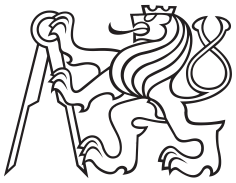
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Aplikace pro tvorbu učebních materiálů a testování studentů

Frontendová část

Tomáš Geržičák

Vedoucí: Ing. Petr Aubrecht, Ph.D.
Obor: Softwarové inženýrství a technologie
Květen 2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Geržičák** Jméno: **Tomáš** Osobní číslo: **474574**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Software pro vytváření školních testů, Frontent

Název bakalářské práce anglicky:

Software for School Tests, Frontend

Pokyny pro vypracování:

Během lockdownu trpí učitelé nedostatkem softwarové podpory pro vzdálenou výuku, speciálně na základních školách. Zaměřte se na oblast testů, automatizovaného zkoušení, dotazníků ap. Tato práce má za úkol implementovat frontent. Úkolem je navrhnout a implementovat vytváření dotazníků, které budou použity pro zkoušení vyučované látky na školách. Analyzujte, jaké typy dotazníků je potřeba vytvořit. Několik nejdůležitějších typů implementujte. Tato práce má za úkol implementovat backend.

1. Zanalyzujte a popište aktuální situaci v oblasti online dotazníků.
2. Navrhněte řešení, které by učitelům usnadnilo vytvoření dotazníků, snadné zadávání, použití pro žáky, kontrolu správnosti.
3. Řešení implementujte v technologii JakartaEE.

Příklady typů dotazníků, které určitě musí být obsažené, jsou výběr jedné možností, výběr z více možností, plain text, číslo.

Seznam doporučené literatury:

[1] The Jakarta EE 8 Tutorial: https://eclipse-ee4j.github.io/jakartaee-tutorial/toc.html
[2] Jakarta EE Cookbook - Second Edition, https://www.packtpub.com/programming/jakarta-ee-cookbook-second-edition
[3] Patterns of Enterprise Application Architecture — Martin Fowler

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Petr Aubrecht, Ph.D. katedra počítačů FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **08.02.2022** Termín odevzdání bakalářské práce: **20.05.2022**

Platnost zadání bakalářské práce: **30.09.2023**

Ing. Petr Aubrecht, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. | Datum převzetí zadání

Podpis studenta

Poděkování

Tímto bych rád poděkoval Ing. Petru Aubrechtovi, Ph.D. za pomoc při vypracování této práce a Janu Krčilovi za spolupráci. Také bych chtěl poděkovat svojí rodině a blízkým a hlavně své přítelkyni za podporu po celou dobu mého studia na ČVUT.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 20. května 2022

Abstrakt

Tato práce se zabývá vytvořením klientské části webové aplikace pro tvorbu online testů a procvičovacích materiálů. Aplikace je primárně vyvinuta pro učitele jako podpora jejich výuky a zefektivnění jejich práce, ale také pro žáky, kteří si mohou na vytvořených testy otestovat své znalosti. Inovuje stávající řešení, jak učitelé vytváří testy pro své žáky.

Práce analyzuje, jak učitelé vytváří testy dnes a jaké programy k tomu používají. Následně definuje požadavky na nový systém.

Součástí je návrh UI aplikace a popis architektury aplikace. Následuje implementace základní tvorby testových šablon a generování testů pro žáky.

Klíčová slova: Tvorba testů, Typescript, Javascript, React.js, Next.js, React-Query UI, UX, frontend

Vedoucí: Ing. Petr Aubrecht, Ph.D.

Abstract

The aim of the bachelors thesis is to create client web aplication for creating exams and teaching materials. This application is primarily developed for teachers to support their teaching and streamline their work, but also for students who can test their knowledge on the created tests.

This thesis innovates existing solutions as teachers create tests for their students. The work analyzes how teachers create tests today and what programs they use to do so.

It then defines the requirements for the new system. It includes the design of the application UI and a description of the application architecture. This is followed by the implementation of the basic creation of test templates and the generation of tests for students.

Keywords: Tests creation, Typescript, Javascript, React.js, Next.js, React-Query, UI, UX, frontend

Title translation: Application for creating exams and studying templates — Front-end part

Obsah

1 Motivace **1**

2 Požadavky **3**

2.1 Funkční požadavky 3

2.2 Nefunkční požadavky 6

3 Analýza **7**

3.1 Současná situace 7

3.2 Existující řešení 8

3.3 Inspirace 10

3.4 Rozhovory s učiteli 12

3.5 Použité technologie 15

3.6 Další použité nástroje 22

4 Architektura **25**

4.1 UI návrh řešení 25

4.2 Architektura frontendové části . . 28

5 Části aplikace **35**

5.1 Tvorba testu 35

5.2 Vyplnění testu 36

5.3 Učitelský přehled testu 38

5.4 Úvodní stránka 38

6 Další zkušenosti z implementace **41**

6.1 První implementace pomocí Vue.js 41

6.2 Vývoj komunikace s backendem . 43

6.3 Spolupráce s vedoucím práce . . . 45

7 Problémy s prací **47**

8 Plány do budoucna **51**

8.1 Testování mezi učiteli 51

8.2 Další možnosti v editoru testových šablon 51

8.3 Rozdělení aplikace do microservice 55

8.4 Implementace tisknutelné verze testu 58

8.5 Statistika výsledků 59

9 Závěr **61**

A Seznam zkratek **63**

B Literatura **65**

Obrázky

3.1 Ukázka tvorby testové otázky v programu Microsoft Forms 9

3.2 Ukázka tvorby testu v programu Moodle 10

3.3 Ukázka tvorby testu v programu Kahoot 11

3.4 Test z platformy Help for English 12

3.5 Test z platformy Treehouse 13

3.6 Ukázka výstupu nástroje ESLint 17

3.7 Ukázka komponent z Material UI. 19

3.8 Ukázka výpisu chyb z platformy sentry. 24

4.1 Návrh sitemapy aplikace 26

4.2 Ukázka barevných schémat, ze kterých jsem vycházel při tvorbě svého barevného schématu. 27

4.3 Výsledná loga pro dark mode (nahoře) a light mode (dole) 27

4.7 Ukázka update requestu a komunikace s backendem. 29

4.8 Ukázka vyrenderovaných typografických komponent 30

4.4 Proces návrhu loga. 32

4.5 Ukázka návhu low fidelity komponent pro studenta 33

4.6 Ukázka návhu low fidelity komponent pro učitele 33

4.9 Rozdíl UI mezi dark modem a klasickým 34

5.1 Ukázka tvorby testové šablony. . 39

5.2 Ukázka uživatelského testu 39

5.3 Ukázka vyhodnocení testu studenta 40

5.4 Ukázka přehledu testu pro učitele. 40

6.1 Ukázka vytvořených endpointů v Postmanu se svými příklady . . . 44

8.1 Návth možné podoby tvorby interaktivní práce s obrázky. Tvorba v šabloně nahoře a podoba v testu dole. 53

8.2 Návth možné podoby tvorby interaktivní práce s obrázky. Tvorba v šabloně nahoře a podoba v testu dole. 54

8.3 Návth možné podoby tvorby interaktivní práce s obrázky. Tvorba v šabloně nahoře a podoba v testu dole. 56

8.4 Sekvenční diagram funkce microservice pro sbírání odpovědí z testu. 57

8.5 Koncepční návrh systému pomocí microservice architektury. 58

Kapitola 1

Motivace

Učitelé základních škol řeší problém tvorby testů a celkového testování žáků za doby online výuky. Vzhledem k tomu, že moje mamka je učitelka matematiky na základní škole ve Vizovicích, mám k tomuto tématu osobní vztah. Dále jsem zde také viděl možnost tuto práci zefektivnit. Většinu testů vytváří tak, že z učebnic přepisuje testy ručně nebo do programu Microsoft Word, a to pak posílá přes aplikaci Microsoft Teams. Dle jejích slov to tak dělá i většina učitelského sboru.

Učitelé ovšem narážejí na to, že když už žákům něco oskenovali a poslali, žáci si museli test vytisknout, vyplnit, nafotit a poslat zpět. Toto byl poměrně neefektivní proces jak pro žáky, tak pro učitele. Zároveň zde není jednotný program vytvořený přímo pro tvorbu testů a testování žáků.

Rád bych proto vytvořil program, který by mohli učitelé používat pro tvorbu testů a cvičebních materiálů a jelikož vím, že implementace takového systému je náročná, přizval jsem k této práci svého kolegu Jana Krčila, který se bude primárně starat o backendovou část aplikace. Tento program by tak měl každému učiteli šetřit čas a zdigitalizovat tvorbu testů a cvičebních materiálů.

Kapitola 2

Požadavky

2.1 Funkční požadavky

2.1.1 Vytvoření šablony testu

Učitel je schopný si vytvořit šablonu testu a pomocí této šablony se budou vytvářet konkrétní instance testů pro žáky. Tuto šablonu si může učitel pojmenovat a přidat k ní doporučený čas na vypracování testu.

Šablona se skládá z jednotlivých částí, ze kterých se poté bude generovat test. Aby se zajistilo, že vygenerovaný test bude unikátní, z každé části otázky se vybere náhodně jedna nebo více (podle nastavení) a z nich se poskládá daný test. Tuto část testu si může učitel pojmenovat, také volitelně specifikovat čas pro vypracování dané části a nastaví, kolik otázek se bude generovat (Standardně 1 otázka). Dále si pak přidává části šablony a do nich jednotlivé otázky.

2.1.2 Vytvoření online testu pro třídu

Učitel si vybere šablonu, ze které chce test generovat, poté třídu které chce test přiřadit. Vybere si, kdy se má test zpřístupnit, čas pro vyplnění testu

a kdy se test pro všechny uzavře. Šablona by měla automaticky nabídnout doporučený čas pro vyplnění testu sečtený z jednotlivých částí šablony.

Princip generování testu z šablony

Test se musí generovat pro každého žáka unikátně. Při vytváření testu pro konkrétní žáky se spustí algoritmus pro každého žáka zvlášť. Tento algoritmus bude postupně procházet šablonu testu, a z každé části šablony vybere náhodně počet otázek, které se mají generovat podle nastavení části šablony. Systém poté uloží danou konfiguraci pro každého žáka.

2.1.3 Vytvoření tisknutelného testu

Učitel má možnost si test také vytvořit ve formě PDF, který si může vytisknout pro prezenční výuku. Test se vygeneruje pro každého žáka unikátně a s jeho jménem.

2.1.4 Vytváření jednotlivých otázek

Šablony by měly obsahovat širokou nabídku otázek, aby generované testy byly použitelné pro téměř jakýkoliv předmět. Vytváření otázky bude rozdělené na 2 sekce: **Zadání** a **Odpovědi**.

Úprava zadání

Zadání by mělo být ve stylu úpravy textu ve wysiwyg editoru, zde by měla být možnost přidat obrázky, videa a zvukové záznamy.

Speciální typy zadání:

- **Matematické slovní úlohy** – Učitel by měl mít možnost vytvořit si otázku jako slovní úlohu do matematiky. Do textu zadání by měly jít

přidat proměnné místo čísel. Pro tyto proměnné by měla jít navolit čísla, která se budou do otázky generovat. Půjde buď zvolit dané kombinace pro celou otázku, nebo pro každou proměnnou zvlášť.

Učitel by také měl mít možnost přidat jednoduchý vzoreček, který bude přebírat proměnné definované v zadání. Pokud bude tento vzorec vyplněný, tak otázku bude moci opravit přímo systém.

Vytváření odpovědí

Odpovědi by měly jít tvořit zcela nezávisle na otázce.

Typy odpovědí:

- **Výběr jedné možnosti** – Učitel bude moci přidat možnosti a z toho vybrat jednu otázku, které je správná.
- **Výběr z více odpovědí** – Učitel bude moci přidat možnosti a z toho vybrat libovolný počet otázek, které jsou správné.
- **Spojování pojmů** – Učitel definuje 2 sloupce, kdy v každém bude stejný počet pojmů, poté označí ty, které k sobě patří. Studentovi se pojmy zobrazí zamíchané a on musí spojit pojem z jednoho sloupce se správným pojmem ze sloupce druhého.
- **Doplňování slov do textu** – Učitel má možnost definovat text, do kterého označí pole a správná slova. Při vyplňování otázky se slova poskládají na jednu stranu a student musí tato slova vybrat a správně přiřadit.

2.1.5 Vyplnění testu studentem

Student by měl být informován o tom, kolik má celkově času na vyplnění testu, kolik otázek již vyplnil a kolik času mu do konce testu zbývá.

2.2 Nefunkční požadavky

2.2.1 Jednoduchost používání

Aplikace bude vytvořena především pro učitele. Řada učitelů není zvyklá na denní používání počítače, a nebo jsou často frustrovaní používáním již hotových řešení, které mnohokrát nebývají uživatelsky přívětivé. Aplikace proto musí mít intuitivní rozhraní a ovládací prvky musí mít jasně vysvětlené chování.

2.2.2 Rychlost při vyplňování testu

Student musí být schopen vyplnit test bez ohledu na vytíženost serveru. To znamená, že pro přecházení mezi otázkami není nutnost stahovat jakákoliv data ze serveru. Tím pádem načítání dalších otázek neubírá čas studentovi při samotném vyplňování testu.

Kapitola 3

Analýza

Pro návrh aplikace jsem čerpal především z rozhovorů, které jsem vedl s mamkou. Bavili jsme se o její zkušenosti s tvorbou testů za doby online výuky. Dále jsem vyzpovídal i pár dalších učitelů, kteří buď byli mými kantory, nebo jsou mými přáteli, a s online výukou mají podobné zkušenosti. Poté jsem analyzoval již stávající řešení a aplikace, na které jsem narazil v průběhu svého života[1], které se zaměřují na tvorbu testů a cvičebních materiálů pro školy.

3.1 Současná situace

V současné době učitelé nemají moc možností, jak testy vytvářet. Typický učitel na základní nebo střední škole si vytváří testy v programu Microsoft Word, kdy si příklady buď vymýšlí nebo je ručně přepisuje z nějaké učebnice anebo ze sbírky příkladů. Ve Wordu je možnost vytvořit si jednoduché testy relativně rychle. Bohužel nastává problém při tvorbě pokročilých testů, kam se přidávají například cvičení s obrázky či s matematickým zadáním.

Dále je zde otázka přepoužívání dat. Moje mamka má například repozitář se všemi testy, které kdy vytvářela, z těchto testů potom ručně vybírá otázky, které kopíruje do testu, jež zrovna tvoří. Tohle se může zdát jako optimální

[1] Z těchto projektů jsem se inspiroval primárně pro návrh části systému pro studenty, kteří vyplňují test.

metoda, bohužel i tak jí toto kopírování sebere moc času a je s ní dlouhodobě nespokojená.

Někteří učitelé používají již předpřipravené testy, které se přikládají k učitelským vydáním učebnic. Typickým příkladem této skupiny jsou učitelé angličtiny. Z mé zkušenosti ze střední školy jsou tyto učebnice a připravené testy v nich velice kvalitní, ale existují pouze v tištěné podobě. I tak ale učitelům usnadňují práci tím, že jsou již hotové. Na druhou stranu je zde riziko, že si studenti tuto příručku pro učitele také pořídí a s tím mají k dispozici i řešení testu, který budou psát. Tímto tématem ale zabíhám mimo zaměření své bakalářské práce.

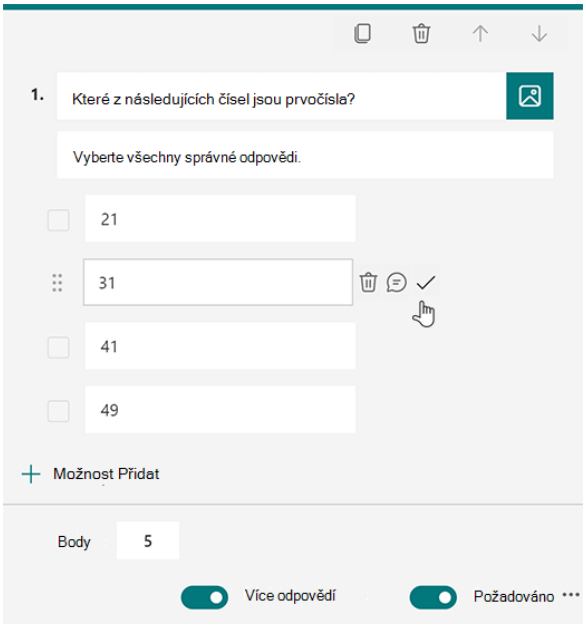
Během koronavirové pandemie v době online výuky se učitelé museli adaptovat na virtuální prostředí. To pro mnohé učitele nebylo vůbec jednoduché, jelikož se museli naučit pracovat s mnoha novými programy.

Tyto programy, které používají, nejsou ale primárně vytvořeny pro tvorbu testů 3.2 a vždy je to jen doplněk k funkci, pro kterou je aplikace určena. To je problém, protože učitelé nemají k dispozici jednotný nástroj pro tvorbu a správu testů a otázek k nim. Pokud by takový nástroj existoval, pomohl by učitelům zlehčit už tak jejich těžkou a namáhavou práci.

