

Modul kvízů pro vzdělávací platformu ROSTEM2050

Jakub Hrabec



Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2024/2025

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: **Jakub Hrabec**
Osobní číslo: **A22010**
Studijní program: **B0613A140020 Softwarové inženýrství**
Forma studia: **Prezenční**
Téma práce: **Modul kvízů pro vzdělávací platformu ROSTEM2050**
Téma práce anglicky: **Quiz Module for the ROSTEM2050 Learning Platform**

Zásady pro vypracování

Body zadání:

- V teoretické části popište a srovnajte moderní vývojové JavaScriptové frameworky. Zaměřte se především na framework Vue.js.
- Získejte a vypište funkční a nefunkční požadavky na webovou aplikaci pro tvorbu a úpravu kvízů.
- Navrhněte intuitivní uživatelské rozhraní aplikace pomocí nástroje Figma.
- Na základě výše uvedeného návrhu a požadavků vytvořte v rámci projektu ROSTEM 2050 sadu Vue.js komponent, které uživatelům aplikace umožní vytvářet a zobrazovat kvízy.
- Dále implementujte funkce pro správu kvízů, jako je editace a odstraňování otázek v kvízech, upozornění na nesprávně vyplněná pole a sledování výsledků účastníků kvízu.
- Zdokumentujte proces vývoje a integraci jednotlivých komponent.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. SHAVIN, Maya. Learning Vue: Core concepts and practical patterns for reusable, composable, scalable user interfaces. Sebastopol: O'Reilly Media, 2023. ISBN 9781492098829.
2. HAVERBEKE, Marijn. Eloquent JavaScript, 4th Edition. San Francisco: No Starch Press, 2024. ISBN 9781718504103.
3. YANG, HcySun. Vue.JS Framework: Design and Implementation. Singapore: Springer, 2023. ISBN 9789819949472.
4. FLANAGAN, David. JavaScript: The Definitive Guide : Master the World's Most-used Programming Language. Sebastopol: O'Reilly Media, 2020. ISBN 9781491952023.
5. SARRION, Eric. JavaScript from Frontend to Backend: Learn full stack JavaScript development using the MEVN stack with quick and easy steps. Birmingham: Packt Publishing, 2022. ISBN 9781801074148.

Vedoucí bakalářské práce: **Ing. Adam Ulrich**
Ústav matematiky

Datum zadání bakalářské práce: **27. října 2024**

Termín odevzdání bakalářské práce: **2. června 2025**



L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 17. listopadu 2024

PROHLÁŠENÍ AUTORA BAKALÁŘSKÉ PRÁCE

Beru na vědomí, že

- odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., v platném znění bez ohledu na výsledek obhajoby;
- bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému a bude dostupná k nahlédnutí;
- jedno vyhotovení bakalářské práce v listinné podobě bude ponecháno Univerzitě Tomáše Bati ve Zlíně k uložení;
- na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- podle § 60 odst. 2 a 3 mohu užít své dílo – bakalářskou práci – nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tj. k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá; neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji, že

- jsem na bakalářské práci pracoval(a) samostatně a použitou literaturu jsem řádně citoval(a); v případě publikace výsledků budu uveden(a) jako spoluautor;
- odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou obsahově totožné.

Ve Zlíně, dne

.....
podpis autora

ABSTRAKT

Bakalářská práce představuje proces vývoje webové aplikace, konkrétně modulu pro tvorbu kvízů, s využitím frameworku Vue. Teoretická část seznámí čtenáře se základními pojmy spojenými s vývojem webových aplikací a přiblíží populární knihovny a vývojové frameworky, s hlavním zaměřením na Vue.js a jeho funkcionalitu. Praktická část popisuje proces vývoje modulu kvízů pro vzdělávací platformu ROSTEM2050, včetně návrhu grafického rozhraní v aplikaci Figma, definice požadavků na modul a komentované ukázky kódu. Závěr práce hodnotí finální aplikaci v kontextu splnění stanovených požadavků.

Klíčová slova: webová aplikace, modul, kvíz, HTML, CSS, Bootstrap, Javascript, framework, Vue, React, Angular

ABSTRACT

This Bachelor's thesis presents the process of developing a web application, specifically a quiz module using the Vue framework. The theoretical part introduces reader to fundamental concepts related to web application development, and includes an overview of popular libraries and development frameworks, with the main focus on Vue.js and its functionality. The practical part describes the development process of the quiz module for the ROSTEM2050 educational platform, including the design of the graphical interface in Figma, the definition of requirements for the module, as well as commented code examples. The conclusion evaluates the final application in the context of meeting specified requirements.

Keywords: web application, module, quiz, HTML, CSS, Bootstrap, Javascript, framework, Vue, React, Angular

Chtěl bych poděkovat vedoucímu mé bakalářské práce, panu Ing. Adamovi Ulrichovi, za vedení práce, ochotu, a poskytnutí cenných rad při vývoji modulu.

Prohlašuji, že jsem při tvorbě této práce použil nástroj generativního modelu AI ChatGPT, dostupného na <https://chatgpt.com/>, za účelem kvalitnější formulace textu. Po použití tohoto nástroje jsem provedl kontrolu obsahu a přebírám za něj plnou zodpovědnost.

OBSAH

ÚVOD	11
I TEORETICKÁ ČÁST	12
1 PROJEKT ROSTEM 2050	13
2 MODUL KVÍZŮ	15
2.1 MOTIVACE PRO PŘIDÁNÍ	15
2.2 FUNKCIONALITA	16
2.3 LIMITY APLIKACE	17
3 POUŽITÉ TECHNOLOGIE	19
3.1 GIT	19
3.2 GITHUB	20
3.3 WSL	20
3.4 DOCKER	21
3.5 FIGMA	21
3.6 JAVASCRIPT	22
3.7 VUE.JS	22
3.7.1 Reaktivita	23
3.7.2 Pinia store	24
3.7.3 Vue.js vs React	24
3.7.4 Vue.js vs Angular	25
II PRAKTICKÁ ČÁST	27
4 ANALÝZA POŽADAVKŮ	28
4.1 FUNKČNÍ POŽADAVKY	28
4.2 NEFUNKČNÍ POŽADAVKY	30
5 NÁVRH UŽIVATELSKÉHO ROZHRANÍ	31
5.1 TVORBA KVÍZŮ	32
5.1.1 Záložka Obecné informace	32
5.1.2 Záložka Otázky	33

5.2	PŘEHLED KVÍZŮ	36
5.3	DETAIL KVÍZU	37
5.4	EDITACE KVÍZU	38
5.5	ŽEBŘÍČEK SKÓRE	38
6	INTEGRACE MODULU S OSTATNÍMI ČÁSTMI APLIKACE	39
6.1	FUNKCE SUBMITQUIZ	40
6.2	FUNKCE UPDATEQUIZ	41
6.3	FUNKCE GETQUIZ	41
6.4	FUNKCE DELETEQUIZ	42
7	IMPLEMENTACE HLAVNÍCH KOMPONENT	44
7.1	KOMPONENTA CREATEQUIZ	44
7.1.1	Navigace v komponentě	44
7.1.2	Správa otázek	45
7.1.3	Ukládání kvízů	47
7.2	KOMPONENTA VIEWQUIZ	48
7.2.1	Správa kvízů	49
7.2.2	Žebříčky studentů	50
7.3	LOCAL STORAGE	50
	ZÁVĚR	51
	SEZNAM POUŽITÉ LITERATURY	53
	SEZNAM OBRÁZKŮ	56
	SEZNAM TABULEK	57
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	58
	SEZNAM PŘÍLOH	59

ÚVOD

Vzdělávání se v poslední době stále více přesouvá do online prostoru, což zvyšuje nároky na interaktivitu výukových platforem. Jedním z oblíbených nástrojů, které mohou studentům i učitelům při výuce pomoci, jsou kvízy. Ty umožňují okamžitou kontrolu znalostí a motivují studenty k aktivnější účasti.

Tato bakalářská práce je zaměřena na vývoj kvízového modulu pro webovou vzdělávací platformu ROSTEM 2050. Cílem modulu je zpestřit workshopy pro studenty a zvýšit tak jejich zapojení, a současně poskytnout lektorům nástroj pro ověření, co si studenti z workshopů odnesli. Modul je implementován pomocí JavaScriptového frameworku Vue.js a umožňuje tvorbu, zobrazení a správu kvízů.

Teoretická část nejprve krátce představí projekt ROSTEM 2050. Dále pojednává o důvodech pro přidání kvízového modulu, základní funkcionalitě a limitech aplikace. Největší část je věnována stručnému popisu technologií použitých pro vývoj modulu.

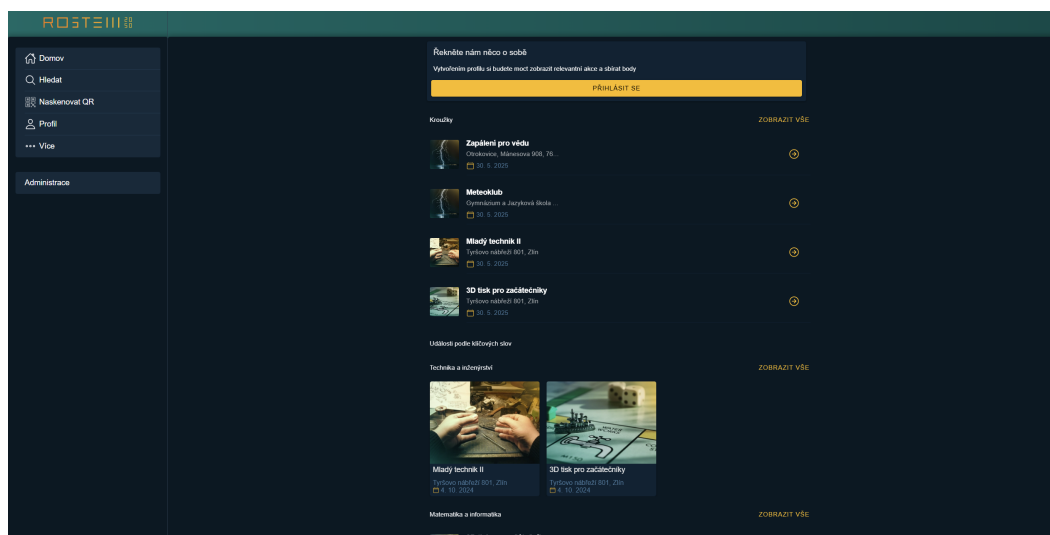
Praktická část je zaměřena na proces implementace modulu kvízů. První kapitola se věnuje analýze požadavků, přičemž jsou vypsány funkční i nefunkční požadavky. Následující kapitola popisuje grafický návrh uživatelského rozhraní ve Figmě a porovnává jej s rozhraním výsledné aplikace. Poslední dvě kapitoly se věnují integraci modulu se zbytkem aplikace a detailnímu popisu implementace modulu s ukázkami kódu.

I. TEORETICKÁ ČÁST

1 PROJEKT ROSTEM 2050

ROSTEM 2050 je webová a mobilní aplikace, která podporuje vzdělávání a osobní rozvoj žáků základních a středních škol v mimoškolních aktivitách. Zaregistrovaní uživatelé se prostřednictvím aplikace mohou přihlašovat na workshopy, volnočasové aktivity a přednášky s polytechnickým zaměřením. [1] Za aktivitu v aplikaci dostávají žáci kredity, což jsou body, které odrážejí aktivitu na workshopech a nasbírané zkušenosti. Nasbírané kredity se potom započítávají do celkového žebříčku nejaktivnějších studentů. Každoročně jsou vyhlášeni vítězové, kteří za svou účast a aktivitu na workshopech získávají ceny.

Aplikace je rozdělena na dvě hlavní části – klientskou a administrátorskou. Klientská část je implementována jako Progressive Web App (PWA) a je dostupná na mobilních zařízeních i v běžném webovém prohlížeči. Administrátorská část je určena pro učitele, vedoucí workshopů a správce aplikace. Uživatelé skrze ni mohou spravovat workshopy, vytvářet aktivity, evidovat účast studentů, nebo využívat nový modul kvízů. Právě návrh a následná implementace modulu pro tvorbu a správu kvízů v administrátorské části je předmětem této bakalářské práce.



Obrázek 1.1 Hlavní stránka ROSTEM 2050

(zdroj: vlastní)

Cílovou skupinou aplikace jsou především zvědaví a tvořiví studenti středních a základních škol, kteří mají zájem o rozšíření svých znalostí v tématech jako jsou technika nebo informatika. Cílem projektu je nejen vzdělávat a rozvíjet talentované studenty, ale také podpořit jejich zájmy a koníčky. Získané zkušenosti mohou studenti posléze uplatnit v budoucí

kariéře nebo v osobním životě.

