. URL: `https://wikisofia.cz/wiki/Uživatelské_rozhraní`.

[10] Pavlíček Josef. User interface design. [online] 2021 [Cit. 2021-05-01]. URL: `https://docs.google.com/presentation/d/10IjuumeQ0b-t3YZkqb8J7BNos8K-usoYgXj1xmQIkuA/edit#slide=id.g99ecd62e4f_2_0`.

[11] Dubinská Lída. *Co je wireframe webu, proč ho potřebujeme a jak ho vytvořit*. [online] 2021 [Cit. 2021-05-22]. URL: `https://www.rascasone.com/cs/blog/co-je-wireframe-predstavujeme-5-duvodu-proc-je-pro-klienty-drateny-model-dulezity`.

[12] Balsamiq Studios. 2008. [online] 2021 [Cit. 2021-02-25]. URL: https://balsamiq.com.

[13] Html & css. *W3C*. [online] 2016 [Cit. 2021-04-05]. URL: https://www.w3.org/standards/webdesign/htmlcss.

[14] Introduction. *Vue.js*. [online] 2021 [Cit. 2021-04-05]. URL: https://v3.vuejs.org/guide/introduction.html.

[15] Krupička David. *Co je to Git a jak ulehčí práci*. [online] 2019 [Cit. 2021-05-22]. URL: https://starkmedia.cz/blog/co-je-to-git-a-jak-ulehci-praci.

[16] Svačinka Martin. *Úvod do Vagrantu*. [online] 2015 [Cit. 2021-05-22]. URL: https://zdrojak.cz/clanky/uvod-do-vagrantu/.

[17] Halenka Michal. *VirtualBox: operační systém nanečisto*. [online] 2014 [Cit. 2021-05-22]. URL: https://www.linuxexpres.cz/software/virtualbox-operacni-system-nanecisto.

[18] apt-get command in linux with examples. *GeeksforGeeks*. [online] 2019 [Cit. 2021-02-25]. URL: https://www.geeksforgeeks.org/apt-get-command-in-linux-with-examples/.

[19] Davis Harry. *Why apt-get update is not working?* [online] 2021 [Cit. 2022-05-05]. URL: https://quick-adviser.com/why-apt-get-update-is-not-working/.

[20] JetBrains s.r.o. *PhpStorm*. [online] 2000 [Cit. 2021-05-22]. URL: https://www.jetbrains.com/phpstorm/.

[21] Koppers Tobias. *webpack*. [online] 2016 [Cit. 2021-05-22]. URL: https://webpack.js.org/.

[22] Migration from vue 2. *Vue.js*. [online] 2014 [Cit. 2021-04-05]. URL: https://v3.vuejs.org/guide/migration/introduction.html#overview.

[23] Dulisz Damian. Multiselect. [online] 2021 [Cit. 2021-04-10]. URL: https://vue-multiselect.js.org/.

[24] Discord Inc. Představ si místo... [online] 2015 [Cit. 2022-05-01]. URL: https://discord.com.

[25] Lang Jean-Philippe. Redmine. [online] 2006 [Cit. 2022-05-01]. URL: https://www.redmine.org.

[26] Grossmann Lukáš. Lekce 5 - extrémní programování. [online] 2020 [Cit. 2022-05-01]. URL: https://www.itnetwork.cz/navrh/metodiky/extremni-programovani.

[27] Sadhna Goyal. Major seminar on feature driven development. *Jennifer Schiller Chair of Applied Software Engineering*, 2008.

[28] Vuex. What is vuex? [online] 2022 [Cit. 2022-05-06]. URL: https://vuex.vuejs.org/.

[29] championswimmer. vuex-persist. [online] 2022 [Cit. 2022-05-06]. URL: https://www.npmjs.com/package/vuex-persist.

[30] Kitner Radek. Co je testování softwaru? [online] 2021 [Cit. 2021-06-01]. URL: https://kitner.cz/testovani_softwaru/co-je-testovani-softwaru/.

[31] Voják Michal. Jak dělat uživatelské testování. [online] 2020 [Cit. 2022-05-07]. URL: https://designdev.cz/jak-delat-uzivatelske-testovani.

[32] Nielsen Jakob. Why you only need to test with 5 users. *Nielsen Norman Group*. [online] 2000 [Cit. 2022-05-07]. URL: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/.

[33] Voják Michal. Příprava testu použitelnosti. [online] 2020 [Cit. 2022-05-07]. URL: https://designdev.cz/priprava-testu-pouzitelnosti.

[34] Voják Michal. Průběh testu použitelnosti. [online] 2020 [Cit. 2022-05-07]. URL: https://designdev.cz/prubeh-testu-pouzitelnosti.

[35] AnyDesk Software GmbH. Access.now. [online] 2022 [Cit. 2022-05-07]. URL: https://anydesk.com/en.

[36] Microsoft. Microsoft teams. [online] 2022 [Cit. 2022-05-07]. URL: https://www.microsoft.com/cs-cz/microsoft-teams/group-chat-software.

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `src`
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `diagramy` ..... složka s diagramy
    - `usecase.eapx` ..... diagramy případů užití
  - `wireframe` ..... složka se zdrojovým souborem drátěných návrhů
    - `BP_wireframe.bmpr` ..... zdrojový soubor s wireframe návrhy
- `BP_Jordan_Pavel_2022.pdf` ..... text práce ve formátu PDF
- `zdroje` ..... složka s kopií zdrojů dostupných ze sítě ČVUT
- `testovani` ..... složka se záznamy s uživatelského testování