3.2 Existující řešení

3.2.1 Microsoft forms

Microsoft Forms je program od firmy Microsoft vytvořený primárně pro tvorbu dotazníků. To znamená, že zde lze vytvářet pouze jednoduché testy.[1] V jednom testu například nejde, aby každý žák dostal různé otázky, lze pouze zamíchat otázky mezi sebou. Na druhou stranu, je zde relativně dost možností tvoření otázek 3.4.2 Dle zkušeností učitelů 3.4.1 ve vyplňování testů žáci našli způsob, jak ve zdrojovém kódu klientské aplikace najít odpovědi. Velkou výhodou ale je, že Microsoft Forms se dají velice jednoduše použít v aplikaci Microsoft Teams, kde učitel jen vytvoří test a hned ho může zadat třídě žáků. Podle dotazovaných učitelů nejdou testy přepoužít pro více tříd ani pro další termíny.

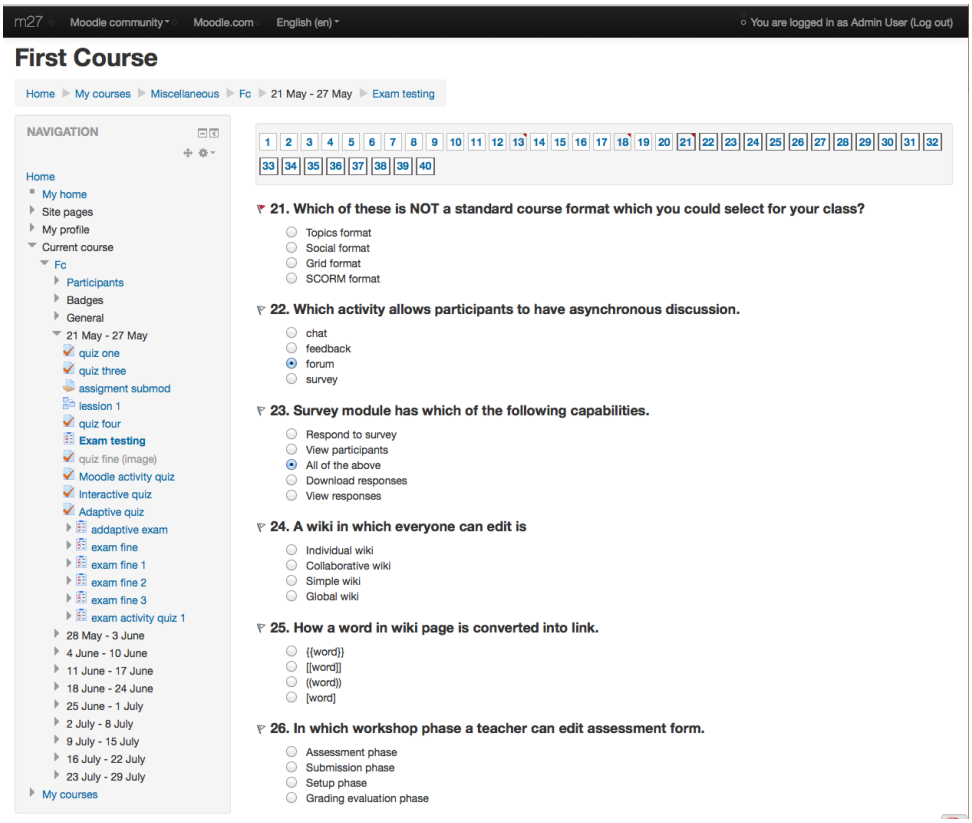


Obrázek 3.1: Ukázka tvorby testové otázky v programu Microsoft Forms

3.2.2 Moodle

Moodle je open-source CMS pro tvorbu výukových materiálů, který používá velké množství vysokých a středních škol pro správu učebních materiálů. Právě v tomto systému existuje možnost vytvořit zkouškový test se spousty typů otázek. Z rozhovorů s učiteli ale vím, že test není zase tak jednoduché vytvořit. Otázky se vytváří do jednoho adresáře a ty se pak přiřazují k testům. Z mých rozhovorů s učiteli však vyplynulo, že to není nejlepším řešením a špatně se s tím pracuje. Musím ovšem zmínit, že Moodle má obstojnou dokumentaci s video ukázkami, avšak pouze v angličtině.[2]

Další velký nedostatek Moodlu je to, že není dost dobře schopný reagovat na velký počet připojených studentů. Pro příklad bych uvedl situaci, kdy jsem v době online výuky psal test přes Moodle. Při téměř každém testu nastala situace, kdy se nás připojilo mnoho a Moodle spadl nebo se přinejmenším jen dost výrazně zpomalil. To nám bohužel všem ubíralo čas k dokončení zkoušky. Místo toho, abychom přemýšleli nad otázkou, jsme byli nuceni čekat, než se načte další stránka. To bohužel často trvalo někdy i několik minut.



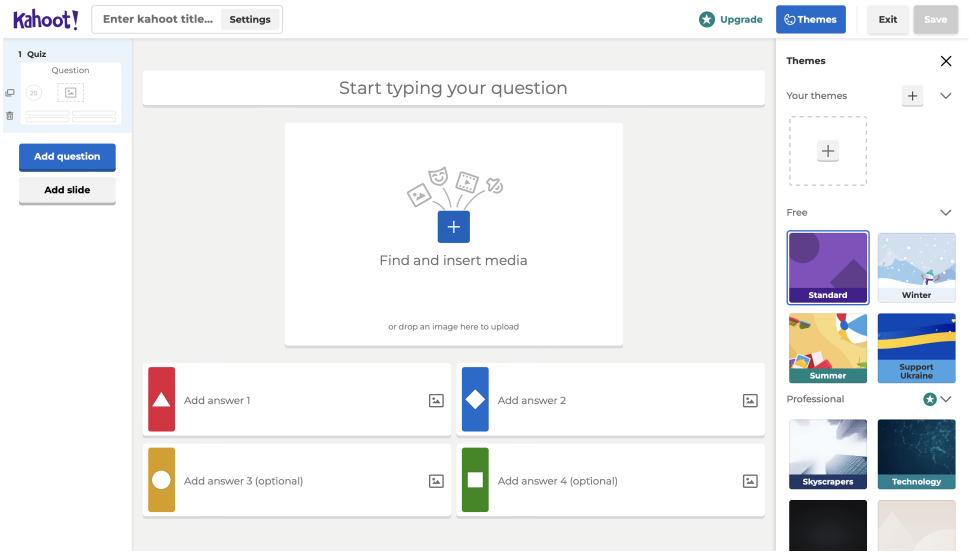
Obrázek 3.2: Ukázka tvorby testu v programu Moodle

3.2.3 Kahoot

Kahoot je online platforma pro vytváření kvízů a naučných materiálů. Cílí avšak spíš na mladší publikum a na hry pro děti, více než na tvorbu testů. Z rozhovorů vyšlo, že Edita Žáková 3.4.1 používá Kahoot pro odlehčení hodin. Jde vytvořit hra, kdy učitel vytvoří kvíz, ale pouze s jedním typem otázky, a to výběr ze čtyř odpovědí. Poté zveřejní QR kód, který si můžou žáci naskenovat pomocí chytrého telefonu a připojit se do hry. Dále potom učitel ukazuje otázky na projektoru a studenti odpovídají pomocí svých chytrých telefonů a získávají body.

3.3 Inspirace

Jako předlohou při tvorbě uživatelského rozhraní jsem se inspiroval různými prvky z uživatelského rozhraní několika programů.



Obrázek 3.3: Ukázka tvorby testu v programu Kahoot

3.3.1 Help for English

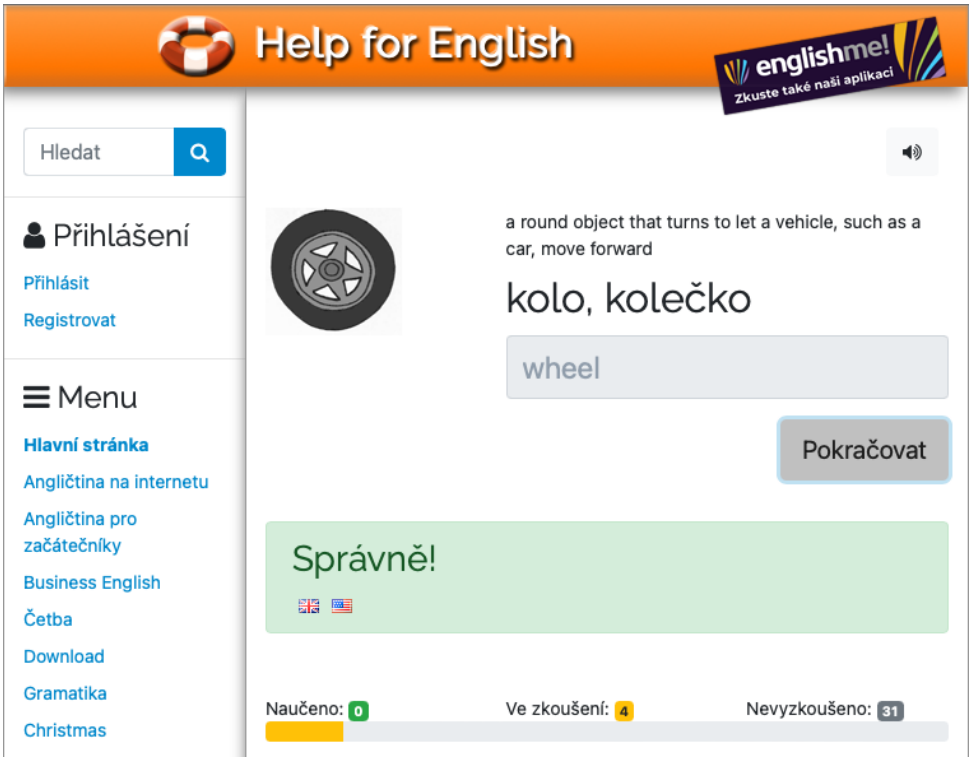
Help For English je portál pro podporu výuky angličtiny. [3] Jsou zde články, které vysvětlují základy angličtiny, ale hlavně je zde velká databáze slovní zásoby rozdělené do kategorií dle témat. Pro zlepšení výuky této slovní zásoby je ke každému tématu připravený test ze všech slovíček, kde si student otestuje své doposud nabité znalosti.

Na základní a střední škole jsem měl soukromou lektorku angličtiny, která mě s tímto portálem seznámila. Sám jsem se často za domácí úkol musel tato slovíčka učit a zmíněné testy mi v učení hodně pomohly.

Princip funkce je jednoduchý, studentovi se zobrazí test a ukáže se mu české slovo, které musí napsat anglicky. Po vyplnění se přehraje výslovnost slova.3.4 Velice se mi líbila logika tohoto testu, pokud student vyplnil slovo špatně, test dané slovo zařadil mezi další zkoušená slova dvakrát. Tudíž student nemohl test jen tak proklikat, ale musel se na daná slova opravdu soustředit.

3.3.2 Treehouse

Treehouse je platforma pro online výuku programování.[4] Výuka je zde rozdělena do jednotlivých kurzů, které rozebírají určitá témata. Kurz je poté



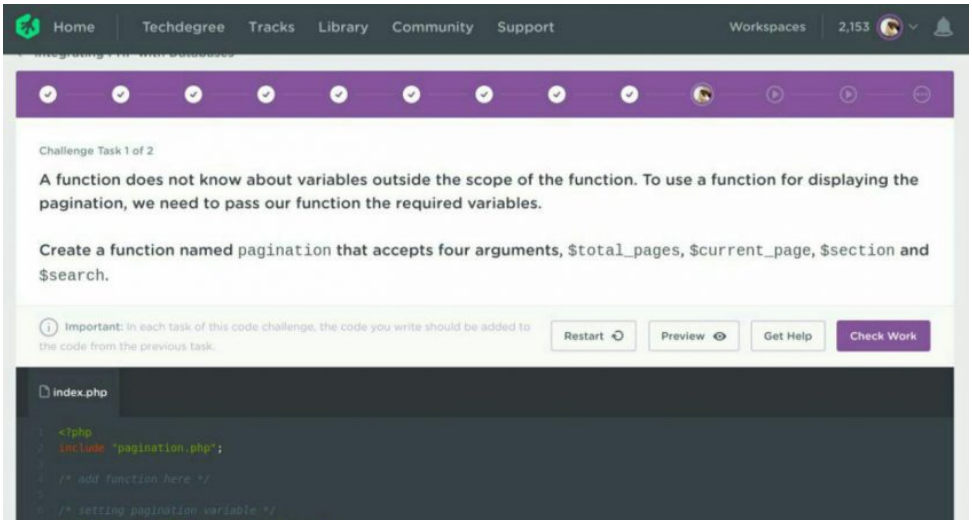
Obrázek 3.4: Test z platformy Help for English

rozdělen do lekcí, kdy každá lekce je zakončena testem znalostí z uplynulé lekce. A právě tyto testy se mi vždy velice líbily. Byly i dost rozmanité v typu otázek, a také se dobře podařila celé UI aplikace a interakce s uživatelem. Typy otázek zde byly klasické - Ano/Ne, dále vybírání z jedné či více odpovědí nebo také programovací úlohy.

Programovací úlohy byly buď jednodušší, kdy byl už kód napsán a student musel doplnit jen jedno, nebo více slov. Nebo potom těžší, kdy se studentovi zobrazil pouze editor a instrukce, co musí naimplementovat. 3.5

3.4 Rozhovory s učiteli

Když vytvářím něco pro učitele, chtěl jsem nejdřív slyšet jejich názor a pohled na věc. Zda vidí v mé práci do budoucna potenciál a zda jim bude užitečná a usnadní jim jejich práci. S každým učitelem jsem probíral, jak vlastně on a jeho kolegové v okolí připravují testy a cvičební materiály. Jak tyto testy vytvářeli za koronavirové pandemie, jaké programy k tomu používali, jestli se jim tyto programy líbily a co by na nich chtěli změnit.



Obrázek 3.5: Test z platformy Treehouse

3.4.1 Edita Žáková

Edit vyučuje na základní škole na prvním stupni ve Zlíně. Říkala mi, že tento rok učí sice jen žáky prvních tříd a tudíž moc testů není potřeba.

Když jsem jí představil moji vizi, zalíbila se jí především myšlenka určité jednoduchosti při zadávání testů. dále ocenila variantu, kdy systém vygeneruje pro každého studenta jiný test a že systém umí test i sám vyhodnotit. Protože i ona se mi přiznala, že tráví několik hodin tvorbou a následným opravováním testů. Z toho důvodu by ocenila, pokud by se celý proces dal zautomatizovat, a ona by tak svůj čas mohla vložit například na vytváření kreativních her do hodin svých nejmenších studentů.

Dále jsme diskutovali i možnost sdílení testů mezi kantory. Například na úrovni školy, kdy by se jí zamlouvalo mít s kolegy sdílený repozitář testů. Také však uznala, že se jí líbí myšlenka toho, kdy by byla možnost testy zpřístupnit každému dalšímu uživateli, a uživatelé by tak mohli testy hodnotit a navrhovat další nová vylepšení pro budoucí využití.

3.4.2 Kateřina Novotná

Kateřina Novotná studuje posledním rokem na přírodovědecké fakultě na Univerzitě Palackého v Olomouci. Konkrétně pak studuje obor geografie

a matematika, při svém studiu také absolvovala praxi na místním gymnáziu. A jelikož během její praxe probíhala online výuka, byla donucena vytvářet všechny testy a podpůrné materiály online.

Kateřina měla zdaleka nejvíce zkušeností s tvorbou online testů z dotazovaných učitelů. Vyzkoušela programy jako Microsoft Forms a Kahoot. Z těchto zmíněných programů používala primárně Microsoft Forms. Její zkušenost s daným programem je taková, že i přes to, že tento program není dokonalý, snažila se využít všechny jeho možnosti na maximum. Líbila se jí přímá integrace s aplikací Microstf Teams, dále také tvorba otázek, možnost test převést do PDF formátu a například i analýza odpovědí studentů. Jako nevýhody zmiňovala hlavně ne tak intuitivní uživatelské prostředí, nemožnost generování unikátních testů pro každého studenta. Dále poté například ještě zmínila, že vygenerovaný test nevypadá z grafické stránky nejatraktivněji. Vzhledem k tomu, že vyučuje geografii a potřebuje do testů vkládat i obrázky, nelíbila se jí práce s nimi. Svěřila se mi, že buď byly moc velké a z toho důvodu nebyl na běžném notebooku vidět zbytek otázky, nebo naopak moc malé, tudíž nebyl dobře rozeznatelný obsah. Také se jí nelíbila možnost, že se daný test nedá moc dobře přepoužívat mezi více třídami a také se zmínila, že zjistila, že při vyplňování testu nejde přeskočit otázka.