Aplikace je podporována Zlínským krajem a na vývoji se podílí mimo jiné také Fakulta aplikované informatiky UTB. [1] Backend je implementován pomocí frameworku Laravel, frontend je tvořen již zmíněnou PWA aplikací. Tyto části projektu nejsou předmětem mé práce, na jejich vývoji se podílejí jiné týmy. Mým úkolem je pouze návrh a realizace modulu pro tvorbu a správu kvízů v rámci administrátorské části aplikace.

Zdrojový kód: <https://github.com/FAISTEM/projekt-repo/tree/quizes-with-backend>

Návrh UI: <https://www.figma.com/design/hA8epIQuclKmgsIaFmHnZp/ROSTEM-2050>

2 MODUL KVÍZŮ

Během vývoje aplikace ROSTEM 2050 se objevil požadavek na rozšíření systému o novou funkcionalitu, která by umožnila lektorům získat lepší přehled o znalostech studentů. Na základě požadavku vznikla myšlenka na vytvoření modulu kvízů, který by danou funkcionalitu zajistil.

2.1 Motivace pro přidání

Jedním z hlavních cílů aplikace ROSTEM 2050 je podpora vzdělávání a rozvoje studentů základních a středních škol formou mimoškolních aktivit. Nový modul kvízů do aplikace přinese několik zásadních změn, které mohou mít pozitivní dopad jak na studenty, tak na vedoucí workshopů.

Kvízy studentům poskytnou interaktivní a především zábavnou formu výuky, čímž je motivují k aktivní účasti. Kvízy představují prvek gamifikace, díky kterým se zvýší míra zapojení uživatelů do aplikace i do samotných workshopů. [2] Studenti, kteří konzistentně dosahují vysokých skóre v kvízech, se navíc umístí na vyšších pozicích v celkovém žebříčku. Tím se zvýší jejich šance na získání ocenění, a žáci tak mají motivaci k větší snaze a pravidelné účasti na workshopech.

Na druhou stranu, z pohledu učitelů a vedoucích workshopů kvízy slouží také jako nástroj pro okamžité vyhodnocení, co si studenti z výuky zapamatovali. Umožňují jednoduše měřit efektivitu vzdělávacích materiálů a identifikovat problematické oblasti. To vyučujícím pomůže upravit obsah budoucích workshopů podle potřeb studentů. Výsledky kvízů poskytují přehled o znalostech studentů v reálném čase, učitel tudíž nemusí na zpětnou vazbu čekat. Kvízy se navíc samy vyhodnotí, lektor tak nemusí práce všech studentů jednu po druhé opravovat.

Další významnou výhodou modulu kvízů je možnost jeho využití v rámci distanční výuky. Protože se jedná o webovou aplikaci, nemusí být studenti na workshopech fyzicky přítomni, aby mohli kvízy vyplňovat.

Implementace modulu kvízů celkově pomůže modernizovat výuku a podpoří aktivní učení. Dále poskytne vedoucím workshopů nástroje pro zpětnou vazbu a přizpůsobení výuky potřebám studentů.

2.2 Funkcionalita

Modul kvízů je přístupný pouze přihlášeným uživatelům s přístupem k administrátorské části aplikace ROSTEM 2050, tedy především vedoucím workshopů a správcům systému. Uživatelé skrze něj mohou vytvářet, upravovat, mazat a vyhodnocovat jednotlivé kvízy. Modul kvízů je součástí administrátorské části systému ROSTEM, v práci se nadále zaměřuji výhradně na funkcionalitu mého modulu. Administrátorské prostředí obsahuje množství dalších funkcí, jako například nástroje pro správu studentů a institucí nebo kalendář, na vývoji těchto funkcí jsem se ovšem nepodílel.

Základní funkcionalita modulu zahrnuje:

- **Vytvoření nového kvízu** – uživatel může zadat název a popis, a vygenerovat přístupový kód. Rovněž v rámci kvízu může přidávat otázky.
- **Editace existujících kvízů** – uživatel může již existující kvíz zpětně upravit. Uživateli je umožněno upravit název, popis a jednotlivé otázky.
- **Mazání existujících kvízů** – uživatel může již existující kvíz smazat.
- **Přidávání otázek do kvízu** – jsou podporovány dva typy otázek:
 - **Otevřená otázka (open)** – lektor při tvorbě otázky určí, jak má vypadat správná odpověď. Student pak napíše vlastní odpověď.
 - **Uzavřená otázka (closed/multichoice)** – student vybírá jednu nebo více správných odpovědí z předem daných možností. Možností může být 2 až 9. Lektor při tvorbě otázky nejprve zvolí počet možností, následně jednotlivé možnosti vyplní, a nakonec určí, které z nich mají být považovány za správné.
- **Správa otázek** – jednotlivé otázky lze v rámci kvízu přidávat, upravovat či mazat. Lze rovněž zpětně změnit typ otázky.
- **Validace** – před odesláním kvízu do backendu jsou validována všechna pole kvízu. Validace je prováděna jak na úrovni jednotlivých otázek, například aby nebylo možné uložit otázku bez zadání nebo bez odpovědi, tak i na úrovni kvízů, aby uživatel nemohl uložit kvíz bez názvu či bez jediné otázky.

- **Intuitivní a přehledné uživatelské rozhraní** – modul je navržen tak, aby byl snadno použitelný i pro méně technicky zdatné uživatele. Pro jeho používání není třeba studovat dokumentaci. Zároveň vizuálně ladí se zbytkem administrátorské části aplikace.

2.3 Limity aplikace

Modul v současnosti splňuje výše uvedené body, přesto však není zcela dokončen. Stále na něm pracuji a do budoucna se plánuje jeho rozšíření. Realizace další funkcionality je omezena nedokončeným napojením na mobilní a webový frontend. Tato část byla vyvíjena mým kolegou, a z důvodu omezené komunikace a kolegovy časové vytíženosti zatím nedošlo k integraci našich částí. Modul je tedy funkční, dosud však nebyl propojen s frontendem. Přestože určité funkce nejsou uvedeny v základní funkcionalitě, považuji jejich přítomnost v modulu kvízu za nezbytnou pro plnohodnotnou funkčnost. Uvedené funkce budou do modulu v budoucnu doplněny.

Funkcionalita, kterou modul kvízů momentálně neobsahuje, nebo je nekompletní:

- **Vyhodnocení kvízů a zobrazení žebříčků** – stránka s žebříčkem studentů v současnosti obsahuje pouze testovací (dummy) data. Modul rovněž zatím neobsahuje pohled s podrobnějšími statistikami o úspěšnosti studentů, lektor aktuálně nemá možnost analyzovat úspěšnost studentů v jednotlivých otázkách a zobrazit jejich celkové skóre. Důvodem absence této funkcionality je závislost na dokončení odpovídající části frontendové aplikace, která dosud nebyla implementována.
- **Udělování kreditů** – tento bod se váže k předchozímu. Jelikož ještě není implementováno vyhodnocení kvízů, není možné studenty ohodnotit.

Modul má být snadným pomocníkem pro zpestření workshopů a získání zpětné vazby od studentů. Proto postrádá některé pokročilejší funkce, které nejsou pro jeho základní využití nezbytné. Implementace určitých funkcí byla zvažována, nicméně se od realizace upustilo z důvodu časové náročnosti, technické složitosti nebo špatné kompatibility s backendem. Příklady funkcí, které obdobné kvízové aplikace zahrnují a případně by mohly být v budoucnu do mého modulu přidány:

- **Otázky s obrázky** – nejedná se o nezbytnou funkcionalitu, v určitých tématech by však mohly být užitečné. Daly by se využít například pro geometrické úlohy.

- **Uložení konceptu kvízu** – jedná se o quality-of-life feature pro lektory. Lektor by mohl svůj rozpracovaný kvíz uložit, aniž by byl kvíz viditelný pro studenty. K rozpracovanému kvízu by se lektor později mohl vrátit.

3 POUŽITÉ TECHNOLOGIE

Před představením samotného řešení a implementace projektu upřesním, jaké technologie jsem při vývoji používal. S většinou z těchto nástrojů jsem se již před začátkem práce na projektu setkal, některé jsem však v rámci projektu používal poprvé, například Docker.

3.1 Git



Obrázek 3.1 Logo Gitu

(zdroj: 3)

Git je open-source systém pro správu verzí (VCS), při vývoji softwaru se využívá k uchovávání a sledování změn v souborech. Vytvořil ho Linus Torvalds v roce 2005 poté, co vývojáři jádra Linuxu opustili jimi dosud používaný verzovací systém BitKeeper, u něhož byla zrušena možnost bezplatného užívání. [4] Git vznikl jako nástroj pro vývoj linuxového jádra, od té doby se ale uchytil a stal se standardem pro verzování kódu. Z výsledků průzkumu Stack Overflow vyplývá, že jej v roce 2022 využívalo více než 96 % profesionálních vývojářů. [5] Umožňuje více uživatelům efektivně spolupracovat na jednom projektu.

Oproti starším systémům je hlavní výhodou Gitu jeho distribuovaná a decentralizovaná povaha. Každý vývojář má na svém počítači kompletní kopii celého repozitáře včetně historie. [6] Díky tomu je práce s Gitem rychlá a především bezpečná. Pokud dojde k nechtěným změnám, lze tyto změny zrušit nebo se vrátit k předchozí verzi. Jelikož je kopie repozitáře uložena lokálně na počítači vývojáře, práce s Gitem funguje dobře i offline. Zaznamenávání změn probíhá pomocí commitů, které obsahují informace o tom, kdo změnu provedl, kdy a co přesně bylo upraveno. Jednotlivými commity lze procházet a dá se mezi nimi libovolně přesouvat.

Git vývojářům také umožňuje větvení (branching) a slučování (merging). Branchingem se myslí oddělení nové vývojové větve od hlavní, což umožňuje implementaci nových funkcí či experimentování bez zasahování do hlavního kódu. Merging je naopak proces, kdy se změny z jedné vývojové větve sloučí do jiné. Git porovná změny mezi oběma větvemi a pokud ne-

dojde ke konfliktu, spojí je. V případě konfliktu musí vývojář ručně určit, jak má výsledný kód vypadat. Díky této funkcionalitě může více vývojářů pracovat na určitých funkcích bez vzájemného ovlivnění. Git tak podporuje týmovou spolupráci, přehlednost kódu a agilní vývoj.

Funkcionalitu větvení a slučování jsem využil i při práci na kvízovém modulu. Původně jsem pracoval na své vlastní branchi *quizes*. Po dokončení části backendu, která byla kolegou implementována na větvi *quizzes-api*, došlo ke sloučení obou větví. Jednalo se o jednu z nejsložitějších částí vývoje, jelikož bylo nutné vyřešit množství konfliktů v souborech. Po úspěšném vyřešení konfliktů jsem vytvořil novou větev *quizzes-with-backend*, ve které je můj modul již napojen na backend.

3.2 GitHub

GitHub je webová platforma pro hostování repozitářů využívající systém Git. Webové rozhraní GitHubu umožňuje vývojářům ukládat, sdílet a spolupracovat na projektech spravovaných přes Git. Kromě základních funkcí pro verzování zahrnuje také nástroje pro správu úkolů a sledování a reportování problémů (issues).

V současnosti má GitHub přes 150 milionů uživatelů a přes 420 milionů hostovaných repozitářů, což z něj činí nejpoblárnější platformu tohoto druhu. [7] Většina funkcí platformy je dostupná zdarma, za příplatek jsou nabízeny některé pokročilé funkce a větší úložiště. GitHub se těší značné oblíbě také v open-source komunitě. Díky forkování mohou uživatelé vytvořit vlastní kopii veřejného projektu, upravit ji, a následně navrhnout změny přes pull request. [8] Správce hlavního repozitáře pak případně může změny přijmout a začlenit je do projektu.

3.3 WSL

Windows Subsystem for Linux (WSL) je funkce, která umožňuje spouštět linuxové prostředí přímo ve Windows. [9] Uživatel využívající oba operační systémy tak nemusí instalovat virtuální stroje ani používat dual-boot. WSL umožňuje používat běžné linuxové nástroje, skripty a příkazy přímo v rámci Windows, čímž usnadňuje práci při vývoji softwaru.

Na Windows lze prostřednictvím WSL nainstalovat určité linuxové distribuce, které se posléze spouštějí jako samostatné prostředí přes terminál. Výhodou WSL je možnost kom-

binovat Windows a Linux nástroje, bez potřeby dual-bootu nebo instalace virtual machine. [9] Umožnilo mi to psát kód v mém oblíbeném Windows editoru (VS Code), kód jsem ale následně mohl spustit a testovat v Linuxovém prostředí.