Když jsem jí představil naši vizi, velice se jí líbila možnost rozdělení testové šablony na části po stejných typech otázek a poté generování unikátních testů. Dále se jí líbila myšlenka parametrizovaných otázek do matematiky, jen namítla, že by bylo dobré, kdyby se zadání generovalo tak, aby bylo zajištěno, že studentům vyjdou "hezké"výsledky. Poté jsme opět řešili možnost sdílení svých materiálů mezi kolegy, tato myšlenka se stejně jako u Edit uchytila.

3.4.3 Ing. Pavel Náplava Ph.D.

Pan Náplava je náš učitel na FEL ČVUT a dle mé zkušenosti je jeden z nejlepších vyučujících. Když jsme diskutovali tvorbu testů, mluvil převážně o nepodařeném uživatelském prostředí aplikace Moodle, kterou při online výuce musel využívat. Také měl problém se způsobem generování testů, který dle jeho slov nebyl nejvydařenějším.

Jako další se zmiňoval o aplikaci Microsoft Forms [1], kdy mu nevyhovovalo málo možností pro tvorbu testových otázek.

Nakonec se nám zmínil o své vizi příběhových otázek – jak je sám nazval –

kdy chtěl vytvořit test tak, aby odpověď jedné otázky závisela na druhé. To by znamenalo, že by student odpověděl na jednu otázku a podle odpovědi by se mu zobrazila další. Musím uznat, že je to velice zajímavý nápad.

3.5 Použité technologie

3.5.1 React.js

React je open-source javascriptová knihovna pro tvorbu klientských aplikací, která je vyvíjena společností Facebook.[5] React se vyznačuje především deklarativním přístupem a velkou rychlostí. Hlavní stavební jednotkou je funkcionální komponenta [2] V Reactu se nepíše čistě HTML, ale používá se knihovna JSX, která umožňuje psát Javascript přímo do HTML.

3.5.2 Typescript

Typescript je programovací jazyk, vyvíjený firmou Microsoft. Jazyk je rozšířením standardu ECMAScript 5 a přidává například statické typování enumy, generiku a další možnosti.

Listing 3.1 Ukázka jednoduché komponenty v React.js s podporou Typescriptu

```
type Props = {
    index: number
    clicked: () => void
}
export const QuestionStep: FC<Props> = ({index, clicked}) => {
    return (
        <Button
            onClick={() => clicked()}
            variant={'contained'}
        >
            {index}
        </Button>
    )
}
```

[2] Dříve se react psal v class komponentách které byly tvořeny třídami, ale od toho se už dnes upustilo a třídy se používají jen ve specifických případech.[6]

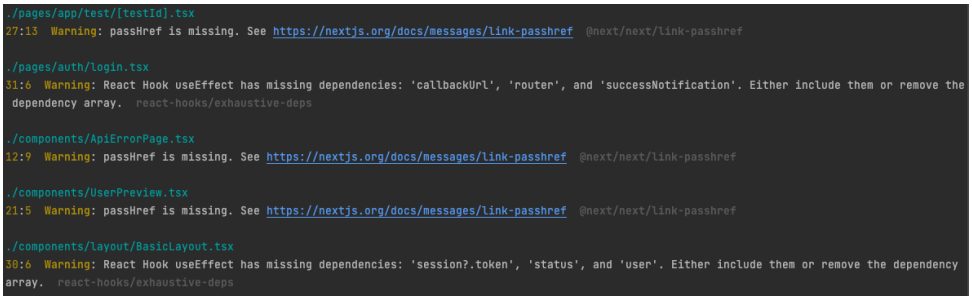
3.5.3 Next.js

Next.js je frontendový framework postavený nad Reactem. [7] Řeší mnoho běžných problémů, které aplikace v reactu mají bez toho, aniž by musel vývojář cokoliv konfigurovat. Při vytvoření aplikace příkazem `yarn create next-app` vytváří plně funkční aplikaci bez nutnosti jakékoliv další konfigurace.Taky zavádí pevnou adresářovou strukturu celé aplikace. Zavádí například tyto další vylepšení:

- **Routing** – Next řeší za vývojáře celý routing aplikace a řeší to velice chytře. V aplikaci existuje složka pages a do této složky může vývojář přidat komponenty nebo složky a podle jmen komponent a složek se vytváří routy v aplikaci.
- **Optimalizace obrázků** – Next dovoluje obalit obrázky do jeho tagu `<Image/>`, který optimalizuje obrázky na stránce.
- **Hot reload** – Toto je funkce, která po zapnutí aplikace hlídá změny ve všech souborech aplikace, když je nějaké soubor změněn, tak se sama překompiluje a vloží (injectne) změny do již fungující aplikace. Tato funkce opravdu šetří čas a usnadňuje vývoj.
- **Nativní podpora TypeScriptu** – Next přidává nativní podporu TypeScriptu a jeho automatickou konfiguraci.
- **API routes** – Možnost si vytvořit jednoduché testovací REST API. Tuto funkci jsem použil například tehdy, když jsem čekal, až se vytvoří skutečné endpointy na backendu. Pro simulaci skutečné komunikace s REST API jsem vytvořil endpointy pomocí Nextu, které vracely mock data. Po dokončení opravdových endpointů na backendu jsem jen změnil URL.
- **Next auth** – Je to plugin do Next.js, který se stará o zabezpečení jednotlivých stránek a autorizaci uživatelů.

3.5.4 NPM

NPM (Node.js package manager) je výchozí správce balíčků knihoven pro Javascriptový vývoj. Tyto balíčky se dají jednoduše nainstalovat do svého projektu pomocí příkazu `npm install ...`. Informace o těchto balíčcích v projektu jsou k dispozici v souboru `package.json`. V tomto souboru jsou taky specifikovány informace o celém projektu. [8] Je zde například jméno celého



Obrázek 3.6: Ukázka výstupu nástroje ESLint

projektu, seznam scriptů, pomocí kterých se dá spustit aplikace nebo spustit podpůrné programy, informace o licenci a další.

3.5.5 Webpack

Webpack je statický bundler pro moderní Javascript. Je to software, který pomáhá při překladu moderních webových aplikací do podoby, které jsou nejlépe čitelné pro prohlížeče. Když webpack zpracovává aplikaci, interně si vytvoří graf závislostí pro jeden a více entry pointů. Potom zkombinuje každý modul pro projekt do jednoho nebo více bundlů, což jsou statické assety, které čte prohlížeč.[9]

3.5.6 ESLint

ESlint je nástroj pro hledání chyb v Javascriptovém projektu. Staticky prohledá celý projekt a upozorní na možné chyby. Tento nástroj je integrován do většiny moderních editorů, a také může probíhat jako součást CI/CD. [10]

3.5.7 Prettier

Prettier je formátovač kódu. Pomocí prettieru se dá udržovat kód konzistentním mezi více vývojáři díky předdefinovanému stylu kódu, který se definuje sadou pravidel, jak se má kód formátovat. Prettier se poté spustí jednoduchým příkazem `prettier -write .`, který naformátuje celý kód podle specifikace.

Listing 3.2 Ukázka nastavení Prettieru

```
module.exports = {
  semi: false,
  singleQuote: true,
  trailingComma: 'all',
}
```

3.5.8 React Query

React query je knihovna, která se stará o fetch, caching a synchronizaci dat ze serveru. Zjednodušuje práci s rest API, která funguje na principu react hooků.[11]

Na rozdíl od Flux patternu React Query nepotřebuje tolik boilerplate kódu pro jednoduché věci, pouze se specifikuje endpoint, ze kterého chci data a o vše se postará React Query. React query řeší sama i cashing dat při opakovaných requestech nebo například opakované načtení dat při vrácení se myší do stránky. [12]

Listing 3.3 Ukázka použití React query pro stažení dat všech šablon testu

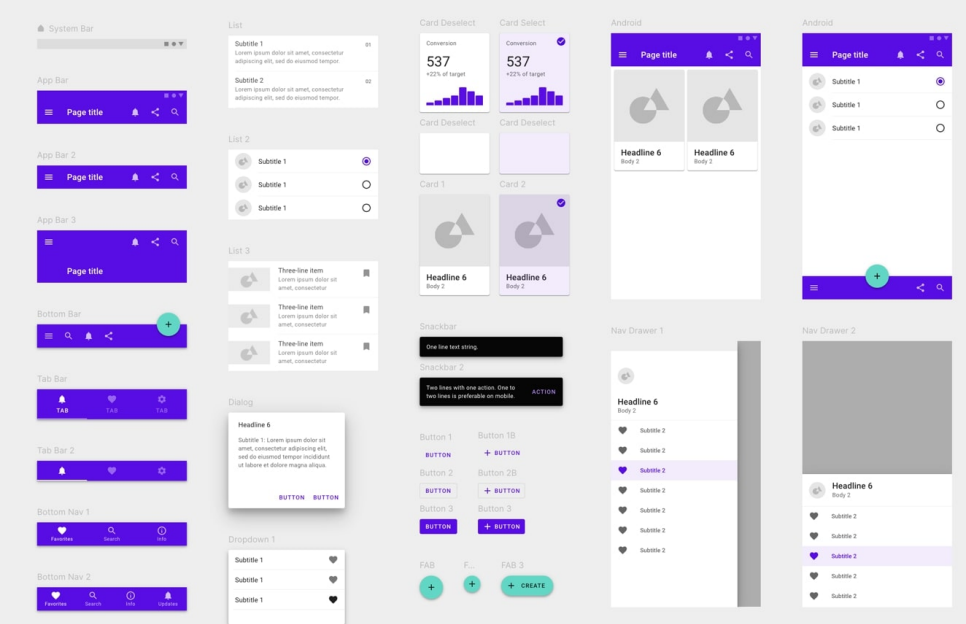
```
const { data } = useQuery<TestTemplate[]>('/test_templates')
```

3.5.9 Material UI

Material UI je UI knihovna, která implementuje Material design od Googlu.

Material je design systém představen firmou Google v roce 2014 pro podporu tvorby kvalitnějších aplikací pro web a mobilní vývoj. [13] Material UI implementuje řadu komponent z Material designu pro použití v React aplikacích. To znamená, že pokud vývojář vytváří novou aplikaci, nemusí vždy pro každou aplikaci vytvářet základní komponenty [3], ale použije právě komponenty z UI knihovny jako je například Material UI. Výhoda tohoto přístupu je, že vývojář používá otestované fuknění komopnenty. Nevýhoda je že pokud zákazník požaduje specifickou funkcionalitu, nemusí na to komponenty z UI knihovny stačit. Material UI také pomáhá s nastavením celkového stylu

[3] Jako tlačítka, laytou, containery nebo formulářové prvky.



Obrázek 3.7: Ukázka komponent z Material UI.

aplikace, proto umožňuje kompletní přenastavení stylu a barev jakékoliv své komponenty v podobě tzv. Theme [14]

3.5.10 Formik

Formik je knihovna pro zjednodušení práce s formuláři. Pomocí jedné komponenty se vytvoří celý formulář. Vývojář se potom vůbec nemusí starat o správu stavu formuláře a jen si nadefinuje, jak se mají jednotlivé pole validovat, a také co se má stát, když se formulář odešle. Pro Formik existuje také velice jednoduché napojení Material UI komponent[15], to umožňuje jednoduché používání Material UI komponent s knihovnou Formik.Formiku bohužel nemá možnost odesílat formulář stisknutím klávesy enter. Proto jsem tuto možnost potřeboval dodělat. Proto jsem pro tuhle funkci vytvořil speciální komponentu, která tuhle funkci zajišťuje.

Listing 3.4 Ukázka použití formuláře s knihovnou Formik

```
<Formik
  initialValues={initVals}
  onSubmit={handleSubmit}
>
  {(formikProps) => (
    <FormikForm formikProps={formikProps}>
    // CONTENT
    </FormikForm>
  )}
</Formik>
```

3.5.11 Styled components

Styled components je knihovna, která umožňuje psaní stylů přímo v react komponentách. Toto je technika, která se označuje jako CSS-in-JS. [16]

Tyto styly se dají definovat přímov souboru s react komponentou, takže se nemusí vytvářet další soubory speciálně pro styly, ale vše je v jednom. Je zde také výhoda, že pokud potřebujeme ostylovat pouze lokální proměnnou, nemusíme vymýšlet globální CSS třídu, ale stačí lokální proměnná, do které se stylu vloží.

Listing 3.5 Ukázka použití Styled components – úprava stylu komponenty Card z Material UI

```
const ModalCard = styled(Card)`
  position: relative;
  padding: 58px 48px;
  max-width: 500px;
  margin: 50px auto;
`;
```

3.5.12 Notistack

Notistack je knihovna pro zobrazení notifikací v Reactu. Notifikaci můžu zavolat z jakékoliv React komponenty pomocí custom hooku, který aplikace přidává.

Pro větší zjednodušení jsem si připravil svůj custom hook, kde jsem si předkonfiguroval jednotlivé typy notifikací, jako notifikace chyby nebo zdařilé operace.

Listing 3.6 Ukázka použití chybové notifikace při validaci.

```
1 const unique = isUnique(values.newOption)
2 if (!unique) {
3   errorNotification('Tato moznost uz existuje!')
4   setLoading(false)
5   resetForm()
6   return
7 }
```

3.5.13 Mui-rte

Mui-rte je jednoduchý textový editor pro Material UI založený na knihovně draft-js. [17] Tuto knihovnu jsem nejdříve použil jako NPN balíček, ale potřeboval jsem změnit zobrazení položek v toolbaru a taky větší možnost stylování celého editoru. Proto jsem do svého projektu zakomponoval zdrojový kód knihovny, potom jsem měl možnost si kód této knihovny upravit podle mých představ.

3.5.14 JSDoc

JSDoc je značkovací jazyk, kterým můžeme anotovat zdrojový kód v jazyce JavaScript.[18] Umožňuje vkládat dokumentaci přímo k funkcím a třídám v programu. Přidává také několik klíčových slov, pomocí kterých můžeme například definovat, jaké má funkce parametry nebo jaké vrací hodnoty.

Listing 3.7 Ukázka dokumentace funkce pomocí JSDoc a ošetření nevalidního vstupu pomocí Sentry.

```
1  /**
2   * Create initial values for formik form for student test
3   * @param questions question Groups to be stored in formik
        props
4   * @return {FormikValues} object for setting up initial
        values in Formik form.
5   */
6   export const setInitValues = (questions: QuestionGroup[]):
        FormikValues => {
7     return questions.reduce((acc, question) => {
8       handleSimpleQuestion(question, acc)
9       if (question.groupType === 'SIMPLE') {
10        handleSimpleQuestion(question, acc)
11      } else if (question.groupType === 'SPOJOVACKA') {
12        handleSpojovacka(question, acc)
13      } else {
14        console.warn('Unsupported question type at: ',
              question)
15        Sentry.captureMessage(`Unsupported question group
              type appeared with ID: ${question.id} with type:
              ${question.groupType}!`)
16      }
17      return acc
18    }, {} as FormikValues)
19  }
```

3.6 Další použité nástroje

3.6.1 Figma

Figma je grafický vektorový editor.[19] Dnes je Figma jeden z nejoblíbenějších nástrojů pro tvorbu UI. [20]. Z funkcí Figmy musím určitě zmínit pokročilou tvorbu prototypů aplikace. Dají se do určitě míry tvořit i animace různých přechodů a uživatelských interakcí. Tak se dá vytvořit opravdu funkční prototyp aplikace, která potom může sloužit jako základ pro testování mezi uživateli, pokud jim sedí rozvržení uživatelských prvků. Další možnosti, jak Figmu použít je pro tvorbu jednoduché vektorové grafiky. Například pro moji bakalářskou práci jsem zde tvořil několik ikon a logo aplikace. Nakonec musím zmínit, že všechny tyto věci jsou v základu zcela zdarma.

3.6.2 SonarLint

SonarLint je plugin do IDE [4] [23], který pomáhá se psaním kódu a vždy nabídne i vysvětlení a řešení jak problém opravit. Kontroluje možné fixy, zapomenuté a nevyužité importy, překlepy v názvech proměnných a metod. SonarLint funguje na pozadí, takže nabízí nápovědu v reálném čase, ale umožňuje i provést analýzu celého projektu najednou, a tak si získat přehled o stavu aplikace.

Webstorm[22] také umožňuje provést analýzu kódu před commitem do gitového repozitáře. To umožní ošetřit možné chyby, které by se mohly dostat do společného kódu.

3.6.3 Postman

Postman je platforma pro tvorbu a testování REST API.[24] Zjednodušuje způsob návrhu REST API a jejich testování na konkrétních endpointech.

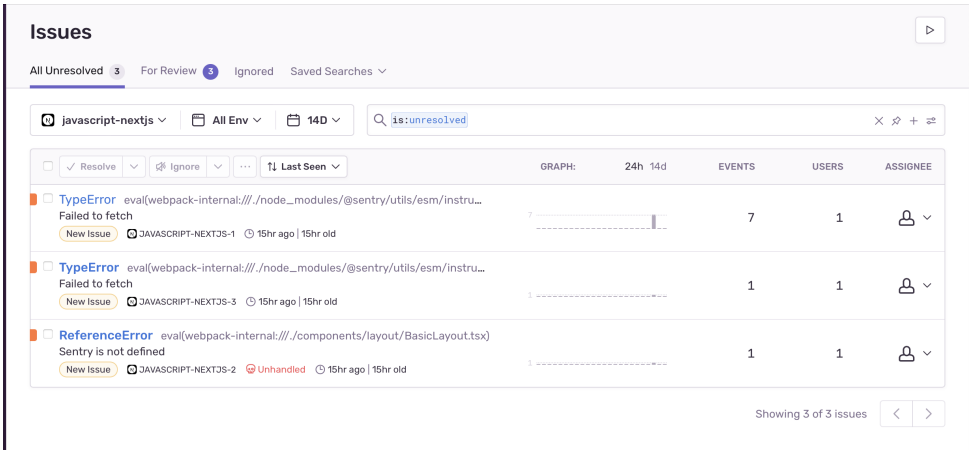
3.6.4 Gravatar

Gravatar je služba, která zjednodušuje používání uživatelských obrázků napříč internetem. [25] Uživatel si zde založí profil se svým e-mailem, kam si nahraje profilový obrázek. Služba poskytuje REST API dotaz pro získání profilového obrázku jen na základě e-mailu uživatele. Tuto službu používám pro zobrazení profilového obrázku v hlavičce uživatele.

3.6.5 Online edu

Při zadání bakalářské práce jsme dostali za úkol vycházet ze systému, který vytvořil Bc. Daniel Koten OnlineEdu. Zde jsme se rozhodli využít pouze části backendové aplikace a frontend jsem si vytvořil celý vlastní.

[4] Hlavně používám programy od firmy JetBrains IntelliJ IDEA[21] a WebStorm[22]



Obrázek 3.8: Ukázka výpisu chyb z platformy sentry.

3.6.6 Sentry

Sentry je platforma pro monitoring chyb, které se stanou v aplikaci. [26] To nám může dát lepší přehled, kde aplikace padá, a co je třeba vylepšit nebo opravit. Z mého pohledu je tato funkcionalita nezbytná, pokud je aplikace již nasazená a dostává reporty o nefunkčnosti určitých operací.

Poskytuje API, pomocí které může aplikace posílat chyby, které se v ní vyskytly. Sentry nativně podporuje velké množství programovacích jazyků včetně Javascriptu. [27]

Kapitola 4

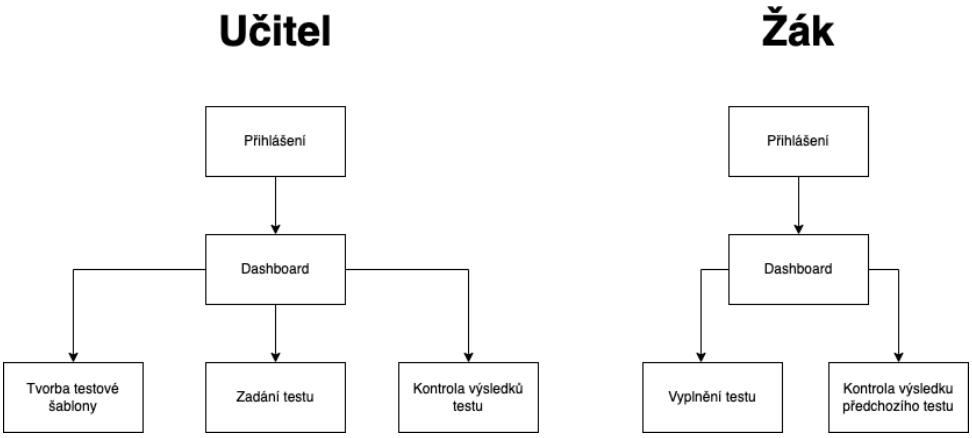
Architektura

4.1 UI návrh řešení

V této části se zaměřuji na návrh uživatelského rozhraní celé aplikace. Vycházím při tom z definovaných požadavků2 a z rozhovorů s učiteli3.4. Vytvořil jsem sitemapu celé aplikace. Dále se pak zaměřuji na návrh low fidelity a high fidelity wireframů. Tyto typy se rozlišují podle míry přesnosti k výslednému produktu (fidelity). Low fidelity – málo přesné, high fidelity – více přesné. Nakonec se věnuji tvorbě loga pro svou aplikaci. Hlavním cílem této kapitoly je přichystat správné podklady pro implementaci.

Začal jsem návrhem sitemapy4.1, kde jsem si vydefinoval jak budou části aplikace rozloženy. Zde počítám s tím, že aplikace bude mít jednu dashboard, která se bude měnit podle role uživatele a dále bude nabízet funkce, které jsou k dispozici pro danou roli.

Po definici sitemapy jsem začal s rozkreslováním jednotlivých stránek ze sitemapy. Zde jsem si dával pozor, aby komponenty, které vytvořím byly jednoduše implementovatelné pomocí Material UI.



Obrázek 4.1: Návrh sitemapy aplikace

4.1.1 Low fidelity prototyp

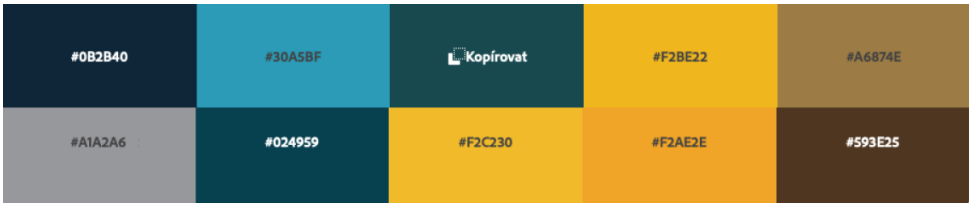
Low fidelity prototypy jsou takové, které nemusí stylem odpovídat výslednému produktu, jejich cílem je rychle vytvořit rozložení prvků v uživatelském rozhraní a otestovat, jestli je toto rozvržení vhodné. Většina těchto prototypů bývá černobílá.

V této fázi jsem se pokoušel navrhnout jednotlivé typy testových otázek, jak by se používaly v testu, a jak se tvořily v testové šabloně.

4.1.2 High fidelity prototyp

High fidelity prototypy už by měly být barevné a UI prvky by měly odpovídat výsledné podobě v aplikaci. Zde jsem už počítal s tím, že pro návrh aplikace použiji Material design [13] jako design systém celé aplikace. To mi zjednodušilo práci, jelikož základní prvky jsou už navrženy v tomto design systému a já jsem mohl tyto prvky jen použít.

Pro barevné schéma jsem se inspiroval z návrhů na platformě Dribbble, pro tento projekt jsem vytvořil kolekci projektů, které sloužily jako podklad při návrhu designu aplikace. [28].



Obrázek 4.2: Ukázka barevných schémat, ze kterých jsem vycházel při tvorbě svého barevného schématu.

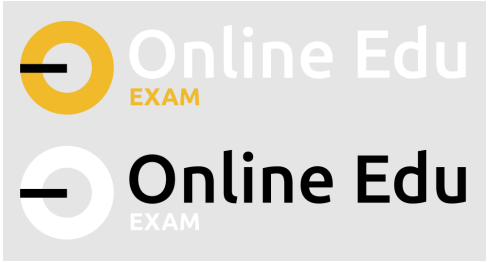
4.1.3 Tvorba loga

Pro aplikaci jsem chtěl vytvořit logo, které by bylo jednoduché a lehce zapamatovatelné.

Proces

Inspiraci jsem hledal v online generátorech pro loga. Jsou to stránky, kde si může uživatel pomocí pár otázek nechat vygenerovat logo přímo pro svůj projekt. Zde jsem si vytipoval pár log, které by se hodily k mému projektu. Potom jsem se snažil loga překreslit do Figmy a inspirovat se z nich při tvorbě vlastního. Jedno se mi hodně zalíbilo a začal jsem si hrát s typografií a barvami.

Potom jsem logo přidal do své aplikace a zjistil jsem, že je nápis v logu hrozně malý a hodně splývá s kolečkem v logu. Po této zkušenosti jsem začal experimentovat s různými tvary kruhu a nakonec jsem text a kruh rozdělil vedle sebe. Po dosazení loga do aplikace jsem ale zjistil, že v light modu splývá kolo s hlavičkou. Takže jsem musel ještě nastavit barvy tak, aby reagovaly na zvolený mode aplikace. [příklady log z generátoru]



Obrázek 4.3: Výsledná loga pro dark mode (nahoře) a light mode (dole)

Listing 4.1 Logo jako React komponenta, která reaguje na dark mode

```
export const Logo = () => {
  const theme = useTheme()
  const darkMode = theme.palette.mode === 'dark'
  const color = darkMode ? '#F0BA2A' : '#fff'
  const heading = darkMode ? '#fff' : '#000'
  return (
    <svg
      width="200" height="40" viewBox="0 0 756 162"
      fill="none"
      xmlns="http://www.w3.org/2000/svg"
    >
      <circle
        cx="75" cy="87" r="57.5"
        stroke={color} strokeWidth="35"
        />
      <rect
        y="81" width="75" height="14" fill="black"
        />
      <path
        d="..."
        fill={heading}
      />
      <path
        d="..."
        fill={color}
      />
    </svg>
  )
}
```

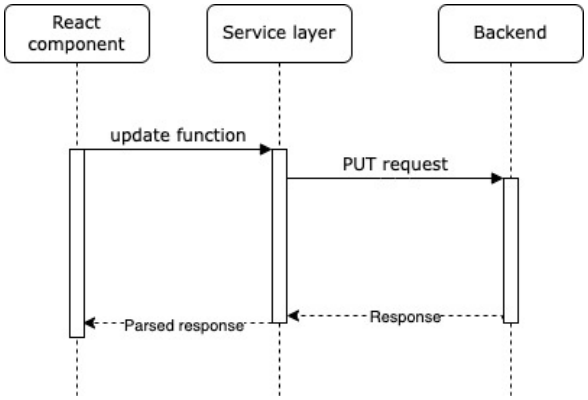
4.2 Architektura frontendové části

Při implementaci jsem se snažil držet zavedených postupů a design patternů pro frontendový vývoj.

Aplikace běží na frameworku Next.js, ten se stará o životní cyklus celé aplikace. Podle funkčnosti je rozdělena na jednotlivé stránky, které řeší jednotlivé funkcionality aplikace.

4.2.1 Service layer

Pro GET requesty používám knihovnu React Query [11], která poskytuje jednoduché deklarativní rozhraní pro fetch dat. 3.5.8. Pro úpravu dat React query používá mutace, které mi přišly zbytečně nepraktické, proto zde používám svůj service layer, který zapouzdřuje requesty pro aplikaci a vystavuje jednotné api.



Obrázek 4.7: Ukázka update requestu a komunikace s backendem.

4.2.2 Layout komponenta

Pro základní layout jsem si vytvořil Layout komponentu, kterou obaluji všechny stránky, zde přidávám například statické prvky typu aplikační lišta, s přihlášeným uživatelem a s notifikacemi. Ale potom se zde řeší také zabezpečení jednotlivých stránek.

4.2.3 Providers

Výchozí bod Next.js aplikace dovoluje obalit obsah celé aplikace tzv. providery. Provider má zpravidla knihovna, která poskytuje nějakou globální funkcionalitu (např. React Query, Next Auth, Material UI). Samozřejmě se dají vytvořit i vlastní providery, já například obaluji celou funkcionalitu providerem, který drží informace o uživateli.

4.2.4 Theme aplikace

Při návrhu celé aplikace jsem dost vycházel ze základních konceptů Material designu a konkrétně z knihovny Material UI. Proto aplikace jako taková nemá moc vlastních stylů, spíše jen upravuje komponenty Material UI. V Material UI existuje možnost vytvoření custom theme [14], kde se pomocí jednoho souboru může nastylovat celá aplikace. V této theme jsou pouze malé změny vůči výchozímu nastavení a to je změna barev a malé úpravy základních komponent.

Typografie

Typografie v celé aplikaci je řešena pomocí jedné typografické komponenty, která zapouzdřuje všechno chování textů. [29] Všechny styly se poté dají jednoduše měnit úpravou theme, kde se dají nadefinovat jiné velikosti nebo vlastní fonty.

Listing 4.2 Ukázka použití typografických komponent

```
1 <Typography variant={'h1'}>Heading</Typography>
2 <Typography variant={'h2'}>Second heading</Typography>
3 <Typography variant={'overline'}>Overline text</Typography>
4 <Typography variant={'body1'}>Classic body text</Typography>
```

Obrázek 4.8: Ukázka vyrenderovaných typografických komponent

Dark mode

Dark mode začíná být ve vývoji UI aplikací čím dál více populární mezi uživateli. [30] Hlavní výhodou dark modu je používání aplikace ve večerních

hodinách, kdy velké množství bílé barvy tahá oči uživatele, toto dark mode elegantně řeší. Další důležitou výhodou je potom šetření baterie, kdy display potřebuje více energie pro zobrazení čistě bílé než černé, kdy u některých typů displayů dokonce může pixely úplně vypnout. Podle studie provedenou skupinou XDA developers [1] zavedení dark modu může u AMOLED display snížit využití baterie až o 63 %. [31]

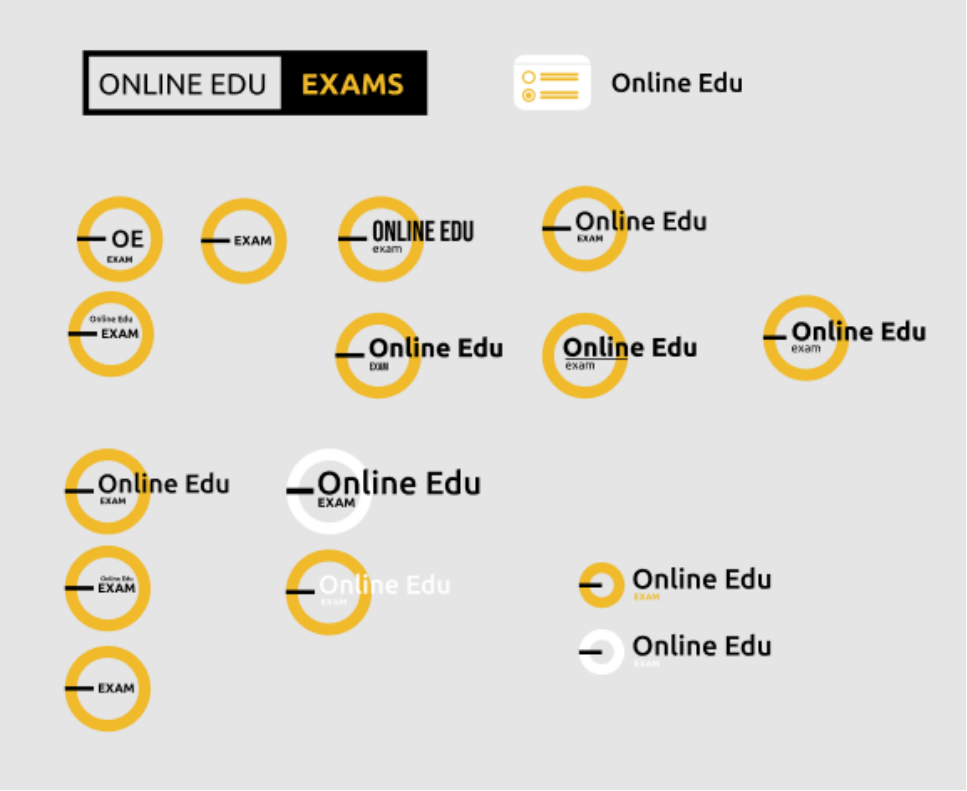
Implementace dark modu jsou různé. Aplikace standardně zobrazuje light mode a nabízí manuální přepínač do dark modu, nebo sama detekuje preferenci pro dark mode a zachová se podle této preference.

Já jsem zvolil způsob automatické detekce preference pro dark mode. Material UI nabízí pro tuto detekci funkci, která detekuje CSS media query. Zároveň jsem si nastavení theme rozdělil na základní úpravy a potom styly pro dark a light theme. Výsledné themes jsou potom dvě, ale obě mají společnou základní část, kde jsou definované základní prvky pro svou theme.