3.4 Docker



Obrázek 3.2 Logo Dockeru

(zdroj: 10)

Docker je platforma pro kontejnerizaci aplikací, umožňuje vývojářům balit a spouštět aplikace v tzv. kontejnerech. V kontejneru je uložena samotná aplikace spolu se vším, co potřebuje ke svému běhu (například knihovny, nastavení nebo závislosti) [11]. Aplikace díky tomu může z pohledu vývojáře běžet stejně na různých počítačích, bez ohledu na rozdíly v prostředí. [12] Docker výrazně usnadňuje spolupráci v týmu, jelikož eliminuje problémy, kdy aplikace na počítači jednoho vývojáře funguje, ale u druhého vývojáře ne. Docker je obecně rychlejší a méně náročný na systémové prostředky, než klasické virtuální stroje. [13]

Kromě řešení problémů s přenositelností má Docker i další využití, například podporuje nástroj Docker Compose pro správu více kontejnerů najednou. V rámci projektu ROSTEM jsem Docker využil právě k tomuto účelu, bylo třeba současně spustit více částí systému (např. mobilní frontend, administrátorskou část, databázi) na jednom localhostu.

3.5 Figma

Figma je online grafický editor pro návrh uživatelského rozhraní (UI). Oproti tradičním desktopovým editorům nevyžaduje instalaci a umožňuje více uživatelům pracovat na stejném návrhu současně. [14] Díky tomu je ideální volbou pro tvorbu návrhu UI v rámci týmu. K návrhům je možné přidávat komentáře, takže členové týmu mají možnost přidávat zpětnou vazbu přímo v aplikaci. U jednotlivých UI prvků jsou uvedeny podrobné informace, jako přesná pozice, rozměry, a hex kód barvy. Vývojáři tak jednoduše získají přístup k přesným specifikacím.

Figma nabízí předpřipravené komponenty a styly, většinou sdružené do ucelených sad (UI

kitů). V aplikaci lze nalézt například sadu UI prvků pro iOS 18 od společnosti Apple. Mezi další často využívané sady patří Bootstrap 5 UI kit, obsahující prvky stejnojmenného CSS frameworku, nebo Material 3 Design Kit. Použití předpřipravených šablon a stylů zajišťuje vizuální konzistenci napříč designem, což je u větších projektů důležité.

3.6 JavaScript

JavaScript je skriptovací jazyk pro tvorbu interaktivních webových stránek. V současnosti jej využívá převážná většina webových stránek a všechny moderní webové prohlížeče obsahují vlastní interprety JavaScriptu. [15, s. 4] Jazyk byl poprvé představen v roce 1995 jako nástroj pro tehdy populární webový prohlížeč Netscape Navigator [16, s. 5], od té doby se stal nejrozšířenějším programovacím jazykem v historii [15, s. 4]. Podle výsledků průzkumu Stack Overflow používalo JavaScript v roce 2024 více než 64 % profesionálních vývojářů, což jej v současné době činí nejpobulárnějším jazykem. [17] K popularitě jazyka přispělo prostředí Node.js, které umožňuje spouštět JavaScript i na straně serveru, a také frameworky jako Vue.js, které usnadňují vývoj moderních webových aplikací.

3.7 Vue.js



Obrázek 3.3 Logo Vue.js

(zdroj: 18)

Vue.js je moderní JavaScript framework pro vývoj dynamických a reaktivních webových aplikací. Klade důraz na jednoduchost a přehlednost. Byl představen Evanem You začátkem roku 2014 a od té doby se stal jedním z nejpobulárnějších frameworků. Z výsledků průzkumu State of Javascript vyplývá, že Vue.js byl v roce 2024 po Reactu druhým nejpobulárnějším frontendovým frameworkem. [19] Framework je využíván mnoha světově známými společnostmi, jako například Netflix, Xiaomi, Adobe, Gitlab nebo Nintendo. [20] Od verze Vue 3 plně podporuje TypeScript. [21, s. 29]

Jednou z hlavních předností frameworku Vue.js je možnost vývoje aplikace pomocí komponent. [22, s. 95] Komponenty představují menší, znovupoužitelné části, ze kterých se

skládá celkové uživatelské rozhraní. Každá komponenta zahrnuje HTML šablonu, logiku napsanou v jazyce JavaScript (případně TypeScript) a CSS stylování. Tyto části jsou v kódu organizovány do samostatných bloků. Blok *template* obsahuje HTML kód, blok *script* slouží pro JavaScript logiku a blok *style* definuje design v kaskádových stylech. Jednotlivé komponenty mohou být využity v rámci jiných projektů a mohou být nezávisle testovány. [22, s. 95] Tento přístup ke komponentám, kdy jsou HTML, CSS a JavaScript sjednoceny do jednoho .vue souboru, se nazývá SFC (Single File Component).

Základem Vue je podpora vývoje SPA (Single Page Applications). V takových aplikacích se obsah stránek mění dynamicky a pro aktualizaci dat není nutné znovunačítání celé webové stránky. Aplikace díky tomu působí svižněji a plynuleji, což zlepšuje uživatelský zážitek.

Ekosystém Vue.js zahrnuje množství užitečných nástrojů. Mezi ně patří například navigační nástroj Vue Router, který umožňuje jednoduchou navigaci mezi Vue komponentami. Navigace probíhá v prohlížeči na straně klienta, tudíž není třeba znovu načítat celou stránku. Za zmínku stojí také sada nástrojů Vue Devtools, umožňující prohlížet strom komponent Vue, vlastnosti a data komponent, a informace o směrování v rámci vývojářské konzole prohlížeče. [23]

3.7.1 Reaktivita

Reaktivita je jednou z klíčových vlastností Vue frameworku. Umožňuje vývojářům vytvářet interaktivní uživatelská rozhraní automatickým sledováním změn v datech a aktualizací DOM. [24] Pokud dojde ke změně dat v komponentě, odpovídající část DOM se sama aktualizuje. Pro aktualizaci není nutné manuálně manipulovat s rozhraním.

Ve vue.js reaktivitu zajišťuje několik základních funkcí: [24]

- **ref()** – používá se pro vytvoření reaktivní proměnné. Typicky se používá pro primitivní datové typy.
- **reactive()** – funguje podobně jako ref(), ale používá se pro objekty či pole.
- **computed()** – definuje odvozené hodnoty, které se automaticky přepočítávají, pokud se změní vstupní data. Používá se například pro výpočty, které závisí na jiných datech.
- **watch()** – sleduje změny konkrétních reaktivních proměnných a umožňuje na tyto změny reagovat. Užitečné v situacích, kdy je potřeba provést vedlejší efekt (například

volání API).

3.7.2 Pinia store

Pinia je oficiální knihovna pro správu globálního stavu ve Vue.js. Byla vydána v roce 2019 a postupně nahradila Vuex jako doporučený nástroj pro správu stavu. [25] Umožňuje centralizované řízení stavu aplikace mimo jednotlivé komponenty. Mezi přednosti Pinie patří přehledná syntaxe, plná podpora TypeScriptu a jednoduché API.

Základem Pinie je tzv. store. Je to samostatná jednotka, která obsahuje stav (state) a akce (actions). Store lze definovat funkcí `defineStore`, takto definované stores pak můžeme používat v ostatních komponentách aplikace. Store je reaktivní, takže jakékoli změny stavu se automaticky projeví ve všech komponentách, kde jsou daná data použita. Pinia je oproti Vuex odlehčenější a má lepší podporu pro TypeScript.

3.7.3 Vue.js vs React

React je JavaScriptový framework vyvíjený společností Meta (dříve Facebook). Od svého vydání v roce 2013 si získal značnou oblibu a již přes deset let patří k nejpoužívanějším frontendovým frameworkům.

S Vue.js má React hodně společného. Oba frameworky jsou open-source a založené primárně na jazyce JavaScript. Vue i React využívají virtuální DOM, který umožňuje aktualizaci pouze těch částí uživatelského rozhraní, které se skutečně změnily, není tedy nutné překreslovat celé UI. Tento proces šetří čas a výpočetní zdroje, a zajišťuje rychlejší načítání. Architektura obou frameworků je založena na komponentách, uživatelské rozhraní je rozděleno na opakovaně použitelné prvky. [26]

Frameworky se liší po stránce syntaxe a stylu psaní komponent. Vue využívá Single File Components (SFC), kdy jsou HTML, CSS a JavaScript sdruženy do jednoho souboru, ale funkčně zůstávají odděleny v rámci bloků. React využívá rozšíření JavaScriptu JSX, které umožňuje kombinovat HTML a CSS přímo v JavaScriptovém kódu. JSX nabízí větší flexibilitu, současně však vyžaduje hlubší znalost JavaScriptu. Vue je obecně považováno za jednodušší na pochopení a dostupnější i pro začátečníky, kteří mají s JavaScriptem omezené zkušenosti. React má strmější křivku učení. [26]

Z hlediska velikosti komunity je React jasným vítězem. Díky zázemí společnosti Meta má

React větší dosah, širší nabídku dostupných řešení a tutoriálů, a více otázek na fórech jako Stack Overflow. Vue má středně velkou, avšak aktivní a rostoucí komunitu. Dokumentace Vue.js má dobrou reputaci, je přehledná a dobře strukturovaná. Dokumentace Reactu prošla v posledních letech velkými změnami a došlo ke zlepšení, orientace v ní je však stále relativně náročná. [26]

Z hlediska výkonu záleží na konkrétním použití. Vue mívá menší velikost balíčku a rychlejší čas sestavení, což zajišťuje lepší odezvu u menších aplikací. React se díky svému širokému ekosystému uplatní v rozsáhlejších systémech. [26]

Při rozhodování mezi Vue.js a Reactem by měl vývojář především zohlednit typ a rozsah projektu. Vue je vhodnější pro méně komplexní aplikace s požadavky na rychlou odezvu, React je lepší volbou pro středně komplexní až komplexní systémy. Finální srovnání lze vidět v tabulce. 3.1

Tabulka 3.1 Srovnání frameworků Vue.js a React

Vlastnost	Vue.js	React
Vydání	2014	2013
Primární jazyk	JavaScript	JavaScript
DOM	Virtuální DOM	Virtuální DOM
Komunita	Středně velká	Velká
Křivka učení	Nižší	Střední
Ekosystém	Středně velký	Velký
Jednoduchost	Vysoká	Střední
Vhodné použití	Menší aplikace	Komplexnější aplikace

(zdroj: vlastní)

3.7.4 Vue.js vs Angular

Angular je vývojový framework založený na jazyce TypeScript. Byl představen v roce 2016 jako nástupce staršího frameworku AngularJS založeného na JavaScriptu. Je vyvíjen společností Google.

Stejně jako Vue je i Angular open-source a využívá komponentový přístup, kdy je uživatelské rozhraní rozděleno do samostatných, znovupoužitelných částí. [27]

Zásadní rozdíl mezi frameworky spočívá v používaném jazyce a složitosti. Vue.js je založen na JavaScriptu, zatímco Angular primárně používá TypeScript. Pro používání Angularu je vyžadována znalost MVC a TypeScriptu, má proto značně strmější křivku učení. Angu-

lar na rozdíl od Vue využívá reálný DOM, který renderuje celou stránku i při změně jedné komponenty, což může vést k pomalejšímu vykreslování. [27]

Angular má ve srovnání s Vue širší komunitu a více dostupných řešení. Má rozsáhlý ekosystém s mnoha rozšířeními a díky tomu je vhodnější pro komplexní aplikace. Angular je oblíbený ve firemním prostředí, kde je požadována stabilita, robustnost a škálovatelnost. [27]

Při rozhodování mezi Vue.js a Angularem je opět nutné vzít v potaz typ aplikace. Framework Vue.js je vhodný pro malé až středně velké projekty, zatímco Angular je vhodnější pro komplexní podnikové aplikace s vysokými nároky na stabilitu a škálovatelnost. Finální srovnání lze vidět v tabulce. 3.2

Tabulka 3.2 Srovnání frameworků Vue.js a Angular

Vlastnost	Vue.js	Angular
Vydání	2014	2016
Primární jazyk	JavaScript	TypeScript
DOM	Virtuální DOM	Reálný DOM
Komunita	Středně velká	Velká
Křivka učení	Nižší	Vyšší
Ekosystém	Středně velký	Velký
Jednoduchost	Vysoká	Nízká
Vhodné použití	Menší aplikace	Rozsáhlejší systémy

(zdroj: vlastní)

II. PRAKTICKÁ ČÁST

4 ANALÝZA POŽADAVKŮ

Analýza požadavků je klíčovou fází při vývoji softwaru. Smyslem analýzy je porozumět tomu, co má výsledný produkt umět a kdo s ním bude pracovat. Dobře zpracovaná analýza minimalizuje riziko nedorozumění mezi vývojářem a koncovým uživatelem aplikace, navíc snižuje náklady na úpravy v pozdějších fázích vývoje.

S kvízy budou pracovat především dva typy uživatelů – učitelé a studenti. Studentům musí být umožněno zobrazit informace o kvízu, po zadání kódu budou moci vyplnit kvíz a následně zobrazit svůj výsledek. Tyto funkce budou implementovány ve webovém frontendu a v mobilní aplikaci. Jelikož pracuji pouze na administrátorské části aplikace, budu se nadále věnovat požadavkům lektorů. Po konzultaci s vedoucím práce a s kolegy jsem stanovil níže uvedené požadavky.

Požadavky popisují, co by měl modul umět. Funkční požadavky se týkají samotné funkcionality a chování aplikace, zatímco nefunkční požadavky se vztahují k technickým parametrům.

4.1 Funkční požadavky

- **F1: Tvorba kvízu** – Uživateli bude umožněno vytvořit kvíz. Uživatel bude mít k dispozici formulář pro zadání veškerých informací o kvízu.
- **F2: Název kvízu** – Uživateli bude umožněno pojmenovat kvíz prostřednictvím vstupního pole.
- **F3: Datum otevření a uzavření kvízu** – Uživateli bude umožněno zadat datum otevření a uzavření kvízu prostřednictvím vstupních polí. Uživatel bude moci dopředu naplánovat, kdy bude kvíz studentům zpřístupněn.
- **F4: Popis kvízu** – Uživateli bude umožněno popsat kvíz prostřednictvím vstupního pole.
- **F5: Přístupnost kvízu** – Uživatel bude moci určit, zda jeho kvíz bude volně přístupný, nebo zda bude vyžadovat kód. K tomuto účelu bude v aplikaci k dispozici checkbox. Kód bude pro kvíz generován automaticky.

- **F6: Přidání otázek** – Uživateli bude umožněno do svého kvízu přidávat otázky prostřednictvím tlačítka + *Nová otázka*.
- **F7: Volba typu otázky** – Uživatel bude mít možnost výběru mezi otevřenou otázkou a uzavřenou (multichoice) otázkou.
- **F8: Otevřená otázka** – V případě výběru otevřené otázky bude uživateli umožněno vyplnit znění správné odpovědi.
- **F9: Uzavřená otázka** – V případě výběru uzavřené otázky bude uživateli umožněno jednotlivé možnosti vyplnit a určit, které z nich mají být považovány za správné.
- **F10: Počet možností** – V případě výběru uzavřené otázky bude uživatel moci zvolit počet možností v otázce, a to v rozmezí od 2 do 9. K tomuto účelu bude v aplikaci k dispozici select menu.
- **F11: Ukládání otázek** – Uživateli bude umožněno uložit otázku prostřednictvím tlačítka *Uložit otázku*.
- **F12: Zahození konceptu otázek** – Uživateli bude během vytváření otázky umožněno zahodit dosavadní postup pomocí tlačítka *Zahodit otázku*.
- **F13: Validace otázek** – Při ukládání otázek systém vyhodnotí, zda jsou všechna požadovaná pole vyplněna. Zadáání otázky a správná odpověď nebo možnosti musí být vyplněny, bez nich by otázka neměla smysl. Pokud bude uživatel chtít uložit otázku a některé z požadovaných polí nebude vyplněno, systém na tuto skutečnost uživatele upozorní prostřednictvím notifikace a otázka nebude uložena.
- **F14: Mazání otázek** – Uživateli bude umožněno smazat otázku prostřednictvím tlačítka *Odstranit*. Systém při mazání zajistí správné přeuspořádání ostatních otázek.
- **F15: Editace otázek** – Uživateli bude umožněno editovat již uloženou otázku prostřednictvím tlačítka *Editovat*. Před editací systém předvyplní pro uživatele všechna pole.
- **F16: Ukládání kvízů** – Uživateli bude umožněno uložit kvíz prostřednictvím tlačítka *Uložit kvíz*.

- **F17: Validace kvízů** – Při ukládání kvízů systém vyhodnotí, zda jsou všechna požadovaná pole vyplněna. Kvíz musí mít vyplněný název a musí obsahovat alespoň 1 otázku. Pokud některé z kritérií není splněno, systém na tuto skutečnost uživatele upozorní prostřednictvím notifikace a kvíz nebude uložen.
- **F18: Zahození konceptu kvízu** – Uživateli bude během vytváření kvízu umožněno zahodit dosavadní postup pomocí tlačítka *Zahodit koncept*.
- **F19: Seznam kvízů** – Uživateli bude umožněno zobrazit seznam všech svých kvízů.
- **F20: Detail kvízu** – Uživateli bude umožněno zobrazit detaily o konkrétním kvízu pomocí akčního tlačítka v přehledu. V detailech budou uvedeny všechny informace, včetně názvu, popisu, otázek a přístupového kódu.
- **F21: Editace kvízu** – Uživateli bude umožněno již vytvořený kvíz editovat pomocí akčního tlačítka v přehledu. Před editací systém předvyplní všechna pole.
- **F22: Mazání kvízu** – Uživateli bude umožněno svůj existující kvíz smazat. K tomuto účelu bude k dispozici akční tlačítko na přehledu kvízů.
- **F23: Navigace** – Systém bude disponovat navigační menu, pomocí kterého bude uživatel moci přecházet mezi pohledy.
- **F24: Vyhodnocení kvízů** – Uživateli bude umožněno zobrazit přehledné vyhodnocení kvízů, včetně žebříčku studentů.

4.2 Nefunkční požadavky

- **N1: Jazyk rozhraní** – Uživatelské rozhraní modulu bude kompletně v českém jazyce.
- **N2: Bezpečnost přístupu** – Přístup ke správě kvízů (vytváření, úprava, mazání) bude umožněn pouze přihlášeným uživatelům v administrátorské části aplikace. Každý uživatel bude mít přístup pouze ke svým kvízům.
- **N3: Dostupnost** – K systému se bude přistupovat přes webový prohlížeč. Systém bude funkční ve všech běžně používaných moderních webových prohlížečích.
- **N4: Data** – Data kvízů budou bezpečně uložena v databázi. Uživatel o své kvízy při odhlášení či obnovení stránky nepřijde.

5 NÁVRH UŽIVATELSKÉHO ROZHRAŇÍ

Před samotnou implementací modulu jsem nejprve navrhl uživatelské rozhraní ve webovém grafickém editoru Figma. Návrh mi pomohl ujasnit si požadavky, promyslet chování jednotlivých částí modulu a vizualizovat rozvržení UI prvků. V průběhu vývoje jsem návrh průběžně konzultoval s vedoucím práce i s kolegy, kteří pracovali na jiných částech systému. Společně jsme posuzovali, které prvky má smysl zachovat, co je naopak zbytečné a co by mohlo způsobit komplikace při reálném nasazení. Konzultace s kolegy mi pomohly předejít chybám a výrazně usnadnily následný vývoj samotné aplikace. Měl jsem již předem jasno, co má rozhraní umět a jaké komponenty budu při práci na modulu potřebovat.

Svůj návrh jsem postupně upravoval na základě zpětné vazby i měnících se požadavků během vývoje. V průběhu vývoje se několikrát stalo, že jsem navrhl UI prvek pro funkci, od které se později upustilo, například kvůli její přílišné složitosti, nevhodnosti, nedostatku času, nebo špatné kompatibilitě s backendem či mobilní aplikací.

Ani finální návrh ve Figmě se zcela neshoduje s výslednou podobou uživatelského rozhraní v aplikaci. Dodatečné změny se týkaly zejména problémů v backendové logice. Například se původně plánovalo, že lektor bude při vytváření kvízu moci zadat ručně vlastní heslo. Od této funkcionality se však nakonec na žádost backend týmu upustilo a ruční zadávání přístupového hesla bylo nahrazeno automaticky generovaným kódem. Jiné změny zase reagovaly na potřeby vývojářů mobilní aplikace.

Jelikož jsem přidával funkcionalitu do již existující aplikace, snažil jsem se dodržet vizuální konzistenci s již implementovanými komponentami, aby celé rozhraní působilo přehledně a jednotně. Používal jsem proto výhradně barvy a UI prvky, které již byly použity v jiných komponentách administrátorské části. Co se týče barevné palety, využíval jsem především tmavě modrou barvu, která je přítomna v hlavním panelu. Červenou barvu jsem využil výhradně k funkcionalitě, jako je mazání kvízů nebo otázek, a také k dalším akcím, jejichž předmětem je něco smazat či zrušit. Zelenou barvu jsem využil například k indikaci správných odpovědí u multichoice otázek.

Vizuální konzistence se nevztahuje pouze k barevné paletě, jednotnost jsem se snažil zachovat i při výběru UI prvků. Styl tlačítek použitých napříč modulem je identický, jako ve zbytku administrátorské části. Totéž platí o tabulkách a formulářích. Ikony byly rovněž pře-

vzaty z ostatních částí aplikace, využil jsem stejnou sadu.

Navigační panel jsem převzal z existující aplikace, v návrhu je napříč všemi obrazovkami stejný. Do hlavního menu jsem přidal novou položku *Kvízy*, která ve výsledné aplikaci obsahuje podsložky *Vytvořit kvíz* a *Přehled kvízů*. V návrhu jsem uvažoval s dalšími podsložkami, například *Kvízy v konceptu*, od implementace této funkcionality se však upustilo. Původně jsem pro modul plánoval vytvořit jen jednu podsložku *Moje kvízy*, pro přehlednost jsem nakonec funkcionality modulu rozdělil do dvou podsložek.

V následujících podkapitolách se detailněji věnuji jednotlivým obrazovkám a komponentám. Popíšu jejich návrh a porovnáám jejich vzhled s finálním stavem v aplikaci.

5.1 Tvorba kvízů

Tvorba kvízů probíhá na záložce *Vytvořit kvíz*. Tato záložka obsahuje dvě základní obrazovky – *Obecné informace* a *Otázky*. Obě tyto karty mají v záhlaví společné navigační menu, které ve finální aplikaci slouží pro přechod mezi těmito kartami. Navigační menu v návrhu obsahuje i položky *Body dovednosti* a *Akreditace*. Ty měly sloužit k nastavení kreditového systému. Tyto karty ve finální aplikaci v současnosti nejsou implementovány, bodování bude do modulu přidáno až po integraci s mobilní aplikací.

5.1.1 Záložka Obecné informace

The screenshot displays the 'Detail kvízu' (Quiz Details) form in the ROSTEM application. The interface features a dark blue sidebar on the left with the 'ROSTEM' logo at the top. The sidebar menu includes items like 'Nástěnka', 'Váš dostupné akce', 'Instituce', 'Moje instituce', 'Upravení instituce', 'Uživatelé', 'Studenti', 'Žebříčky studentů', 'Aktivity', 'Kvízy', and 'Vytvořit kvíz'. The 'Kvízy' section is expanded, showing 'Moje kvízy', 'Kvízy v konceptu', 'Kvízy instituce', 'Kvízy ke schválení', and 'Kvízy k akreditaci'. The main content area has a light gray header with 'Kvízy / Nový kvíz' and three buttons: 'Uložit koncept', 'Odeslat ke schválení', and 'Zadejte koncept'. Below the header is a tabbed interface with four tabs: 'Obecné informace' (selected), 'Otázky', 'Body dovednosti', and 'Akreditace'. The 'Obecné informace' tab contains the following fields: 'Název kvízu' (text input), 'Datum otevření' (date input), 'Datum uzavření' (date input), 'Popis kvízu' (text input), and a 'Heslo' section with two radio buttons: 'Nevyžadovat heslo' and 'Vyžadovat heslo' (selected). A 'Zadejte heslo' (password) input field is also present.

Obrázek 5.1 Návrh UI stránky Obecné informace

(zdroj: vlastní)

Po přechodu na stránku *Vytvořit kvíz* se uživateli jako první zobrazí záložka *Obecné informace*. Tato část obsahuje jednoduchý a přehledný formulář pro zadání základních údajů, jako je název kvízu, datum otevření a uzavření nebo popis. 5.1 Design formuláře je velmi podobný ostatním formulářům v administrátorské části aplikace, je tedy zajištěna konzistence vzhledu.