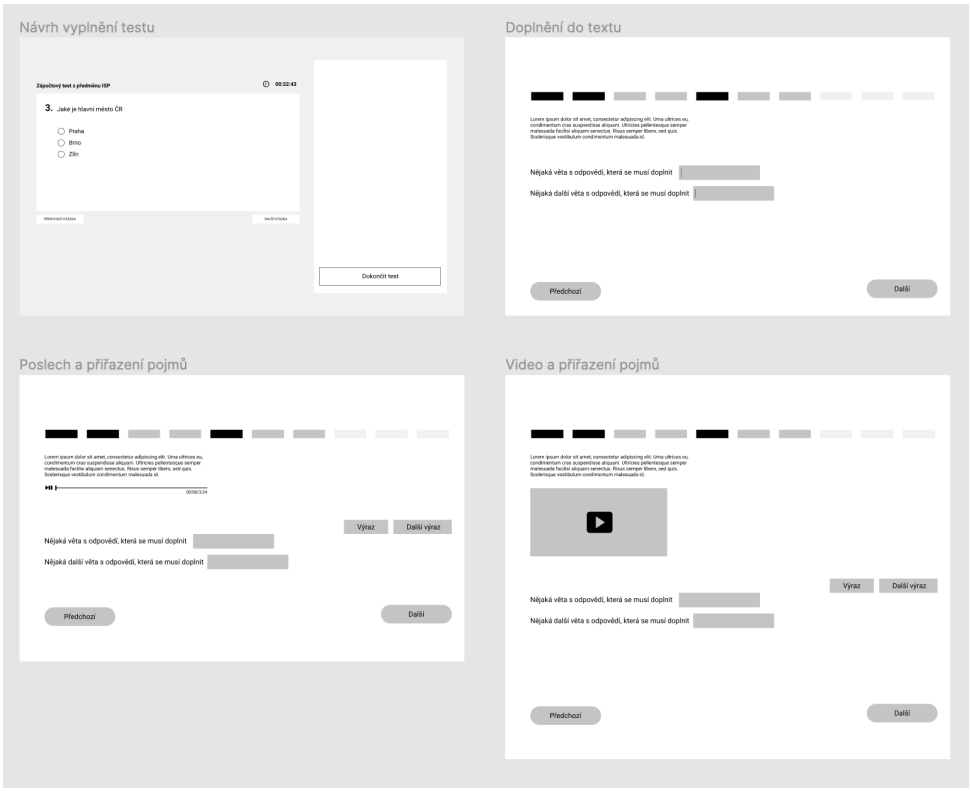
Listing 4.3 Volba theme podle preference dark mode

```
1 const prefersDarkMode = useMediaQuery('(prefers-color-scheme:
      dark)');
2 const themeSetup = prefersDarkMode ? darkTheme : lightTheme
3 const theme = createTheme(themeSetup)
```

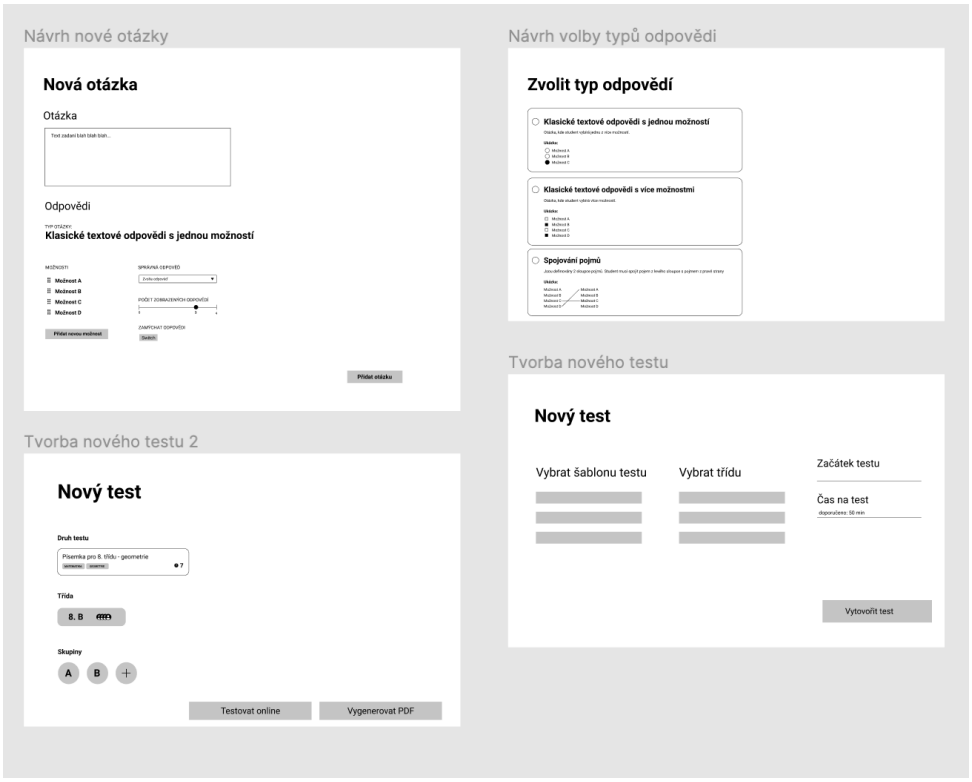
[1] https://www.xda-developers.com/google-wants-developers-to-add-dark-themes-to-save-battery-life/



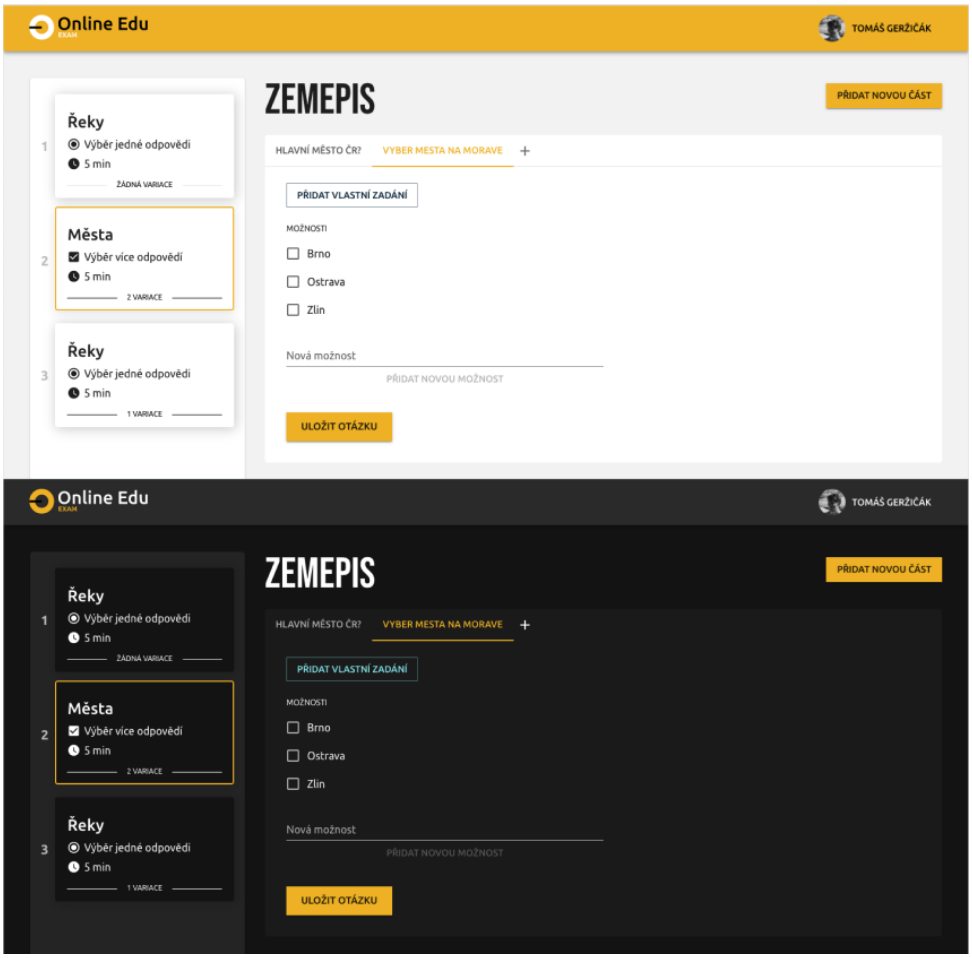
Obrázek 4.4: Proces návrhu loga.



Obrázek 4.5: Ukázka návhu low fidelity komponent pro studenta



Obrázek 4.6: Ukázka návhu low fidelity komponent pro učitele



Obrázek 4.9: Rozdíl UI mezi dark modem a klasickým

Kapitola 5

Části aplikace

Aplikaci jsem rozdělil podle funkčnosti na stránky, které mají každá své URL. Zde jsem využil funkci Next.js pro vytváření routes na základě adresářové struktury ve složce pages. [1]. Tato komponenta se stará o rozparsování URL a načtení dat z backendu, při načítání dat se zobrazuje příslušný skeleton loader pro lepší uživatelský zážitek a v případě chyby zobrazí chybovou hlášku. Hotová data se potom předávají do komponenty, která se stará o zobrazení obsahu stránky.

Pro umístění těchto komponent jsem vytvořil složku bundles. Každá stránka je jeden bundle a skládá se z hlavní komponenty, která se jmenuje stejně jako název bundlu a slouží jako základ pro celou stránku. Definuje rozložení prvků na stránce nebo stavy, ve kterých se stránka může nacházet. Například při vyplňování testu studentem se zde definují stavy, kdy se zobrazí text před testem, potom samotné vyplňování testu a nakonec zobrazení výsledků. V každém bundlu je ještě složka components, kde jsou konkrétní části bundlu a soubor utils.ts, kde můžou být pomocné funkce systému.

5.1 Tvorba testu

Tvorbu testu jsem chtěl udělat co nejjednodušší, ale zároveň umožnit i pokročilé úpravy. Při návrhu UI jsem se hlavně inspiroval aplikacemi pro tvorbu

[1] Dokumentace pro vytváření stránek v Next.js: `https://nextjs.org/docs/basic-features/pages`

prezentací jakou jsou Microsoft PowerPoint, Apple Keynote, ale i aplikací Kahoot[32] Tvorbu testu jsem si představil jako tvorbu prezentace se kterou jsou již učitelé seznámeni. Místo slidů jsou části testu a tam, kde je obsah slidu je možnost editace otázky (Přidávání variací, úprava možností, počtu bodů, atd.)

5.2 Vyplnění testu

Vyplnění testu je dostupné studentům, kteří jsou přiřazeni z testu. Tato obrazovka se drží již vžitých design patternů práce s dotazníkem nebo testem. Primárně musí být vidět otázka a odpovědi na ní. Dále je důležité, aby uživatel viděl, kde se v testu nachází a zda jsou vidět šipky pro rotování mezi otázkami.

Při návrhu této obrazovky jsem se inspiroval u programu Moodle, ale i například u Google Forms. S návrhem konktrétních prvků jsem si inspiroval příspěvky z platformy Dribbble, například boční zobrazení stavu testu. [2]

5.2.1 Zobrazení stavu vyplnění testu

U stavu testu bych zmínil propracování tlačítek pro zobrazení otázky. Toto tlačítko má různé stavy:

- Otázka nebyla zobrazena – černý text bez pozadí
- Otázka byla zobrazena – přidána obrysová čára
- Otázka je zodpovězena – přidáno pozadí. U komplexnějších otázek je pozadí vyplněno odspoda jen do poměru odpovězených otázek ku počtu všech otázek. (například pokud je vyplněna jedna otázka ze 3, je pozadí vyplněno pouze z 1/3)
- Otázka je nyní zobrazena – plné pozadí a tlačítko je posunuté o pár pixelů nahoru a přidán stín.

[2] `https://dribbble.com/shots/15334762-Online-Exam-System`

5.2.2 Ukládání odpovědí

Ze svých vlastních zkušeností i zkušeností mých spolužáků se systémem Moodle vím, že pokud test v tomto systému vyplňuje hodně lidí, systému se zpomalují reakce. To je nejvíce patrné při vyplňování testu a přecházení na další otázku. V systému Moodle se čeká na každou odpověď serveru při požadavku přejítí na další otázku. Tato odpověď může v případě, že systém používá hodně studentů zároveň, trvat i minutu, kde student zbytečně ztrácí čas čekáním na server, místo čtení další otázky.

Z této špatné zkušenosti jsem si vzal ponaučení a snažil jsem se svou aplikaci navrhnout tak, aby k čekání na server docházelo co nejméně a to hlavně při vyplňování testu studentem. Hlavní výhodou je, že zadání všech otázek se stáhne ještě před začátkem testu a uloží se v paměti klienta. Poté, když student prochází jednotlivými otázkami,aktivní otázka se už renderuje podle dat z cache a klient nemusí čekat na server s žádostí dat o další otázce.

Jako další optimalizaci tohoto problému jsem implementoval asynchronní odesílání odpovědí na otázky na server. Funguje to tak, že student vyplní otázku a při přechodu na další se vyšle asynchronní odeslání odpovědi, které probíhá na pozadí a student může bez jakéhokoliv čekání přejít k další otázce.

5.2.3 Další funkce

Při návrhu této stránky bylo nutné myslet na to, pokud uživatel nečekaně přeruší test. To znamená, že může omylem zavřít prohlížeč nebo se může vybít baterie v počítači. Proto bylo nutné implementovat metody, aby se uživatel při opětovném přihlášení mohl do testu ihned vrátit. Proto jsem implementoval možnost, že se samozřejmě při opětovném načtení stránky načtou všechny odpovědi uživatele, ale také si stránka pamatuje, na které otázce se uživatel nacházel a přímo mu umožní pokračovat tam, kde skončil. Pro tuto funkcionalitu jsem využil browser API localstorage[33], která umožní uložit data do lokální paměti počítače.

Listing 5.1 Inicializace testu s přihlédnutím na část uloženou v localstorage

```
1 const cachedActiveStep = localStorage.getItem(
      LOCAL_STORAGE_ACTIVE_STEP)
2 const [activeStep, setActiveStep] = useState(Number(
      cachedActiveStep) ?? 0)
```

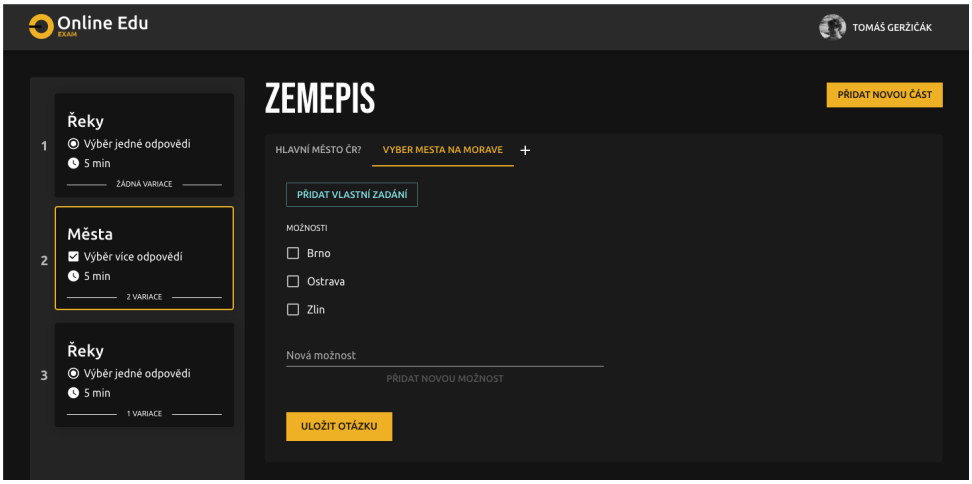
5.3 Učitelský přehled testu

Tento přehled je stránka, kde učitel vidí přehled odevzdání daného testu. Vidí zde, kolik studentů odevzdalo, kdy a kolik dostali bodů. Dále se může podívat přímo do odpovědí na testy jednotlivých studentů.

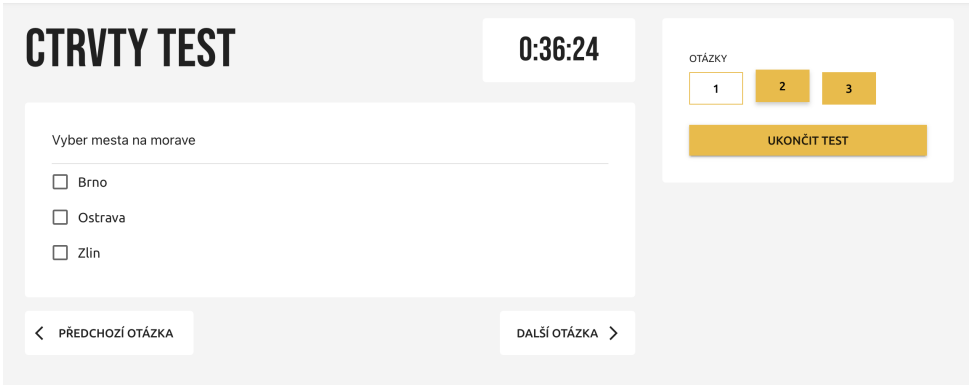
Při návrhu použití této stránky jsme mysleli na to, že učitel tuto stránku zapne a bude čekat, než studenti odevzdají své testy. Tady jsme ale nechtěli, aby musel učitel obnovovat stránku manuálně, aby se dozvěděl aktuální stav odevzdání. Proto jsme zde přidali spojení přes WebSocket, kdy server posílá klientovi zprávu a ten na základě přijetí zprávy znovu načte data pro konkrétní test.

5.4 Úvodní stránka

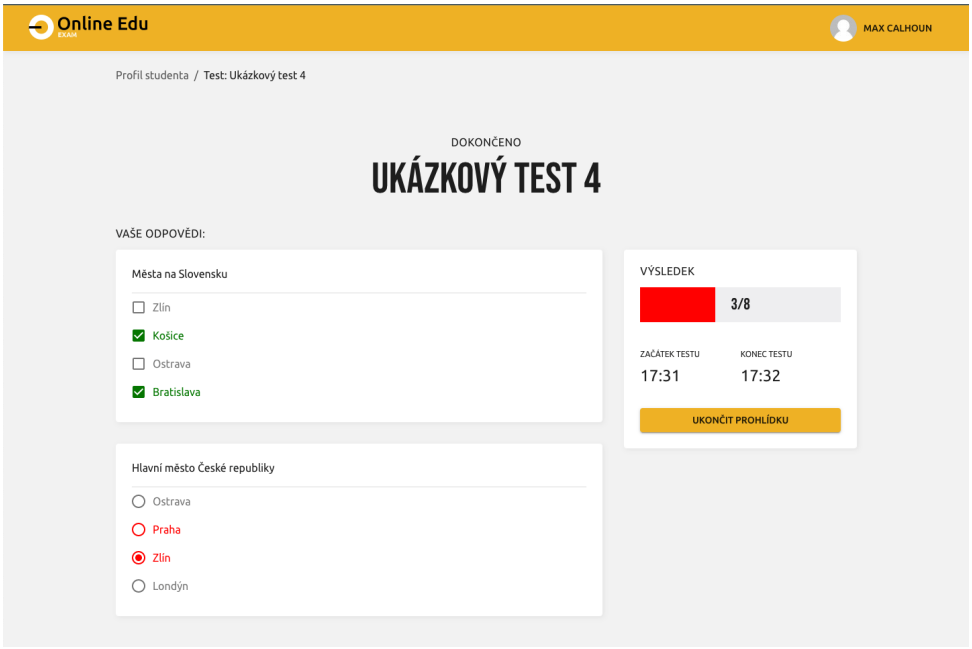
Pro každou aplikaci je důležitá stránka, která představuje celý produkt a všechny jeho funkce. Svou stránku jsem chtěl udělat co nejjednodušeji, aby byla v podobném stylu jako celá aplikace a aby představila základní funkce aplikace. V úvodu jsem vsadil na velký nadpis a krátký úderný text. Potom velký obrázek, který má uživatele nalákat na funkce aplikace. Pod tento úvod jsem už jen dal tři kartičky, které blíže popisují funkce aplikace.



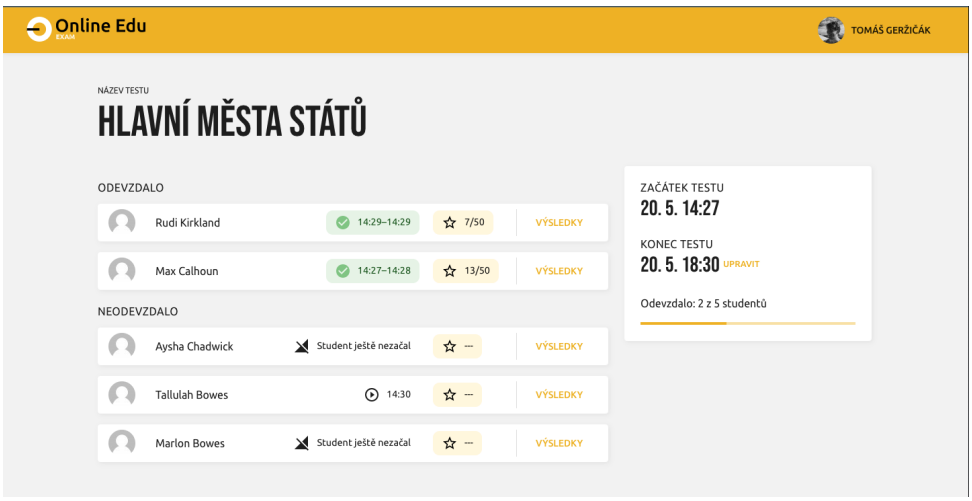
Obrázek 5.1: Ukázka tvorby testové šablony.



Obrázek 5.2: Ukázka uživatelského testu



Obrázek 5.3: Ukázka vyhodnocení testu studenta



Obrázek 5.4: Ukázka přehledu testu pro učitele.

Kapitola 6

Další zkušenosti z implementace

6.1 První implementace pomocí Vue.js

Jako první jsem chtěl napsat celou aplikaci pomocí knihovny Vue.js a frameworku Nuxt.js, se kterým jsem byl zvyklý pracovat na menších projektech. Framework Vue je v dnešní době na přerodu z verze 2 na verzi 3, která zlepšuje dost věcí, ale hlavně je celá napsaná v Typescriptu, který je použitelnější pro větší projekty, protože zavádí typy do jinak volně typovaného Javascriptu. Bohužel tato verze není úplně dodělaná a zásadní knihovny, které jsou potřebné pro vývoj moderní aplikace ještě plně nepřešly na novou verzi frameworku, například jde o UI knihovnu Vuetify [34] nebo framework Nuxt, který je v době psaní tohoto textu stále ve vývojové fázi [1] a není úplně použitelný pro stabilní vývoj.

Z tohoto důvodu jsem zvolil Vue verzi 2 s frameworkem Nuxt.js a protože jsem chtěl využít Typescriptu, je možné s Nuxtem taky použít Typescript, ale není to úplně nelepší řešení. Taky pro správu stavu jsem použil trochu těžkopádnější řešení v podobě Vuexu. Knihovna Vuex je implementace flux patternu od firmy Facebook, bohužel se mi s tímto design patternem dost špatně pracuje, není to úplně nejčistší postup. Na druhou stranu jsem tento framework už znal a celkově se mi s ním dost dobře pracovalo, proto jsem se na začátku rozhodl pro toto řešení.

[1] https://v3.nuxtjs.org/community/roadmap

6.1.1 Integrace Typescriptu

Jelikož vím, že tento projekt bude větší a bude zde velké množství datových typů, nechtěl jsem používat klasický Javascript, ale využít možnosti Typescriptu. Protože Javascript je dynamicky typovaný jazyk a jednoduše nepodporuje tvorbu typů, věděl jsem, že by se tím o dost ztížil pozdější vývoj. Proto jsem chtěl, aby můj projekt byl celý v typescriptu.

Bohužel Vue ve verzi 2 nepodporuje nativně Typescript a jeho integrace není úplně správná pro produkční aplikaci. Na druhou stranu se mi líbila práce s knihovou Vue property decoreator [2], která umožňuje vytvářet class komponenty ve Vue s Typescriptem. I když se mi tato knihovna líbila, dlouhodobější práce s ní nebyla tak pohodlná, jak jsem si myslel.

6.1.2 Problémy se správou dat

Pro správu dat jsem zvolil knihovnu Vuex[35]. Vuex je knihovna pro správu globálního stavu aplikace, implementuje Flux pattern.

Práce s Vuexem ovšem není úplně ideální, pro jednoduché věci se musí napsat hodně kódu a potom následná správa není úplně bezpracná. Navíc Vuex poskytuje pouze způsob, jak se data ukládají, ale jejich logiku stahování a cashování si musí uživatel napsat sám. Dokonce i nastavení není úplně jednoduché a nebylo mi úplně příjemné, že tak základní věc se musí tolik konfigurovat.

I tato nepříjemnost mi pomohla se rozhodnout tuto technologii opustit a radši vsadit na jistotu Reactu.

6.1.3 Přechod na React

S Vue.js jsem měl doposud jenom dobré zkušenosti, bohužel pro tento projekt a nedokončenost 3. verze frameworku Nuxt.js mě přinutili ke změně technologie na knihovnu React s frameworkem Next.js. Když to můžu hodnotit zpětně, jsem se svým rozhodnutím spokojený.

[2] https://github.com/kaorun343/vue-property-decorator

6.2 Vývoj komunikace s backendem

Jelikož backend nevyvíjím já, ale kolega Jan Krčil, bylo nutné stanovit si zásady pro předávání dat a vývoji REST API. Pro zjednodušení a urychlení vývoje jsem musel vyřešit, jak vyvíjet frontendovou část bez potřebných endpointů. Nejdříve jsem data neřešil a vše vyvíjel bez dotazů na jakékoliv endpointy. V jisté fázi vývoje jsem ale potřeboval připojit svou aplikaci k reálnému serveru, abych mohl uzavřít vývoj určitých komponent. Tohle jsem v průběhu vývoje řešil různými způsoby.

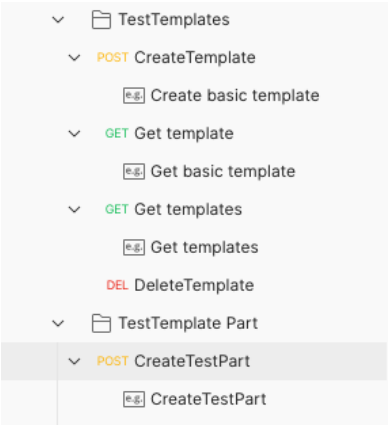
API routes v NEXT.js

Zde jsem využil funkci NEXT.js která umožňuje vytvářet vlastní endpointy. Je to užitečné pokud chci například maskovat dotazy na backend nebo pokud náš backend nespecifikuje CORS [link] headery, potom můžeme dotazovat backend s pomocí NEXT.js api.

Pomocí tohoto API jsem si namockoval endpointy, které ještě v té době nebyly vytvořené. Pouze jsem definoval cestu resource a potom vracel namockovaná data. Poté jsem ve své aplikaci mohl implemotovat dotazy na klasické endpointy.

Listing 6.1 Ukázka použití Next.js api routes

```
// stranka: pages/api/test_templates/[templateId]/index.ts
// cesta k souboru pres: /api/test_templates/1
export default function handler(
    req: NextApiRequest,
    res: NextApiResponse<TestTemplate>
  ) {
    res.status(200).json(
      {
        id: 2,
        name: "Test template preview",
        parts: [{
                id:1,
                name: "First part",
                duration: 10,
                questions: questionsWithAnswers}]
      }
    )
  }
```



Obrázek 6.1: Ukázka vytvořených endpointů v Postmanu se svými příklady

Postman mock server

S nárůstem funkcionality se začala celá složka s mockovanými api routes zvětšovat a s úpravami struktury posílaných objektů začalo tohle řešení spíše zpomalovat a používání reálného backendu nebylo ještě tak stabilní, abych se na něj mohl spolehnout při každé situaci, kdy jsem potřeboval. Proto jsem musel vymyslet efektivnější variantu mockování dat, se kterými bude má aplikace pracovat.

Jako nejlepší způsob se jevila platforma Postman[24]. Tuto jsme už od začátku používali k testování a dokumentaci api route pro vývoj backendu. V Postmanu je možnost si vytvořit svůj Mock server [36], který má svoji veřejnou URL a kopíruje api routy našeho backendu. Nastavuje se velice jednoduše, pouze se pro celou kolekci api route vytvoří mock server a u každé routy, kterou chci přidat se dají nadefinovat data, která mock server odešle pomocí definice příkladů pro každou cestu. Toto Postman usnadňuje tak, že pokud zavoláme tento endpoint z reálného backendu, můžeme výsledek hned uložit do examplu a ten se hned propíše do mock serveru, na který se poté připojuje frontendová aplikace.

Tento způsob mi vyhovoval v tom, že jsem si takhle mohl vyvíjet bez potřeby lokálního rozjetí backendové aplikace. Nebo pokud byla v aplikaci chyba při volání specifického endpointu. Mohl jsem tak pokračovat ve vývoji aplikace, jen jsem vytvořil request, kde jsem popsal chybu, pro kolegu Jana Krčila, aby chybu mohl odstranit.

Integrace reálného API

Jakmile byla větší část API vytvořena, byl čas na integraci reálného backendu. Zde jsem narazil na problém, kdy některá část endpointů byla už vytvořena, ale některé ještě chyběly nebo obsahovaly chyby. Proto jsem se musel adaptovat na tuto situaci. Rozhodl jsem se použít již existující endpointy ve službě Postman a taky již funkční endpointy.

Ve funkci, která globálně pomáhá při komunikaci frontendové aplikace se serverem jsem implementoval mechanizmus přechodu na backendovou API. Při tomto přechodu funguje takhle: dotaz z frontendu zpracuje a pokusí se odeslat požadavek na server, pokud tento požadavek skončí chybou, stáhne funkce data s mockovaných dat z Postmana. Pro jistotu a další vývoj zobrazí v konzoli varování o chybném endpointu.

Listing 6.2 Funkce pro správu requestů na server

```
1  export const request = async <Body>(
2      queryKey: string,
3      method: RequestMethod,
4      body: any = null,
5      headers: HeaderType[] | null = null):Promise<Body> => {
6      try {
7        const baseUrl = 'http://localhost:8080/rest'
8        const url = baseUrl + queryKey
9        return await myFetch(url, method, body, headers) as
             Body
10     } catch (e) {
11       console.warn(`This endpoint is not implemented: \n ${
             url}`)
12       const testBaseUrl = 'https://5111cf20-bc9d-45ff-8712-
             cb935c41e59b.mock.pstmn.io'
13       return await myFetch(testBaseUrl + queryKey, method,
             body) as Body
14     }
15   }
```

6.3 Spolupráce s vedoucím práce

Pro správný chod projektu jsme si stanovili s vedoucím práce standupy každý týden v pátek ráno, kdy jsme si vymezili všichni čas, který si vyhradíme pro konzultace a prezentace své práce za uplynulý týden. Tyto standupy jsme, až na pár vyjímek, dodržovali a pravidelně se scházeli a konzultovali výstupy.

Bohužel někdy se nám stávalo, že vymezený čas nestačil a nestihli jsme dokončit rozebírané téma a další společný čas se hledal velmi těžce. Toto v některých situacích znamenalo zabrždění práce na aplikaci, protože jsme museli čekat, než budeme mít všichni čas problém vyřešit.

Kapitola 7

Problémy s prací

Podcenění analýzy

Jako první věc, co jsem si uvědomil, že jsem udělal špatně byla nedostatečná příprava a průzkum mezi učiteli, a také nedostatečné vydefinování všech požadavků, které učitelé potřebovali. Měl jsem věnovat více času rozkreslení všech stránek a funkcí celé aplikace. Návrhům stránek a základních prvků jsem se sice věnoval, ale řekl bych, že jsem nad tím mohl strávit více času. Proto se stalo, že jsem musel v pozdějších fázích vývoje některé části aplikace přepracovávat od začátku.

Málo rozhovorů s učiteli

Také si uvědomuji, že jsem měl oslovit více učitelů a přímo je nechat se podílet na vývoji aplikace. I proto jsem tento problém napsal do plánů do budoucna 8.1 a určitě ho chci zrealizovat.

Zdržení práce na backendu

Na začátku práce jsem předpokládal, že základní funkce backendu budou vyvinuty relativně rychle a potom na těchto základech budeme stavět. Bohužel

se nám dost často stávalo, že vnitřní navržená architektura backendu nebyla správně navržena, a proto se musela dost často přepisovat a navrhovat znova. Podle mého názoru to bylo zapříčiněno tím, že jsme se snažili navrhnout jednu strukturu, do které se budou ukládat všechny typy otázek a odpovědí. To zpětně beru jako chybu, protože nám toto řešení přineslo hodně problémů. I když se tento problém mé práce přímo netýkal, bohužel mě to ovlivnilo nepřímo zdržením vývoje backendu.

Jako řešení bych viděl každou otázku zpracovávat zvlášť a ukládat ji do vlastních tabulek a distribuovat každý typ přes vlastní endpoint, aby se nemusela ohýbat jednotná struktura.

Velká závislost na backendu

V pozdějších fázích vývoje, kdy jsem měl celou aplikaci napojenou na API backendu se dost často stávalo, že některé endpointy nefungovali a také se dost často měnila datová struktura jednotlivých endpointů. To znamenalo vždy předělávání množství komponent, které s touto strukturou pracují.

Řešení tohoto problému bylo definovat si přesné vstupní hodnoty pro své komponenty, a pokud by se nějak měnila API, tak zavést své mapování, které data upraví na jedno místě.

Nedostatek komunikace

Kvůli koronavirové pandemii jsme byli oba s kolegou zvyklí na home office jak z práce, tak ze školy. Proto jsme se moc na živo nescházeli a většinu věcí na bakalářskou práci jsme řešili online přes video hovory.

Bohužel tento postup vedl k velkému množství komunikačního šumu, který jsme neměli jak eliminovat. To vedlo ke špatnému pochopení některých návrhových a implementačních konceptů. I to je důvod, proč se aplikace musela tolikrát přepisovat. Jako pokus o zkvalitnění těchto podmínek jsme zkoušeli online pair programming [37]. To nám trochu pomohlo, ale i tak jsme nebyli moc produktivní.

Až na konci naší práce jsme zase přešli do prezenčních podmínek, kdy jsme uspořádali několik dnů, kdy jsme byli oba s kolegou ve společné místnosti

a celý den jsme programovali. Za tato sezení jsme stihli velkou část práce, i se nám lépe komunikovalo a byli jsme oba produktivnější. Závěrem bych chtěl dodat a zdůraznit, že jakkoliv dobře nám dnešní technologie umožňují komunikaci na dálku, vždy je lepší osobní kontakt.