Návrh obsahuje možnost vyžadovat heslo pomocí radio buttonu. Uživatel vybírá mezi dvěma možnostmi:

- **Nevyžadovat heslo** – kvíz bude přístupný všem studentům.
- **Vyžadovat heslo** – kvíz bude přístupný jen studentům, kteří znají heslo.

Možnost výběru z těchto dvou variant zajišťuje flexibilitu při pořádání workshopů. Otevřené kvízy jsou přístupné i pro studenty, kteří se workshopu nemohou fyzicky zúčastnit, například kvůli zdravotním problémům. Na druhou stranu, při vyplňování kvízu z domova lze snadno podvádět. V budoucnu proto bude nutné přenastavit systém kreditů, aby za kvízy bez hesla studenti nedostávali kredity.

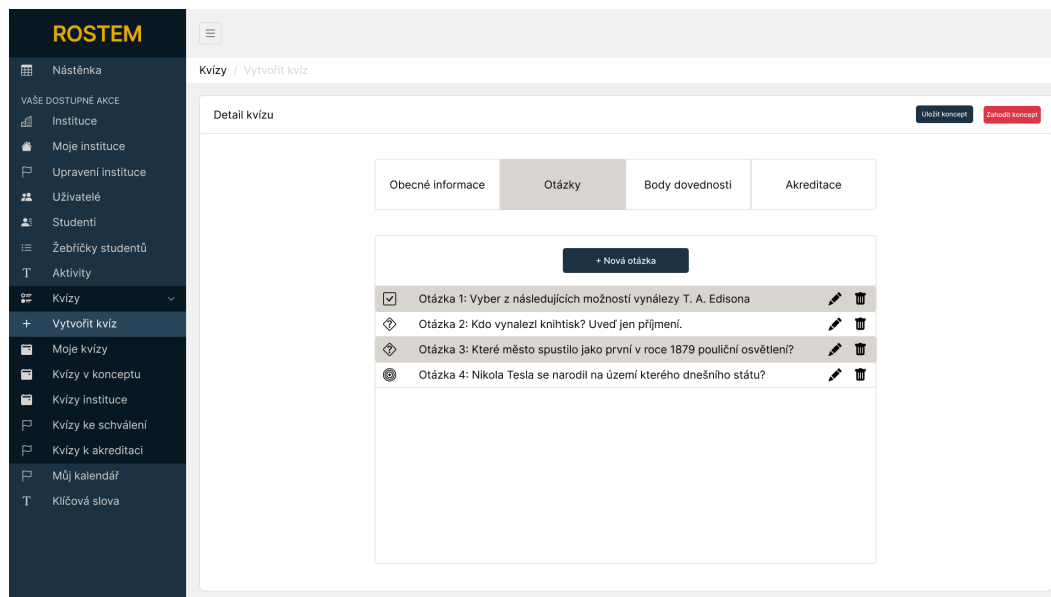
V návrhu se počítalo s tím, že pokud uživatel zvolí možnost *Vyžadovat heslo*, aktivuje se pole pro ruční zadání hesla. Ve výsledné aplikaci však došlo ke změně. Radio button a ruční zadávání hesla nahradil jediný checkbox *Generovat heslo*. Pokud uživatel checkbox nezaškrtně, kvíz bude otevřený a heslo nebude vyžadováno. V případě zaškrtnutí se automaticky vygeneruje přístupový kód, který je posléze dostupný v přehledu kvízů, aby jej lektor mohl sdílet se studenty. K této změně došlo na základě požadavku ze strany backend týmu.

V původní verzi návrhu se uvažovalo i o dalších UI prvcích, například o dropdown menu pro přiřazení kvízu k určitému workshopu, nebo o select menu pro výběr věkové kategorie studentů. Tyto možnosti byly během konzultací zamítnuty.

5.1.2 Záložka Otázky

Tato část slouží pro správu otázek v rámci kvízu. Záložka obsahuje dva základní pohledy:

- **Přehled otázek** – zobrazení seznamu všech otázek v daném kvízu. 5.2
- **Editor otázek** – formulář pro tvorbu nové otázky nebo editaci již existující otázky. 5.3



Obrázek 5.2 Návrh UI přehledu otázek

(zdroj: vlastní)

Přehled tvoří seznam všech otázek přidanych do kvízu. U každé otázky jsou zobrazeny dva ovládací prvky – tlačítko pro editaci, a tlačítko pro odstranění. Pokud v rámci kvízu lektor ještě žádnou otázku nepřidal, zobrazí se v seznamu jen informační text *Zatím zde nejsou žádné otázky*. Nové otázky lze přidávat pomocí tlačítka *+ Nová otázka* nad seznamem.

Tlačítka pro editaci a odstranění otázky jsou v návrhu označeny pouze ikonami. Ikona tužky indikuje editaci a ikona koše indikuje odstranění otázky. Ikony byly z důvodu přehlednosti ve výsledné aplikaci nahrazeny barevně rozlišenými tlačítkami s textem.

Další úpravou oproti návrhu bylo odebrání ikon indikujících typ otázky v seznamu. Tyto ikony byly vyhodnoceny jako nepřehledné, typ otázky navíc není nutné v seznamu zobrazit. Ve finální aplikaci proto není typ otázky v přehledu viditelný. Menší změny se týkají také ovládacích tlačítek v záhlaví. Tlačítka pro zahoezení konceptu a odeslání kvízu byla pro lepší přehlednost přesunuta pod navigační menu. Tlačítko *Uložit koncept* ve výsledné aplikaci neexistuje, jelikož funkcionality ukládání konceptů nakonec nebyla implementována.

Editor otázek je tvořen formulářem. První položkou je pole pro zadání otázky, následuje menu pro výběr typu otázky. V návrhu se počítalo se třemi typy otázek/odpovědí:

- **Otevřená odpověď** – zadání správné odpovědi jako text.
- **Výběr: 1 správná odpověď** – výběr správné odpovědi z vícero možností, přičemž jen

jedna odpověď je správná.

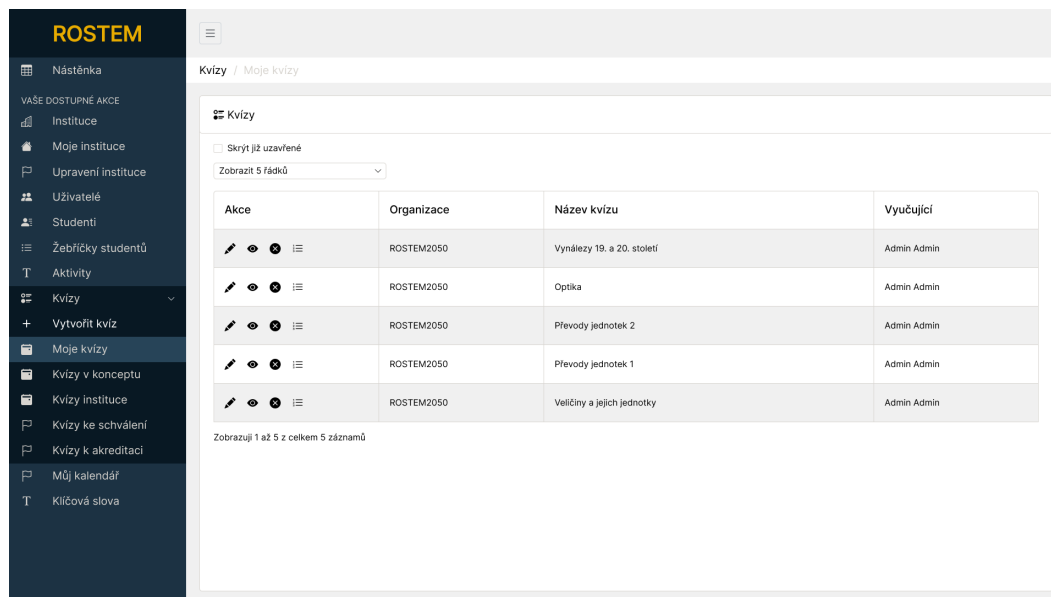
- **Výběr: více správných** – výběr z více možností, přičemž správných odpovědí může být více.

Zbytek formuláře se liší v závislosti na zvoleném typu otázky. Možnost *Otevřená odpověď* obsahuje jen jedno pole pro zadání správné odpovědi. U výběrových otázek je nejprve k dispozici dropdown, ve kterém uživatel může zvolit počet možností. Počet textových polí pro zadání jednotlivých možností se pak mění v závislosti na počtu zvoleném v dropdown menu. Výběr s jednou správnou odpovědí dále obsahuje dropdown menu pro výběr správné odpovědi, zatímco výběr s více správnými odpověďmi používá checkboxy pro volbu správných možností.

Obrázek 5.3 Návrh UI editoru otázek

(zdroj: vlastní)

Funkcionalita se však ve výsledné aplikaci značně liší. Po analýze bylo rozhodnuto obě varianty multichoice sjednotit. Volba jen jedné správné možnosti u otázky typu *Výběr: více správných* je totiž z praktického hlediska stejná, jako otázka typu *Výběr: 1 správná odpověď*. Ve výsledné aplikaci jsou proto k dispozici jen dva typy odpovědí: *Otevřená* a *Uzavřená (multichoice)*. Finální aplikace na rozdíl od návrhu také disponuje polem pro zadání časového limitu.



Obrázek 5.4 Návrh UI přehledu kvízů

(zdroj: vlastní)

5.2 Přehled kvízů

Na stránce přehledu je zobrazena tabulka obsahující všechny kvízy vytvořené uživatelem.

5.4 Původní návrh počítal se čtyřmi sloupci:

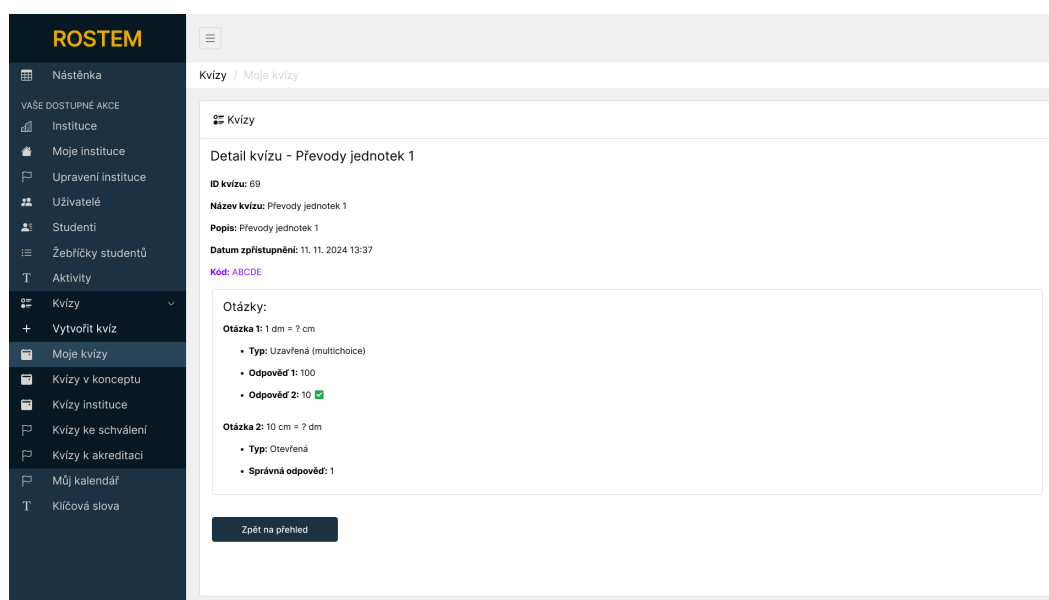
- **Akce** – v tomto sloupci jsou zobrazena akční tlačítka pro manipulaci s jednotlivými kvízy. Tlačítko s ikonou tužky uživatele přesměruje na editaci kvízu, tlačítko s ikonou oka na stránku s detaily kvízu. Tlačítko s ikonou křížku slouží ke smazání daného kvízu a tlačítko označené ikonou seznamu slouží ke zobrazení žebříčků studentů.
- **Organizace** – název organizace.
- **Název kvízu**
- **Vyučující** – jméno a příjmení vyučujícího.

Při implementaci došlo k nahrazení některých sloupců. Vzhledem k tomu, že si daný lektor může zobrazit pouze své vlastní kvízy, je sloupec *Vyučující* nadbytečný. Při tvorbě návrhu jsem navíc nevěděl, že vyučující může být členem pouze jedné organizace. Sloupec *Organizace* je tudíž také redundantní.

Položka *Organizace* byla nahrazena položkou *ID kvízu*. Sloupec *Vyučující* byl nahrazen sloupcem *Kód*, ve kterém je zobrazen přístupový kód kvízu. V případě otevřeného kvízu zůstává tento sloupec prázdný.