Práce ve dvou lidech

Práci ve dvou na bakalářské práci nehodnotím úplně šťastně, jelikož zde byla řada faktorů, které nás oba dost zpomalovaly. Jako hlavní věc bych zařadil nedostatek komunikace 7 spolu s komunikací na dálku, na kterou jsme byli zvyklí ještě z online výuky. To nám ovšem vůbec nesvědčilo, a proto jsme ke konci vývoje přešli do plně prezenční formy.

Další byly problémy s gitem, když jsem přispíval do backendové aplikace. Můj nápad na vytvoření git flow[1] nevyšel a radši jsme se rozhodli pro commity přímo do master větve. Bohužel se často stávalo, že se nám pushem do git repozitáře přepsaly soubory. Nakonec jsme přišli na to, že git klient programu Intellij IDEA se snaží vždy udělat merge s lokální větší gitu, což dokáže v nějakých případech rozbít celou branch.

[1] https://www.gitkraken.com/learn/git/git-flow

Kapitola 8

Plány do budoucna

8.1 Testování mezi učiteli

Pro další vývoj aplikace je důležité mít větší přehled požadavků od učitelů. Proto bych chtěl oslovit více učitelů, se kterými bych prodiskutoval jejich tvorbu testů, a také by byli ochotni se podílet na vývoji mé aplikace.

8.2 Další možnosti v editoru testových šablon

V rámci své bakalářské práce jsem implementoval základní typy zadání a možností odpovědí při tvorbě testů. Při průzkumu mezi učiteli jsem ale zjistil, že by byl zájem o další, více specifické typy možností.

V následující kapitole shrnu další možné typy testových otázek a odpovědí, které jsme s učiteli vymysleli a částečně navrhli.

Programovací úloha

Myšlenka tohoto typu otázky je primárně inspirována testy z platformy Treehouse[4]. Otázky jsou zde klasické, ale i programovací, kdy je studentovi předložen určitý kód a on ho musí doplnit, nebo napsat úplně nový podle specifikace.

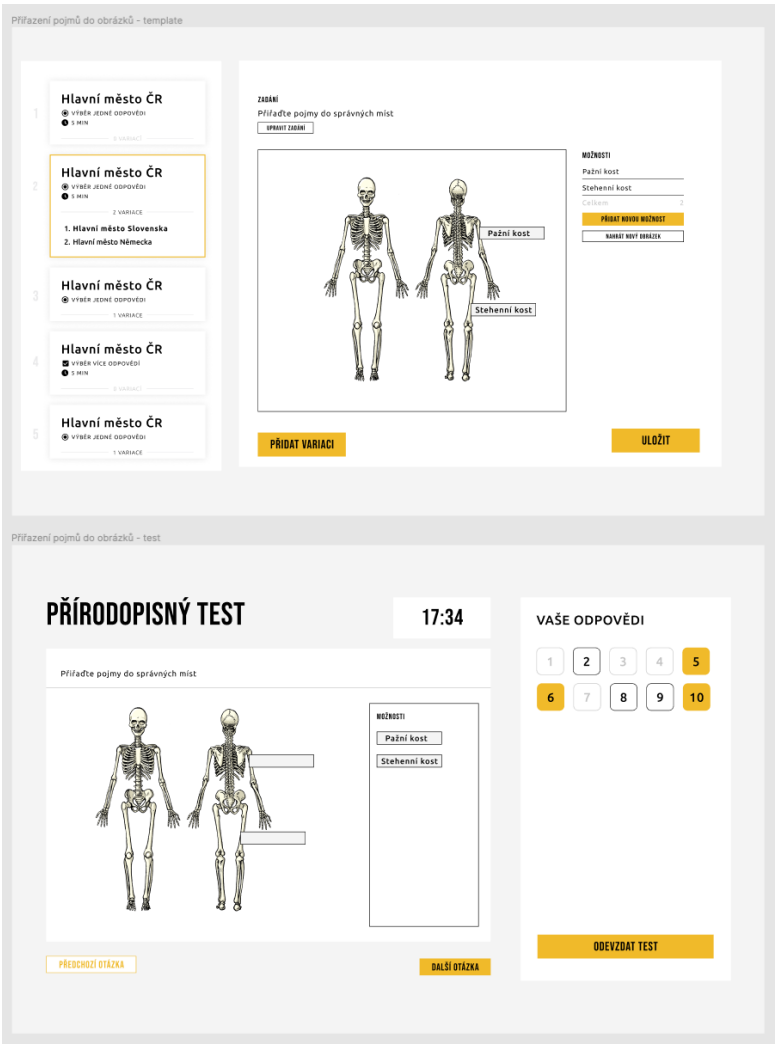
Moje představa o tomto typu úlohy je vlastně stejná, student by dostal již předem připravený kód, který by musel upravit. Nebo by dostal zadání, co přesně musí naprogramovat a jen prázdný editor. Pro jednoduché příklady by mohl učitel připravit svůj testovací program. Pro obtížnější případy by mohla aplikace odesílat kód do nějaké služby, která by kód analyzovala a vrátila odpověď. Zde jsem si vzal příklad ze školního systému Brute. Moje aplikace by jen poskytovala interface pro psaní kódu a vyhodnocení by zůstalo na vlastním serveru.

Jak už jsem zmiňoval, že by aplikace měla poskytovat prostředí pro psaní kódu, samozřejmě nechci vytvářet vlastní editor kódu, o to se například pokusili v Treehousu. Věřím, že vytvořit takový editor nebylo vůbec jednoduché, ovšem nedopadlo to úplně nejlépe a pořád jsem tam pozoroval nedostatky.

Interaktivní práce s obrázky

Učitelé přírodovědných předmětů často potřebují, aby studenti byli schopni popsat obrázek, například lidské tělo nebo slepou mapu. Proto potřebují mít možnost importovat obrázek a do obrázku vkládat popisky. A studenti musí být schopni do obrázku tyto popisky správně vložit.

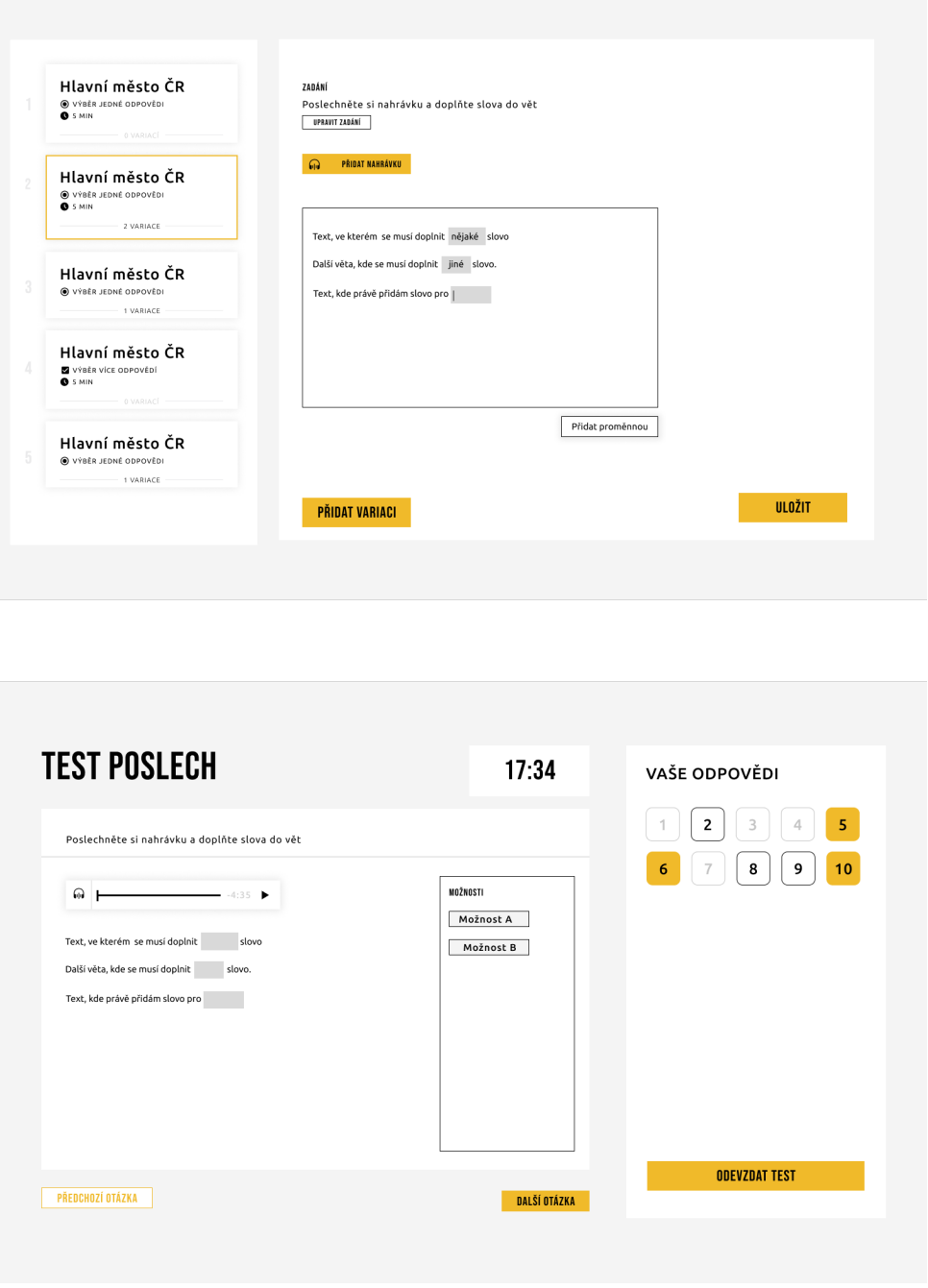
Dovolím si říct, že tento úkol každý učitel i student zná, a proto chci návrh vytvořit co nejjednodušeji. Při tvorbě testu učitel naimportuje obrázek a poté může na obrázek přímo vkládat textové popisky. Z těchto popisků se poté vygeneruje pole, kde student při vyplňování testu přetahuje dostupné popisky pomocí funkce drag and drop.



Obrázek 8.1: Návth možné podoby tvorby interaktivní práce s obrázky. Tvorba v šabloně nahoře a podoba v testu dole.

Poslechová cvičení

Tato cvičení budou využívat primárně učitelé jazyků, jde o klasické textové zadání s tím, že učitel přiloží zvukový soubor, který si žák přehraje a vyplní odpověď.V návrhu této stránky jsou použité odpovědi: Doplnění do textu.



Obrázek 8.2: Návth možné podoby tvorby interaktivní práce s obrázky. Tvorba v šabloně nahoře a podoba v testu dole.

Matematická zadání

Jak už jsem definoval v požadavcích, matematická zadání jsou typy zadání, kdy učitel může do textu přidat proměnné. Těmto proměnným nastaví učitel rozsah generovaných hodnot. Po specifikaci zadání a proměnných zadá učitel výpočet pomocí vzorce složeného ze všech proměnných. Systém se postará o vše ostatní. Každému studentovi vygeneruje zadání z náhodných hodnot v rozsahu proměnných, spočítá výsledek pomocí vzorce a možné odpovědi. (S jediným správným řešením).

8.3 Rozdělení aplikace do microservice

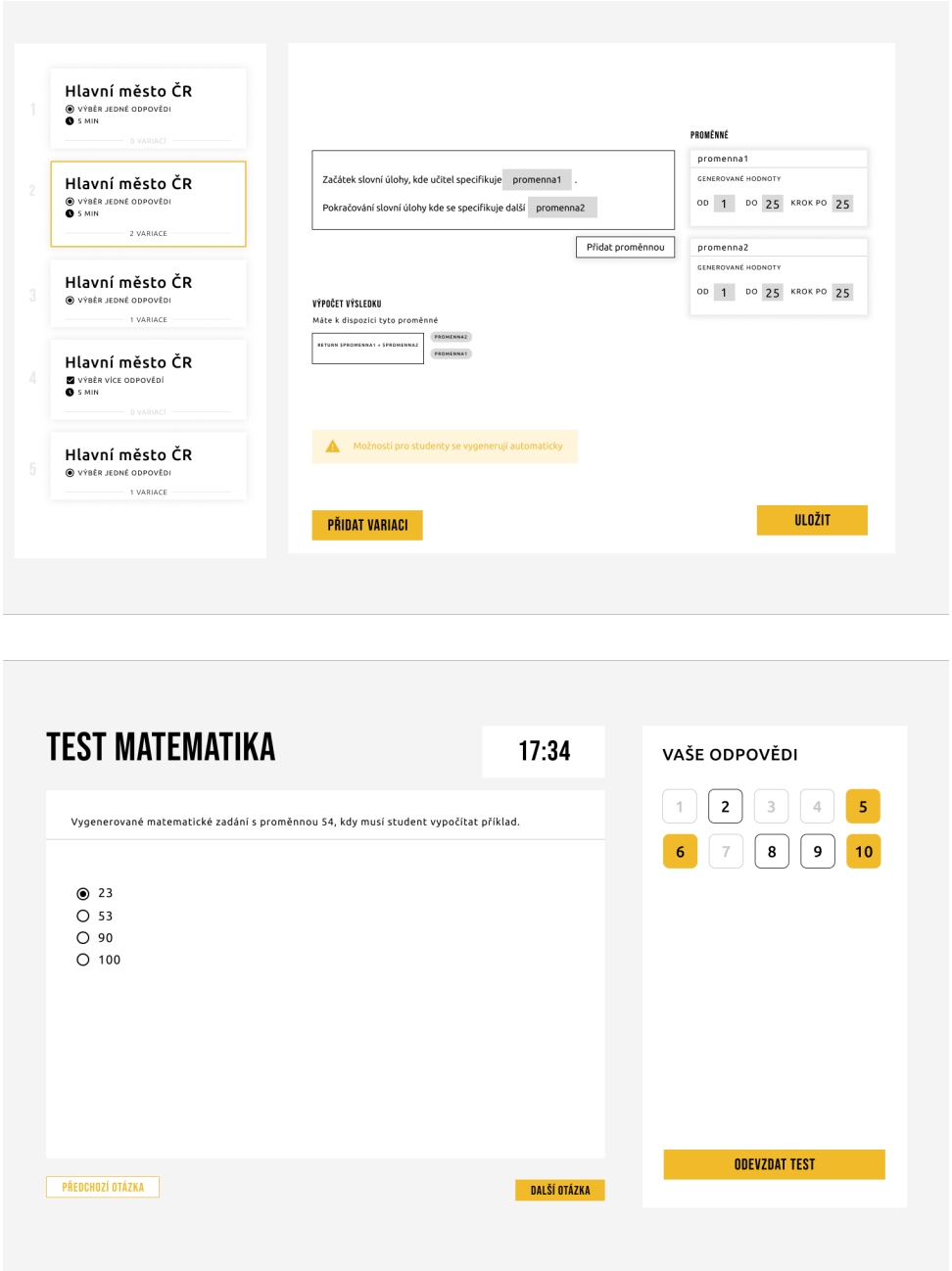
Je zřejmé, že pokud by aplikaci používalo velké množství lidí a začala se také rozšiřovat její funkcionalita, bude potřeba aplikaci rozdělit do více microservice. Rozdělení do více microservice nejel že zlepší aplikaci obsluhovat více požadavků, ale také zjednoduší vývoj, kdy každá microservice se bude vyvíjet zvlášť a bude mít vlastní zdrojový kód, tak vznikne více menších aplikací a ne jedna velká.

8.3.1 Implementace microservice pro sběr odpovědí z testů

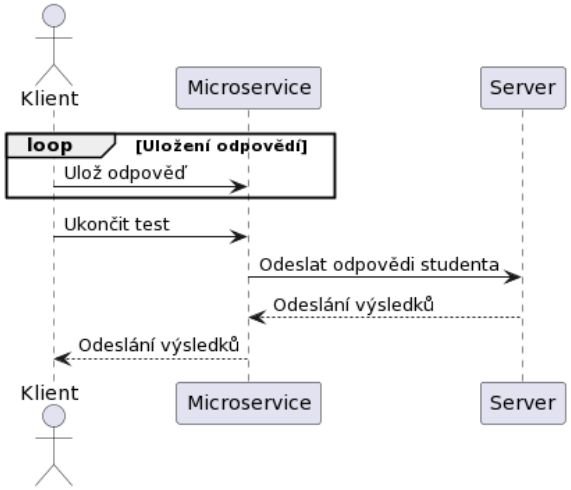
Jak jsem již zmiňoval v sekci o ukládání odpovědí, při vyšším počtu uživatelů aplikace dojde k tomu, že monolitický hlavní server nebude stíhat velké množství požadavků, které se na něj budou posílat při vyplňování testů. Pro tento typ funkcionality by se ale mohla vytvořit microservice, která by byla určena ke sběru data od studentů a po ukončení testu se všechny odpovědi odešlou na hlavní server, kde se zpracují.