Menší rozdíl oproti návrhu se týká také akčních tlačítek, pro lepší přehlednost byla barevně rozlišena. Ikona křížku byla nahrazena ikonou koše, která je pro označení funkcionality mazání vhodnější.

5.3 Detail kvízu



Obrázek 5.5 Návrh UI detailu kvízu

(zdroj: vlastní)

Na stránku s detailem kvízu se uživatel dostane po kliknutí na ikonu oka v přehledu kvízů. Detail kvízu zobrazuje podrobné informace o kvízu, jako ID, název, popis, přístupový kód a informace o jednotlivých otázkách. Kód je na stránce barevně odlišen, aby byl pro lektora snadno dohledatelný. Informace o otázce jsou zobrazeny v oddělené sekci a zahrnují pořadí v rámci kvízu, zadání, typ, a jednotlivé odpovědi. U multichoice otázek jsou správné možnosti označeny zelenou ikonou. 5.5

V detailu kvízu nedošlo při implementaci ve srovnání s návrhem k žádným zásadním změnám.

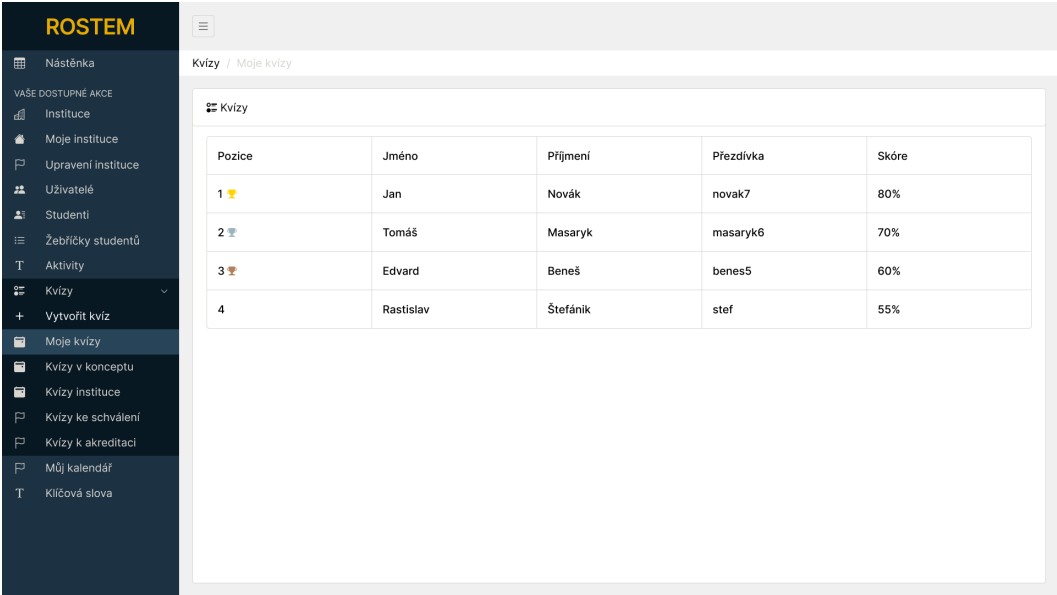
5.4 Editace kvízu

Na stránku s editací se uživatel dostane kliknutím na tlačítko s ikonou tužky v přehledu. Editace probíhá prostřednictvím formuláře, kde uživatel může měnit název a popis kvízu, a informace o jednotlivých otázkách. V úpravách otázek je zahrnuta změna typu otázky, nebo změna počtu možností u otázek typu multichoice. Lektor také může odebírat již existující otázky a přidávat otázky nové.

Ve výsledné aplikaci nedošlo na této stránce k žádným změnám oproti návrhu. Zpětně však hodnotím, že design stránky s editací kvízu mohl být sjednocen se vzhledem stránky pro tvorbu kvízů.

5.5 Žebříček skóre

Na stránku se zobrazením skóre studentů se uživatel dostane pomocí tlačítka s ikonou seznamu na přehledu kvízů. Žebříček je tvořen tabulkou obsahující pozici, jméno, příjmení a přezdívkou studenta. V posledním sloupci je zobrazeno procentuální skóre. V návrhu jsou u prvních tří pozic ikony pohárů, ve výsledné aplikaci jsou místo toho první tři řádky barevně zvýrazněny. 5.6



Pozice	Jméno	Příjmení	Přezdívka	Skóre
1 🏆	Jan	Novák	novak7	80%
2 🥈	Tomáš	Masaryk	masaryk6	70%
3 🥉	Edvard	Beneš	benes5	60%
4	Rastislav	Štefánik	stef	55%

Obrázek 5.6 Návrh UI žebříčku

(zdroj: vlastní)

6 INTEGRACE MODULU S OSTATNÍMI ČÁSTMI APLIKACE

Pro zajištění správné funkcionality kvízového modulu je potřeba, aby modul komunikoval s:

- **backendem**, jelikož je třeba ukládat informace o kvízech.
- **frontendem**, aby se kvízy vytvořené prostřednictvím modulu mohly zobrazit v mobilním a webovém rozhraní, a aby bylo možné zobrazit skóre studentů v administrátorském prostředí.

Backend je realizován prostřednictvím Laravel kontrolerů a databáze pro ukládání dat. Frontend je implementován jako samostatná PWA aplikace. Jelikož integrace s frontendem dosud nebyla dokončena, budu se nadále věnovat integraci s backendem.

Komunikace mezi modulem a backendem probíhá prostřednictvím API požadavků. Klíčovou roli hraje soubor *functions.js*, který obsahuje veškeré funkce zajišťující komunikaci s backendem. Tyto funkce implementují HTTP metody GET, POST, PATCH a DELETE. Funkce jsou následně importovány do ostatních komponent:

```
1 import functions from './functions.js';
```

Výpis 6.1 Import funkcí (zdroj: vlastní)

Funkce definované v souboru *functions.js* jsou volány v následujících situacích:

- **Při ukládání kvízu.** Pro ukládání kvízů je využívána funkce *submitQuiz*, která implementuje volání metody POST.
- **Při zobrazení přehledu všech kvízů nebo detailů o konkrétním kvízu.** K načtení informací o kvízech slouží funkce *getQuiz* implementující volání metody GET.
- **Při ukládání změn v kvízu.** Ukládání změn je realizováno prostřednictvím funkce *updateQuiz*, v níž je implementována metoda PATCH.
- **Při mazání kvízu.** Funkcionality mazání je dosaženo pomocí funkce *deleteQuiz* implementující metodu DELETE.

Všechny funkce ze souboru *functions.js* jsou exportovány, aby mohly být využity v ostatních komponentách:

```
1 export default {  
2   submitQuiz,  
3   updateQuiz,  
4   getQuiz,  
5   deleteQuiz,  
6 };
```

Výpis 6.2 Export funkcí (zdroj: vlastní)

6.1 Funkce submitQuiz

Tato funkce slouží k odeslání nově vytvořeného kvízu na backend pomocí HTTP metody POST. Funkce přijímá dva parametry:

- **submitUrl** – relativní cesta k API endpointu.
- **data** – objekt s daty kvízu.

Při úspěšném odeslání se uživateli zobrazí zelené potvrzovací oznámení, v případě selhání odeslání je naopak zobrazeno červené chybové upozornění. Oznámení jsou implementována pomocí knihovny *toastr*. Logování v blocích *catch* a *finally* je určeno k debugování a při nasazení bude odstraněno.

```
1 async function submitQuiz(submitUrl, data) {  
2   try {  
3     await request('POST', submitUrl, data);  
4     toastr.success('Kvíz byl úspěšně odeslán!');  
5   } catch (e) {  
6     console.error("Požadavek POST selhal: ", e);  
7     toastr.error("Požadavek POST selhal!");  
8   } finally {  
9     console.log(JSON.stringify(data));  
10  }  
11 }
```

Výpis 6.3 Funkce submitQuiz (zdroj: vlastní)

Funkce je implementována jako asynchronní (async/await), aby se při komunikaci se serverem nezablokovalo hlavní vlákno aplikace. Uživatelské rozhraní díky tomu zůstává responzivní.

Do backendu se odešle JSON payload. Níže je uveden příklad odesílaného objektu:

```
1 {  
2   "quizName": "Převody jednotek",  
3   "dateOpen": "2025-05-26",
```

```
4      "dateClose": "2025-05-30",
5      "description": "Jednoduché převody pro studenty.",
6      "open": false,
7      "questions": [
8          {
9              "text": "1 dm = ? cm",
10             "limit": null,
11             "type": "open",
12             "correctAnswer": "10",
13             "options": []
14         }
15     ]
16 }
```

Výpis 6.4 Payload při odeslání kvízu

V uvedeném příkladu je pole *options* prázdné, jelikož se jedná o otevřený typ otázky. Po úspěšném odeslání server vrátí odpověď s HTTP status kódem 200 OK. V těle odpovědi se nachází objekt obsahující veškeré informace o vytvořeném kvízu, včetně přiřazeného ID.

6.2 Funkce `updateQuiz`

Tato funkce slouží k odeslání aktualizovaných dat o kvízu na backend pomocí HTTP metody PATCH. Funkce přijímá parametry:

- **quizId** – identifikátor kvízu, který má být aktualizován. Tento identifikátor je vložen do URL API endpointu.
- **data** – objekt obsahující aktualizovaná data kvízu.

Stejně jako ve funkci *submitQuiz* jsou i zde použity hlášky z knihovny *toastr*. Funkce *updateQuiz* je také implementována jako asynchronní, což zajišťuje neblokující komunikaci se serverem.

6.3 Funkce `getQuiz`

Funkce *getQuiz* slouží k načtení informací o kvízu z databáze. Funkce pracuje pouze s jedním parametrem *getUrl*, což je cesta k API endpointu, ze kterého jsou informace o kvízu načteny. Lze získat informace o všech kvízech (zasláním dotazu na `/api/admin/quizzes`), nebo jen o jednom kvízu (pokud je v URL za `quizzes` uvedeno i ID kvízu). Pro načtení informací o kvízech je využita HTTP metoda GET. Implementace notifikací a asynchronní charakter jsou aplikovány i v této funkci.

```

1 async function getQuiz(getUrl) {
2   try {
3     const response = await request('GET', getUrl);
4     console.log("API odpověď: ", response);
5     return response;
6   } catch (e) {
7     console.error("Požadavek GET selhal: ", e);
8     toastr.error("Požadavek GET selhal!");
9     return null;
10  }
11 }

```

Výpis 6.5 Funkce getQuiz

6.4 Funkce deleteQuiz

Poslední funkce definovaná v souboru functions.js slouží k odstranění kvízu z databáze, a to prostřednictvím metody DELETE. Kromě identifikátoru kvízu je přijímána také funkce updateQuizList, která se po úspěšném smazání postará o odstranění daného kvízu z lokálního seznamu filtrováním pole *quizData*. Funkce přijímá následující parametry:

- **quizId** – identifikátor kvízu, který má být smazán.
- **updateQuizList** – funkce zajišťující smazání kvízu ze seznamu.

Stejně jako u dalších funkcí i zde nechybí toastr upozornění pro notifikování uživatele o úspěšném či neúspěšném smazání.

```

1 async function deleteQuiz(quizId, updateQuizList) {
2   try {
3     await request('DELETE', '/api/admin/quizzes/${quizId}');
4     toastr.success("Kvíz byl úspěšně smazán!");
5     if (updateQuizList) {
6       updateQuizList(quizId);
7     }
8   } catch (e) {
9     console.error("Požadavek DELETE selhal:", e); //
10    debug
11    toastr.error("Požadavek DELETE selhal!");
12  }

```

Výpis 6.6 Funkce deleteQuiz

Při úspěšném smazání kvízu s daným ID bude tělo odpovědi obsahovat *delete: true*, a následně data smazaného kvízu. V komponentě *ViewQuiz* je následně implementována funkce *handleQuiz*:

```
1 async function handleDelete(quizId) {  
2   await functions.deleteQuiz(quizId, (deletedQuizId) => {  
3     quizData.value = quizData.value.filter(quiz => quiz.  
4       id !== deletedQuizId);  
5   });  
}
```

Výpis 6.7 Funkce handleDelete

Tato funkce je zavolána poté, co uživatel na přehledu kvízů klikne na tlačítko pro smazání kvízu. Tím je zajištěno, že je kvíz z tabulky odebrán ihned po odstranění z databáze.

7 IMPLEMENTACE HLAVNÍCH KOMPONENT

Modul kvízů se skládá ze dvou stěžejních komponent. Jedná se o komponenty:

- **CreateQuiz** – tato komponenta obsahuje veškerou logiku týkající se vytváření kvízů a přidávání otázek do kvízů.
- **ViewQuiz** – komponenta s logikou pro zobrazení seznamu kvízů, zobrazení detailů o kvízech, editaci a mazání kvízů.

Tyto dvě komponenty zajišťují chod modulu. V následujících kapitolách detailně popíšu funkce obsažené v těchto komponentách.

7.1 Komponenta CreateQuiz

Komponenta *CreateQuiz* je hlavním rozhraním pro tvorbu kvízů. Slouží k zadávání veškerých informací o kvízu (název, popis, datum otevření nebo uzavření), přidávání otázek různých typů, a k odesílání kvízů do databáze. Je zde také implementována logika pro validaci vstupů. Klíčovou roli v komponentě hrají funkce *newQuestion*, jež slouží pro přidávání, editaci a validaci jednotlivých otázek, a *newQuiz*, která slouží pro ukládání a validaci celých kvízů.

7.1.1 Navigace v komponentě

V záhlaví komponenty se nachází navigační menu se stránkami *Obecné informace* a *Otázky*. Stránka *Obecné informace* je zvolena jako výchozí a nachází se na ní formulář, kde uživatel zadává základní informace o kvízu, jako název či popis. Na stránce *Otázky* pak uživatel může přidávat, editovat a odstraňovat otázky. Stránka *Otázky* obsahuje dva pohledy – pohled se seznamem otázek a pohled editoru otázek. Výchozím zobrazením je seznam otázek, na pohled s tvorbou či editací lze přistupovat prostřednictvím tlačítek + *Nová otázka* nebo *Editovat*.

Přepínání mezi záložkami *Obecné informace* a *Otázky* je implementováno pomocí proměnné *activeTab* a funkce *setActiveTab*:

```
1 const activeTab = ref('obecne-informace');  
2  
3 function setActiveTab(tab) {  
4     activeTab.value = tab;  
5 }
```

Výpis 7.1 Implementace přepínání záložek

Podobným způsobem funguje i přepínání typu otázky, k tomu je určena proměnná *selectedType*. Přepínání mezi přehledem otázek a editorem otázek na záložce *Otázky* je implementováno prostřednictvím proměnné *showNewQuestion*. Výchozí hodnota této proměnné je *false*, což znamená, že výchozím zobrazením je přehled otázek. Po kliknutí na tlačítko pro přidání nové otázky nebo na tlačítko pro editaci otázky je zavolána funkce, která hodnotu *showNewQuestion* nastaví na *true*, tudíž se zobrazí formulář pro tvorbu či editaci otázky. Naopak, pokud se uživatel nachází na kartě s tvorbou otázky a klikne na tlačítko *Uložit otázku* nebo *Zahodit otázku*, zavolá se funkce, která hodnotu *showNewQuestion* změní na *false*, a uživatel se tak vrátí zpět na pohled s přehledem.

7.1.2 Správa otázek

Nejprve se zaměřím na logiku vytváření otázek. Vytváření otázek zajišťuje komplexní funkce *newQuestion*. Tato funkce je zavolána poté, co uživatel klikne na tlačítko *Uložit otázku* při vytváření nové otázky, případně na tlačítko *Uložit změny* při editaci již existující otázky. Funkce je tedy využívána jak k ukládání dat nové otázky, tak k aktualizaci dat již existující otázky.

Funkce využívá validační atribut, proměnnou *isValid*. Před uložením otázky do Local Storage proběhne validace polí. Pokud některé z požadovaných polí uživatel nevyplnil, bude na tuto skutečnost upozorněn. Hodnota *isValid* se změní na *false*, což zapříčiní, že otázka nebude odeslána do Local Storage a uživateli se zobrazí toastr upozornění s informací, které pole je vyplněno chybně. Ohraničení nesprávně vyplněných polí se navíc zbarví do červena.

První je kontrolováno zadání otázky, zda není prázdné. V případě otevřené otázky se kontroluje text správné odpovědi, u uzavřené multichoice otázky je nutné zkontrolovat všechny možnosti. Ostatní pole nejsou vyžadována, tudíž u nich není třeba provádět validaci.

Pokud je validace úspěšná, otázka se uloží. Nejprve se uloží zadání, limit a typ otázky. Hodnota správné odpovědi *correctAnswer* a pole s možnostmi *options* v první fázi zůstávají nevyplněny. Zda budou nabývat nějakou hodnotu závisí na typu otázky. Pokud je otázka otevřená, pak pole *options* zůstane prázdné a do *correctAnswer* se uloží správná odpověď.

Naopak v uzavřených otázkách bude *correctAnswer* nabývat hodnotu *null* a v poli *options* budou uloženy jednotlivé možnosti. Položky v tomto poli mají také přiřazenu hodnotu *isCorrect*, která udává, zda je možnost správná či nikoliv.

```

1 [
2   {
3     "text": "Otevřená otázka",
4     "limit": null,
5     "type": "open",
6     "correctAnswer": "Otevřená odpověď",
7     "options": []
8   },
9   {
10    "text": "Uzavřená otázka (3 možnosti)",
11    "limit": null,
12    "type": "multi",
13    "correctAnswer": null,
14    "options": [
15      {
16        "text": "Možnost 1 správně",
17        "isCorrect": true
18      },
19      {
20        "text": "Možnost 2 špatně",
21        "isCorrect": false
22      },
23      {
24        "text": "Možnost 3 správně",
25        "isCorrect": true
26      }
27    ]
28  }
29 ]

```

Výpis 7.2 Příklad odeslaného kvízu (pole questions)

Uvedený příklad je obsah pole *questions* po odeslání kvízu do databáze. Kvíz v příkladu má první otázku otevřenou ("type": "open"), proto je pole *options* prázdné, a *correctAnswer* obsahuje správnou odpověď. Druhá otázka je typu multichoice ("type": "multi"), *correctAnswer* je proto *null*, a pole *options* obsahuje jednotlivé možnosti.

Multichoice otázka ve výše uvedeném příkladu obsahuje tři možnosti. Možností u multichoice může být 2 až 9, počet možností lze zvolit pomocí dropdown menu. Změny jsou pak zpracovány funkcí *updateNumberOfInputs*, která zajistí, že je zobrazen správný počet inputů pro hodnotu zvolenou v dropdown menu.

V backendu byla implementována funkcionalita, která před uložením převede text správné odpovědi u otevřené otázky na malá písmena a odstraní z něj diakritiku a speciální znaky. V příkladu výše jsem v první otázce zadal správné znění odpovědi "correctAnswer": "Otevřená odpověď", v odpovědi backendu je však "text": "otevrena odpoved". Normalizace textu eliminuje problémy, kdy student odpoví na zadanou otázku správně, avšak bez diakritiky.

Otázka je následně uložena do Local Storage. Při ukládání je důležitá proměnná *isEditing*, jež hodnota udává, zda uživatel vytváří novou, nebo upravuje již existující otázku. Pokud je hodnota *isEditing* nastavena na *false*, znamená to, že uživatel vytváří novou otázku, a ukládání proto probíhá pomocí Local Storage funkce *addQuestion*. Hodnota *isEditing* *true* značí, že uživatel do editoru přistoupil prostřednictvím tlačítka *Editovat*, tedy že nevytváří otázku novou, jen upravuje existující. Při aktualizaci již existující otázky je volána Local Storage funkce *editQuestion*.

```
1         if (isEditing.value && editingIndex.value !== null) {
2             quizStore.updateQuestion(editingIndex.value,
3                                     newQuestion);
4             isEditing.value = false;
5             editingIndex.value = null;
6         } else {
7             quizStore.addQuestion(newQuestion);
        }
```

Výpis 7.3 Ukládání otázky v rámci funkce newQuestion

Po uložení nového kvízu či aktualizaci existujícího kvízu je zavolána funkce *resetFormFields*. Úkolem této funkce je vyčistit všechna pole dokončení vytváření nebo editace.

Pokud se během vytváření či editování otázky lektor rozhodne, že chce svůj dosavadní postup zahodit, může tak učinit pomocí tlačítka *Zahodit otázku*. Toto tlačítko je napojeno na funkci *discardQuestion*, která resetuje všechna pole.

Existující odpovědi je možné jednoduše smazat prostřednictvím tlačítka *Odstranit* v seznamu kvízů. O odstranění se stará funkce *deleteQuestion*, která jako parametr přijímá index otázky, jež má být odstraněna. Metoda *splice* zajistí po smazání otázky správné pořadí ostatních otázek.

7.1.3 Ukládání kvízů

Logika odesílání kvízů do databáze je implementována ve funkci *newQuiz*. Podobně jako při ukládání jednotlivých otázek je i při ukládání celého kvízu prováděna validace pomocí proměnné *isValid*. V tomto případě je kontrolováno, zda je vyplněný název kvízu, a také zda kvíz obsahuje alespoň jednu otázku. Pokud není některá z těchto podmínek splněna, kvíz není odeslán do backendu a uživatel je na chybu upozorněn prostřednictvím toastru.

Pokud lektor při vytváření kvízu zaškrtl pole *Generovat heslo*, bude proměnná *isOpen*

nabývat hodnotu false. To znamená, že kvíz nebude otevřený, pro přístup bude vyžadováno heslo. Naopak pokud při vytváření kvízu možnost *Generovat heslo* uživatel nezvolí, kvíz bude volně přístupný všem. Heslo lektor nezadává manuálně, na backend se odešle pouze informace o tom, zda má být kvíz otevřený nebo uzavřený. Pokud bude hodnota open nastavena na false, pak backend vygeneruje přístupový kód. Tento kód si lektor následně bude moci zobrazit na stránce *Přehled kvízů*.

Po úspěšném odeslání kvízu do databáze je nakonec volána funkce *clearQuiz*, která se stará o reset všech polí a otázek.

7.2 Komponenta ViewQuiz

Komponenta *ViewQuiz.vue* slouží pro zobrazení a správu kvízů. Základem komponenty je tabulka s přehledem všech kvízů uživatele. Tabulka obsahuje následující sloupce: *Akce*, *ID kvízu*, *Název kvízu* a *Kód*. Sloupec *Akce* obsahuje akční tlačítka, sloužící k přesměrování mezi jednotlivými pohledy komponenty. První zelené tlačítko s ikonou oka přesměruje uživatele na detaily kvízu, kde si uživatel bude moci zobrazit všechny informace o daném kvízu. Druhé modré tlačítko s ikonou tužky přesměruje uživatele na záložku s editací kvízu, kde je uživateli umožněno editovat již existující kvíz. Třetí tlačítko označené ikonou seznamu slouží pro zobrazení žebříčků studentů a poslední červené tlačítko s ikonou koše slouží pro smazání daného kvízu.

Po přístupu ke komponentě *VueQuiz* jsou načteny informace o všech kvízech. Toho je dosaženo pomocí funkce *fetchQuizzes*:

```
1 async function fetchQuizzes() {  
2   quizData.value = await functions.getQuiz('http://  
3     localhost/api/admin/quizzes');  
4   if (!quizData.value || quizData.value.success === false)  
5     {  
6       return;  
7   }  
}
```

Výpis 7.4 Funkce fetchQuizzes

Funkce *fetchQuizzes* je spouštěna bezprostředně po inicializaci komponenty. Tím je zajištěno, že data o kvízech se v tabulce zobrazí bez toho, aniž by uživatel musel cokoli dělat.

```
1 onMounted(async () => {
```

```
2     console.log("mounted");  
3     fetchQuizzes();  
4 });
```

Výpis 7.5 Volání funkce fetchQuizzes při inicializaci komponenty

7.2.1 Správa kvízů

Funkcionalita zobrazení detailních informací o kvízu i editace kvízu je zajištěna pomocí funkce *fetchQuizDetails*. Funkce přijímá 2 parametry:

- **quizId** – ID kvízu, jež chceme editovat, nebo o němž chceme získat informace,
- **editMode** – boolean, udává, zda jsme v editačním módu či nikoliv.

Funkce nejprve načte data o kvízu s určitým ID z databáze, což pro zobrazení detailů o kvízu stačí. Zbytek funkce je prováděn pouze za předpokladu, že je proměnná *editMode* nastavena na true, tedy je aktivní režim úprav. K tomu dojde v případě, kdy uživatel klikne na tlačítko pro editaci kvízu. Do reaktivní proměnné *editingQuiz* se uloží aktuální data kvízu přicházející z API, aby se nimi dalo dále pracovat v editačním formuláři.