Ilustrační příklad

Pokud například budou otázky, které budou v průměru trvat vypracovat jednu minutu, test bude psát třicet studentů a testů se ve zkouškovém období bude psát například 10 najednou a systém bude používat například 10 škol, je to 3000 requestů za minutu pouze pro sbírání odpovědí od studentů.



Obrázek 8.3: Návth možné podoby tvorby interaktivní práce s obrázky. Tvorba v šabloně nahoře a podoba v testu dole.



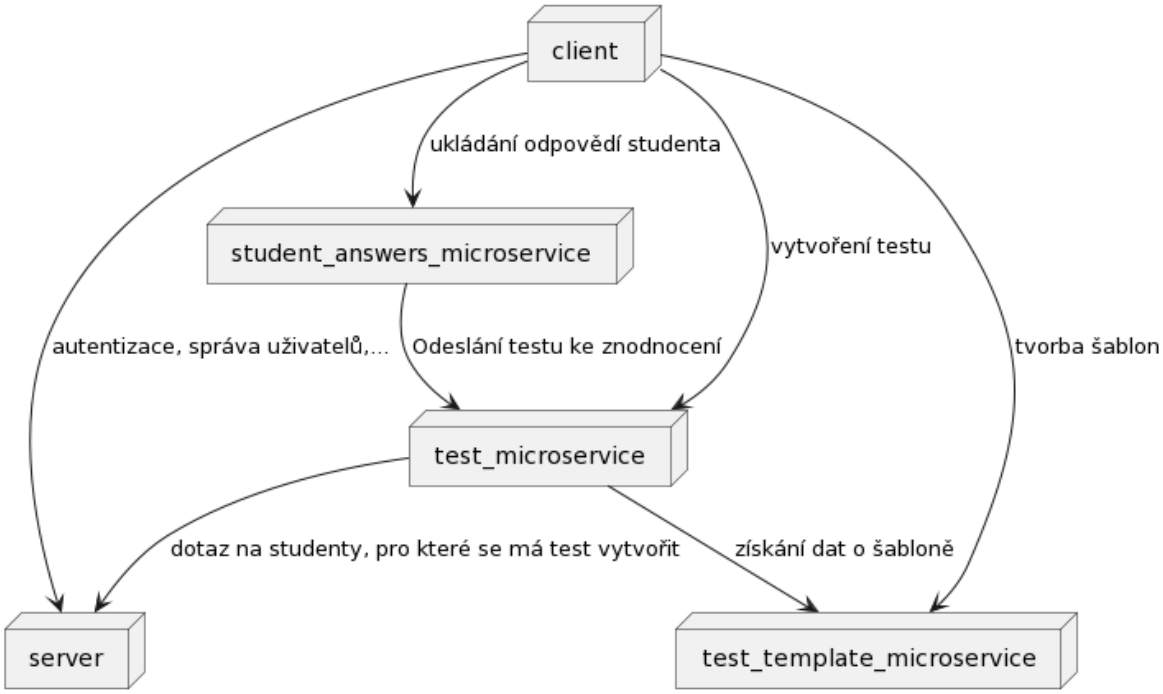
Obrázek 8.4: Sekvenční diagram funkce microservice pro sbírání odpovědí z testu.

Možnosti implementace

Funkce této microservice je poměrně jednoduchá, musí vlastně jen uchovávat odpovědi studenta. K implementaci této microservice bych vytvořil jednoduchou aplikaci pomocí Express.js a pro ukládání dat bych využil MongoDB, což je NoSQL databáze, kde by se jen ukládaly záznamy z odpovědí studenta. Následné parsovní by se dělo až na straně serveru.

8.3.2 Microservice pro tvorbu testových šablon

Pokud by aplikaci používalo opravdu hodně škol a také hodně učitelů, budou se také hodně tvořit testy, to může také dost zatěžovat hlavní server. Tvorba testových šablon se jeví jako další dobré rozdělení aplikace. Tato microservice může mít svoji vlastní databázi se všemi šablonami a kromě edpointů pro tvorbu testových šablon bude vystavovat pouze endpoint, který vrátí celou šablonu ve specifikovaném formátu, ze kterého se tvoří hlavní test. Tím, že bude mít tato microservice oddělenou databázi, zmírní se taky zatížení hlavní databáze.



Obrázek 8.5: Koncepční návrh systému pomocí microservice architektury.

8.3.3 Microservice pro správu testů

Jako poslední rozdělení aplikace vidím odtržení správy a vyhodnocování testů do své vlastní microservice. Tato microservice by tak měla svoji menší databázi se všemi testy a výsledky těchto testů. Měla by na starosti generování testů (s tím spojenou kominukaci s microservice pro tvorbu testových šablon), taky poskytování konkretních testů pro žáka a také následné vyhodnocení celého testu (komunikaci s microservice pro sběr odpovědí z testu).

8.4 Implementace tisknutelné verze testu

Při generování testu bude možnost test vygenerovat do PDF pro prezenční výuku. Při volbě této možnosti se vygenerují testy pro každého žáka v před připravené šabloně a se jménem konkrétního žáka. To by poté mělo pomoct učitelům v organizaci.

8.5 Statistika výsledků

Z průzkumů mezi učiteli jsem zjistil, že by se jim hodilo mít k dispozici analýzu testů a odevzdání. U každého testu by byla analýza například při kterých otázkách se nejvíce chybovalo nebo graf v přehledem, kdy kolik lidí odevzdalo.

Jako další plán by byl implementoval strojové učení, které by se pokoušelo analyzovat, jestli někdo při testu nepodváděl a procházel by odpovědi. Například by mohla hlídat situace, kdy student posílá odpovědi na otázky každých 20 sekund. To by mohlo znamenat, že odpovědi pouze opisuje.

Kapitola 9

Závěr

Cílem práce bylo analyzovat problémy při tvorbě testovacích formulářů při online výuce. Poté navrhnout řešení tohoto problému a implementovat jednoduché části řešení.

Při analýze jsem vycházel ze svých zkušeností a z rozhovorů s učiteli. Analyzoval jsem proces tvorby testů a učebních materiálů a navrhl jsem řešení pro jejich digitalizaci a usnadnění práce učitele.

Při návrhu datových struktur celé aplikace jsme narazili na problém velké variability možných typů otázek a na problém s persistencí těchto dat. Zde jsme museli vymyslet takovou strukturu dat, aby byla připravena i na budoucí rozšiřování o nové otázky. Nakonec se vše vyřešilo a mohli jsme přistoupit k implementační fázi.

Podařilo se implementovat základní funkcionality aplikace, které byly v zadání.

Učitel může vytvářet testové šablony pomocí základních typů otázek. Tyto otázky jsou: výběr jedné odpovědi, výběr více odpovědí, spojování výrazů a doplňování slov do textu. Pak může učitel podle této šablony vygenerovat unikátní test pro každého žáka, aby se zamezilo opisování. Žák tento test vyplní, zobrazí se mu výsledek, tyto výsledky poté vidí i učitel. Z hlediska UI jsem aplikaci navrhl tak, aby působila co nejpřívětivěji a mohli ji používat všichni učitelé bez rozdílu věku a zkušenosti práce s PC.

Aplikace má ale před ostrým nasazením do provozu ještě dlouhou cestu. Je potřeba vytvořit více typů otázek pro tvorbu testů, aby aplikaci mohla využívat většina učitelů. Dále chci provést větší průzkum mezi učiteli, nechat je aplikaci vyzkoušet a s jejich pomocí aplikaci vylepšovat a dodělat tak, aby jim pomohla zefektivnit jejich práci.

Příloha A

Seznam zkratek

API Application Programming Interface

CMS Content management system

CRUD Create, Read, Update, Delete

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protoco

HTTPS Hypertext Transfer Protocol Secure

IDE Integrated Development Environment

JSON JavaScript Object Notation

JSX JavaScript Syntax Extension

REST Representational State Transfer

UI User interface

UML Unified Modeling Language

wysiwyg What you see is what you get

Příloha B

Literatura

[1] Microsoft: *Vytvoření kvízu v Microsoft Forms*. https://support.microsoft.com/cs-cz/office/vytvořen%C3%AD-kv%C3%ADzu-v-microsoft-forms-a082a018-24a1-48c1-b176-4b3616cdc83d. – (cit. 18. 5. 2022)

[2] Moodle: *Building Quiz*. https://docs.moodle.org/27/en/Building_Quiz. – (cit. 12. 5. 2022)

[3] English, Help for: *Angličtina na internetu zdarma!* https://www.helpforenglish.cz. – (cit. 19. 5. 2022)

[4] Treehouse: *Best online coding classes for at-home learning*. https://teamtreehouse.com. – (cit. 19. 5. 2022)

[5] Facebook: *React*. https://github.com/facebook/react. – (cit. 15. 5. 2022)

[6] Pragimtech: *Class Components in React*. https://www.pragimtech.com/blog/reactjs/introduction-to-class-components-in-react/. – (cit. 12. 5. 2022)

[7] Next: *Next.js by Vercel – The React Framework*. https://nextjs.org

[8] NPM: *package.json | NPM Docs*. https://docs.npmjs.com/cli/v8/configuring-npm/package-json. – (cit. 17. 5. 2022)

[9] Webpack: *Webpack: Concepts*. https://webpack.js.org/concepts/. – (cit. 17. 5. 2022)

[10] ESLint: *ESLint - Pluggable JavaScript linter*. https://eslint.org. – (cit. 15. 5. 2022)

[11] Linsley, Tanner: *React Query – Hooks for fetching, caching and updating asynchronous data in React*. https://react-query.tanstack.com/, . – (cit. 15. 5. 2022)

[12] Linsley, Tanner: *React Query – Window Focus Refetching*. https://react-query.tanstack.com/guides/window-focus-refetching. – (cit. 15. 5. 2022)

[13] Google: *Material Design*. https://material.io/design. – (cit. 15. 5. 2022)

[14] UI, Material: *Theming – Material UI*. https://mui.com/material-ui/customization/theming/#main-content

[15] Stackworx: *Formik MUI*. https://github.com/stackworx/formik-mui. – (cit. 12. 5. 2022)

[16] Pfeiffer, Andrei: *A Thorough Analysis of CSS-in-JS*. https://css-tricks.com/a-thorough-analysis-of-css-in-js/. – (cit. 12. 5. 2022)

[17] Lopez, Erik: *Mui rte*. https://github.com/niuware/mui-rte

[18] JSDoc: *Getting Started with JSDoc 3*. https://jsdoc.app/about-getting-started.html. – (cit. 18. 5. 2022)

[19] Figma: *Figma: the collaborative interface design tool*. https://www.figma.com. Version: 2022. – (cit. 15. 5. 2022)

[20] Taylor Palmer, Jordan B.: *2021 Design Tools Survey | UX Tools*. https://uxtools.co/survey-2021/#ui-design. – (cit. 12. 5. 2022)

[21] JetBrains: *IntelliJ IDEA: The Capable & Ergonomic Java IDE by JetBrains*. https://www.jetbrains.com/idea/. – (cit. 12. 5. 2022)

[22] JetBrains: *WebStorm: The Smartest JavaScript IDE, by JetBrains*. https://www.jetbrains.com/webstorm/. – (cit. 12. 5. 2022)

[23] sonarlint: *Clean Code starts in your IDE*. https://www.sonarlint.org. – (cit. 18. 5. 2022)

[24] Postman: *Postman*. https://www.postman.com/

[25] Gravatar: *Gravatar - Celosvětově uznávané Avatary*. https://cs.gravatar.com. – (cit. 12. 5. 2022)

[26] Sentry: *Sentry*. https://sentry.io/. – (cit. 17. 5. 2022)

[27] Sentry: *Integrations | Sentry*. https://sentry.io/integrations/. – (cit. 18. 5. 2022)

[28] Dribbble: *Online-exams by Tomáš Geržičák*. https://dribbble.com/gerzitom/collections/5395970-online-exams

[29] UI, Material: *Typography – Material UI*. https://mui.com/material-ui/customization/typography/

[30] Steiner, Thomas: *Let there be darkness! Maybe...*. https://medium.com/dev-channel/let-there-be-darkness-maybe-9facd9c3023d

[31] Jerelyn, Amanda: *Why dark mode web designs are gaining popularity*. https://www.searchenginewatch.com/2020/09/30/why-dark-mode-web-designs-are-gaining-popularity/. Version: 2020

[32] Kahoot: *Kahoot! | Learning games | Make learning awesome!* https://kahoot.com. – (cit. 12. 5. 2022)

[33] Obaseki, Nosa: *localStorage in JavaScript: A complete guide*. https://blog.logrocket.com/localstorage-javascript-complete-guide/. Version: 2020. – (cit. 16. 5. 2022)

[34] Vuetify: *Vuetify — A Material Design Framework for Vue.js*. https://vuetifyjs.com/en/. – (cit. 12. 5. 2022)

[35] Vuex: *What is Vuex? | Vuex*. https://vuex.vuejs.org. – (cit. 12. 5. 2022)

[36] Postman: *Setting up mock servers*. https://learning.postman.com/docs/designing-and-developing-your-api/mocking-data/setting-up-mock/

[37] Codementor: *Pair Programming: What, Why, and How*. https://www.codementor.io/pair-programming. – (cit. 17. 5. 2022)

[38] Tidwell, Jenifer: *Designing interfaces - patterns for effective interaction design (2. ed.)*. O'Reilly, 2011. – 1–547 S. – ISBN 978–1–449–37970–4

[39] Žára, Ondřej: *JavaScript: Programátorské techniky a webové technologie*. Computer Press, 2015. – 1–184 S. – ISBN 978–80–251–4573–9

[40] Crockford, Douglas: *JavaScript - the good parts: unearthing the excellence in JavaScript*. O'Reilly, 2008. – I–XIII, 1–153 S. – ISBN 978–0–596–51774–8

[41] Michálek, Martin: *Vzhůru do CSS3*. 2015. – ISBN 978–80–260–8440–2

[42] Michálek, Martin: *Vzhůru do (responzivního) webdesignu*. 2017. – 1–268 S. – ISBN 978–80–88253–00–6

[43] React: *React – a JavaScript library for building user interfaces*. https://reactjs.org/

[44] Prettier: *Prettier · Opinionated Code Formatter*. https://prettier.io/. – (cit. 15. 5. 2022)

[45] UI, Material: *MaterialUI – React UI components library*. https://mui.com/

[46] Dehnokhalaji, Hossein: *Notistack*. https://github.com/iamhosseindhv/notistack

[47] Conf, React: *React Today and Tomorrow and 90% Cleaner React With Hooks*. https://www.youtube.com/watch?v=dpw9EHDh2bM, 2018

[48] Chen, Thor: *React Design Patterns: Return Component From Hooks*. https://blog.bitsrc.io/new-react-design-pattern-return-component-from-hooks-79215c3eac00. Version: Feb 2022

[49] React: *React patterns on github*. https://reactpatterns.com/

[50] Next: *API Routes Introduction Next*. https://nextjs.org/docs/api-routes/introduction

[51] Twarog, Adrian: *UI / UX Design Tutorial – From Zero to Hero with Wireframe + Prototype + Design in Figma*. https://www.freecodecamp.org/news/ui-ux-design-tutorial-from-zero-to-hero-with-wireframe-prototype-figma/

[52] Google: *Dark theme - Material Design*. https://material.io/design/color/dark-theme.html#properties

[53] Ferrer, Josep: *Why is Dark Mode so captivating?* https://uxdesign.cc/why-is-dark-mode-so-captivating-92f2ed4e0dc5. Version: 2022

[54] Altexsoft: *Functional and Nonfunctional Requirements: Specification and Types*. https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/. Version: 2021

[55] Microsoft: *TypeScript: JavaScript With Syntax For Types*. https://www.typescriptlang.org. Version: 2022

[56] Criswell, Lindsay: *Destructuring Props in React*. https://medium.com/\spacefactor\@m{}lcriswell/destructuring-props-in-react-b1c295005ce0. Version: 2018. – (cit. 15. 5. 2022)

[57] GATHONI, MARY: *Destructuring Props in React*. https://www.makeuseof.com/must-follow-react-practices/. Version: březen 2022. – (cit. 15. 5. 2022)

[58] Adedotun, Adebiyi: *How To Use Styled-Components In React*. https://www.smashingmagazine.com/2020/07/styled-components-react/. Version: červenec 2020. – (cit. 15. 5. 2022)

[59] House, Chris: *A Complete Guide to Grid*. https://css-tricks.com/snippets/css/complete-guide-grid/. Version: květen 2021. – (cit. 15. 5. 2022)

[60] Date-fns: *Modern JavaScript date utility library*. https://date-fns.org. Version: 2022. – (cit. 16. 5. 2022)

[61] Michálek, Martin: *Webpack: Úplné základy a tutoriál k tomu*. https://www.vzhurudolu.cz/prirucka/webpack. – (cit. 17. 5. 2022)

[62] GAGLIARDI, VALENTINO: *Code documentation for JavaScript with JSDoc: an introduction*. https://www.valentinog.com/blog/jsdoc/. – (cit. 18. 5. 2022)