Probíhá také kontrola, zda jsou všechna pole správně vyplněna. K situaci, kdy nějaké pole nebude vyplněno, by díky validaci při ukládání nemělo dojít, nicméně jsem ošetřil i tuto možnost. Pokud chybí hodnota časového limitu (timeru), je automaticky nastavena na hodnotu 300, což odpovídá výchozí hodnotě přiřazované backendem otázkám bez časového omezení. Také se nastaví typ otázky podle hodnoty z backendu. Hodnota 2 označuje multichoice otázku, zatímco hodnota 3 indikuje otázku otevřenou. Hodnota 1 byla původně vyhrazena pro multichoice s jednou správnou odpovědí.

Během editace může uživatel také měnit počet otázek v kvízu, nebo typ jednotlivých otázek. Tuto funkcionalitu zajišťují funkce *handleQuestionTypeChange* a *updateAnswerCount*. Funkce *addQuestion* do editovaného kvízu přidá novou otázku. Funkce *removeQuestion* zajistí, že je otázka s určitým indexem odstraněna, a že se po odstranění indexy ostatních otázek správně změní.

Poté, co uživatel provede změny, může je uložit prostřednictvím tlačítka *Uložit změny*. Ukládání aktualizovaného kvízu probíhá pomocí funkce *saveQuizChanges*. Před odesláním aktualizovaného kvízu do databáze je i zde provedena validace, uživatel nemůže odeslat

kvíz bez otázek či bez názvu. Ukládání aktualizovaných dat do databáze je implementováno pomocí funkce `updateQuiz` z Local Storage.

7.2.2 Žebříčky studentů

Na pohled s žebříčkem studentů se přistupuje pomocí fialového tlačítka s ikonou seznamu. V tomto pohledu je zobrazen přehled s položkami pozice, jméno, příjmení, přezdívk, na závěr je uvedeno dosažené skóre v procentech. První tři pozice jsou barevně označeny. Funkcionalita zobrazení žebříčků v momentální chvíli není dostupná, v tabulce jsou zobrazeny pouze testovací data.

7.3 Local storage

Při ukládání otázek jsou otázky přidány do Local Storage. Následně jsou všechny otázky hromadně odeslány do databáze při odeslání celého kvízu. Lokální ukládání má oproti ukládání do databáze řadu výhod. Data jsou okamžitě dostupná, bez nutnosti komunikace se serverem, což zajišťuje rychlejší odezvu. Ukládání do Local Storage je navíc jednodušší na implementaci a snižuje vytížení databáze.

Funkcionalita Local Storage je obsažena v souboru *QuizStore.js*. Přidávání otázek do Local Storage probíhá pomocí akce *addQuestion*. Funkce přijímá data otázky a pomocí metody `push` je přidá do databáze. K aktualizaci dat v Local Storage funguje akce *updateQuestion*. Ta přijímá aktualizovaná data a index aktualizované otázky.

```
1 addQuestion(question) {  
2   console.log("Přidávám otázku:", question);  
3   this.questions.push(question);  
4 },  
5  
6  
7 updateQuestion(index, updatedQuestion) {  
8   console.log("Aktualizuji otázku s indexem:", index, "  
9     aktualizovaná data:", updatedQuestion);  
10  this.questions[index] = updatedQuestion;  
11 },
```

Výpis 7.6 Funkce pro manipulaci s otázkami v Local Storage

ZÁVĚR

Bakalářská práce se věnovala vývoji modulu kvízů pro online vzdělávací platformu ROSTEM 2050. Teoretická část čtenáře seznámila s projektem ROSTEM a s důvody pro přidání kvízového modulu. Následně stručně představila technologie použité při vývoji modulu. Praktická část práce byla zaměřena na analýzu požadavků, implementaci hlavních komponent a jejich integraci s backendem.

Hlavním cílem bylo vytvořit nástroj, který by zpestřil workshopy pro studenty a poskytl lektorům možnost efektivně ověřit, co si studenti z workshopů odnesli. Výsledný modul v současnosti umožňuje vytváření a úpravu kvízů, správu otázek a validaci vstupů. Modul byl rovněž úspěšně integrován s backendem. Ačkoli se podařilo naplnit hlavní funkcionalitu modulu, chybí zde napojení na frontend, které bude třeba v budoucnu doplnit.

V rámci projektu jsem získal první zkušenosti s vývojem větší aplikace ve frameworku Vue.js. Zpočátku pro mne bylo náročné zorientovat se ve struktuře projektu a pochopit propojení jednotlivých částí. Největší výzvou byla integrace modulu s backendem, během níž jsme s kolegou řešili mnoho technických problémů, například problém s generováním hesel pro nové kvízy po smazání posledního kvízu nebo nesprávné ukládání dat do databáze.

Přínosem této práce je modul kvízů pro aplikaci ROSTEM 2050, který může být dále vyvíjen a rozšiřován. Práce zároveň měla přínos i pro mě osobně, jelikož jsem se naučil pracovat s nástroji jako Docker nebo Vue.js a osvojil jsem si principy webového vývoje v rámci týmu.

SEZNAM POUŽITÉ LITERATURY

- [1] MGR. VLADIMÍR DOSTÁLEK, 2023. *Zlínský kraj spustil unikátní aplikaci pro podporu polytechnického vzdělávání - ROSTEM 2050*. Online. Dostupné z: <https://zlinskykraj.cz/aktuality/zlinsky-kraj-spustil-unikatni-aplikaci-pro-podporu-polytechnickeho-vzdelavani-rostem-2050-6467>. [cit. 2025-05-29].
- [2] LOOYESTYN, Jemma; KERNOT, Jocelyn; BOSHOF, Kobie; RYAN, Jillian; EDNEY, Sarah et al., 2017. Does gamification increase engagement with online programs? A systematic review. Online. *PLOS ONE*. 2017. 12, č. 3, s. 1–19. Dostupné z: <https://doi.org/10.1371/journal.pone.0173403>. [cit. 2025-04-28].
- [3] GIT SCM, [b.r.]. *Git icon (PNG)*. Online. Dostupné z: <https://git-scm.com/images/logos/downloads/Git-Icon-1788C.png>. [cit. 2025-05-29].
- [4] GIT SCM, [b.r.]. *Git - A Short History of Git*. Online. Dostupné z: <https://git-scm.com/book/ms/v2/Getting-Started-A-Short-History-of-Git>. [cit. 2025-05-06].
- [5] STACK OVERFLOW, 2022. *Stack Overflow Developer Survey 2022*. Online. Dostupné z: <https://survey.stackoverflow.co/2022/#version-control-version-control-system-prof>. [cit. 2025-05-06].
- [6] GIT SCM, [b.r.]. *Git - What is Git?* Online. Dostupné z: <https://git-scm.com/book/ms/v2/Getting-Started-What-is-Git%3F>. [cit. 2025-05-06].
- [7] GITHUB, 2025. *About GitHub*. Online. Dostupné z: <https://github.com/about>. [cit. 2025-05-07].
- [8] GITHUB, 2025. *Fork a repository - GitHub Docs*. Online. Dostupné z: <https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/working-with-forks/fork-a-repo>. [cit. 2025-05-07].
- [9] MICROSOFT, 2025. *What is Windows Subsystem for Linux*. Online. Dostupné z: <https://learn.microsoft.com/en-us/windows/wsl/about>. [cit. 2025-05-25].
- [10] DOCKER INC., [b.r.]. *Logo, Icon, and Brand Guidelines*. Online. Dostupné z: <https://www.docker.com/company/newsroom/media-resources/>. [cit. 2025-05-29].

- [11] DOCKER, INC., [b.r.]. *What is a Container?* Online. Dostupné z: <https://www.docker.com/resources/what-container/>. [cit. 2025-05-16].
- [12] PAVEL KOŠAN, [b.r.]. *Docker jako významná složka moderní datové infrastruktury*. Online. Dostupné z: <https://digitalniarchitekti.cz/clanek/docker/>. [cit. 2025-05-16].
- [13] MARCEL SCHMIED, 2024. *Performance-Comparison of VMs and Docker Containers - QF-Test*. Online. Dostupné z: <https://www.qftest.com/en/blog/article/performance-comparison-of-vm-and-docker-containers.html>. [cit. 2025-05-28].
- [14] ROBBIE GONZALEZ, 2017. *Figma Wants Designers to Collaborate Google-Docs Style*. Online. Dostupné z: <https://www.wired.com/story/figma-updates/>. [cit. 2025-05-28].
- [15] FLANAGAN, David, 2024. *JavaScript: The Definitive Guide : Master the World's Most-used Programming Language*. No Starch Press. 1. vyd. ISBN 978-1-491-95202-3.
- [16] HAVERBEKE, Marijn, 2024. *Eloquent JavaScript, 4th Edition*. No Starch Press. 4. vyd. ISBN 978-1-718-50410-3.
- [17] STACK OVERFLOW, 2024. *2024 Stack Overflow Developer Survey*. Online. Dostupné z: <https://survey.stackoverflow.co/2024/technology#most-popular-technologies-language-prof>. [cit. 2025-05-29].
- [18] VUE.JS, 2016. *Vue.js logo*. Online. Dostupné z: <https://github.com/vuejs/art/blob/9f8ee218d32b52c2d150212f22798a7a61689361/logo.png>. [cit. 2025-05-27].
- [19] STATE OF JS, 2024. *State of JavaScript 2024: Front-end Frameworks*. Online. Dostupné z: <https://2024.stateofjs.com/so-S0/libraries/front-end-frameworks/>. [cit. 2025-05-29].
- [20] GRZYBOWSKA, Kaja, [b.r.]. *Top 15 Inspiring Companies Using Vue.js in 2024*. Online. Dostupné z: <https://www.monterail.com/blog/top-companies-using-vue-js>. [cit. 2025-05-29].
- [21] YANG, HcySun, 2023. *Vue.JS Framework: Design and Implementation*. Springer Nature Singapore. 1. vyd. ISBN 978-9-819-94947-2.

- [22] SARRION, Eric, 2022. *JavaScript from Frontend to Backend: Learn full stack JavaScript development using the MEVN stack with quick and easy steps*. Packt Publishing. 1. vyd. ISBN 978-1-801-07414-8.
- [23] SHAVIN, Maya, 2023. *Learning Vue: Core Concepts and Practical Patterns for Reusable, Composable, and Scalable User Interfaces*. O'Reilly Media. 1. vyd. ISBN 978-1-492-09882-9.
- [24] USMANMALIK, 2025. *Understanding Vue.js Reactivity System: ref, reactive, computed, watch, and watchEffect*. Online. Dostupné z: <https://medium.com/%40usmanmalik748/understanding-vue-js-reactivity-system-ref-reactive-computed-watch-and-watcheffect-ef4bbb2cca49>. [cit. 2025-05-31].
- [25] GERCHEV, Ivaylo, 2024. *Leveraging Pinia to simplify complex Vue state management*. Online. Dostupné z: <https://blog.logrocket.com/complex-vue-3-state-management-pinia/>. [cit. 2025-05-31].
- [26] NOWAK, Maja, [b.r.]. *Vue vs React: Choosing the Best Framework for Your Next Project*. Online. Dostupné z: <https://www.monterail.com/blog/vue-vs-react>. [cit. 2025-05-31].
- [27] WICKRAMASINGHE, Shanika, 2023. *Angular vs Vue: A Head-to-Head Comparison*. Online. Dostupné z: <https://kinsta.com/blog/angular-vs-vue/>. [cit. 2025-05-31].

SEZNAM OBRÁZKŮ

Obrázek 1.1	Hlavní stránka ROSTEM	13
Obrázek 3.1	Logo Gitu	19
Obrázek 3.2	Logo Dockeru	21
Obrázek 3.3	Logo Vue.js	22
Obrázek 5.1	Návrh UI stránky Obecné informace	32
Obrázek 5.2	Návrh UI přehledu otázek	34
Obrázek 5.3	Návrh UI editoru otázek.....	35
Obrázek 5.4	Návrh UI přehledu kvízů.....	36
Obrázek 5.5	Návrh UI detailu kvízu	37
Obrázek 5.6	Návrh UI žebříčku	38

SEZNAM TABULEK

Tabulka 3.1	Srovnání frameworků Vue.js a React.....	25
Tabulka 3.2	Srovnání frameworků Vue.js a Angular	26

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

UTB	Univerzita Tomáše Bati
PWA	Progressive Web App
VCS	Version Control System
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
WSL	Windows Subsystem for Linux
DOM	Document Object Model
SFC	Single File Component
SPA	Single Page Application
MVC	Model-View-Controller
UI	User Interface
HTTP	Hypertext Transfer Protocol
JS	JavaScript
URL	Uniform Resource Locator
API	Application Programming Interface

SEZNAM PŘÍLOH