TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Naučná webová aplikace pro základní školu

Bakalářská práce

Studijní program: B0613A140005AI – Aplikovaná informatika
Studijní obor: 1802T007 – Informační technologie

Autor práce: **Lukáš Dufek**
Vedoucí práce: Ing. Jan Hybš

Liberec 2023

FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Zadání bakalářské práce

Naučná webová aplikace pro základní školu

Jméno a příjmení:	**Lukáš Dufek**
Osobní číslo:	M19000194
Studijní program:	B0613A140005 Informační technologie
Specializace:	Aplikovaná informatika
Zadávající katedra:	Ústav nových technologií a aplikované informatiky
Akademický rok:	2022/2023

Zásady pro vypracování:

1. Proveďte rešerši tématiky webových naučných portálů.
2. Navrhněte webovou aplikaci, sloužící pro správu interaktivních testů cílené na žáky prvního stupně základních škol.
3. Implementujte webovou aplikaci, umožňující žákům jednoduše absolvovat interaktivní testy. K implementaci použijte moderní webové technologie a frameworky.
4. Webovou aplikaci doplňte o odměnový systém pro žáky formou hry.
5. Demonstrujte výslednou aplikaci, ověřte její spolehlivost a jednoduchost přístupu.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	30 – 40 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] PLEȘOIANU, Ana-Maria (2020). *Teachers Perspective On The Effects Of The Reward System On Evaluation Outcomes*. [online] ResearchGate. Available at: https://www.researchgate.net/publication/342252326_Teachers_Perspective_On_The_Effects_Of_The_Reward [Accessed 6 Oct. 2022].

[2] JAVED, A. and MUHAMMAD, N. (2021). Teachers Perceptions about Reward Systems in Classroom. *Journal of Education and Social Studies*, [online] 2(2), pp.59–62. doi:10.52223/jess.20212204.

[3] DE AGUILERA, Miguel and NOGUERO, Alfonso Mendiz (2003). *Video games and education: (Education in the face of a 'parallel school')*. [online] ResearchGate. Available at: https://www.researchgate.net/publication/220686511_Video_games_and_education_Education_in_the_face_of [Accessed 6 Oct. 2022].

[4] LOWERY, B.R. and KNICK, F.G. (1982). *Micro-Computer Video Games and Spatial Visualisation Acquisition*. [online] ResearchGate. Available at: https://www.researchgate.net/publication/246819209_Micro-Computer_Video_Games_and_Spatial_Visualisation_Acquisition [Accessed 6 Oct. 2022].

Vedoucí práce:	Ing. Jan Hybš Ústav nových technologií a aplikované informatiky
Datum zadání práce:	12. října 2022
Předpokládaný termín odevzdání:	22. května 2023

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci dne 19. října 2022

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

16. 5. 2023 Lukáš Dufek

Naučná webová aplikace pro základní školu

Abstrakt

Tato bakalářské práce se zabývá problematikou tvorby webových aplikací. Cílem práce bylo vytvořit naučnou webovou aplikaci zaměřenou na žáky 1.stupně základních škol. Hlavní důraz byl kladen na to, aby žáci měli motivaci tuto aplikaci používat. Aplikace je rozdělena na matematickou a herní část. V matematické části žáci získávají odměny v podobě herní měny (stříbrné a zlaté mince), za úspěšně vyřešená cvičení a slovní úlohy. Mince pak využijí v herní části, kde si za ně vylepšují svoji pohádkovou postavu, se kterou se pak posouvají v příběhu dál. Aplikace umožňuje i přístup pro učitele, kteří zde mohou vytvářet a přidávat slovní úlohy pro své žáky. Aplikace byla vytvořena za pomocí frontedového frameworku VueJS.

Práce je tématicky rozdělena do čtyř části. V první části je nejprve popsána tématika webových aplikací. Dále jsou zde představeny technologie, které byly využity a nakonec je provedena rešerše naučných webových portálů. Druhá část se zabývá návrhem aplikace. Jsou zde popsány body, které bylo potřeba vyřešit, včetně doporučených postupů. Třetí část se zabývá samotnou implementací. V této části je podrobně popsán celý postup při tvorbě aplikace. V poslední části jsou popsány výsledky, které byly získány při demonstraci této práce, včetně tipů na vylepšení.

Klíčová slova: VueJS, webová aplikace, odměnový systém, žák a učitel, matematika pro 1.stupeň

Educational Web Application for Elementary School

Abstract

This bachelor thesis deals with the topic of web application development. The aim of the work was to create an educational web application aimed at the pupils of the first grade of primary schools. The main emphasis was on the motivation of the pupils to use the application. The app is divided into a mathematical and a game part. In the mathematical part, pupils receive rewards in the form of in-game currency (silver and gold coins), for successfully solving exercises and word problems. The coins are then used in the game part, where can be used to improve the user's fairy tale character, which then develops further in the story. The app also allows teachers to create and add word problems for their students. The app was created using the fronted framework VueJS.

The work is thematically divided into four parts. In the first part, the topic of web applications is described. Then the technologies that have been used, and the learning web portals research. The second part deals with the design of the application. The points that needed to be addressed are described, including the procedures. The third part deals with the implementation itself. In this part, the whole process of creating the application is described in detail. The last part describes the results that were obtained during the demonstration of this work, including tips for improvement.

Keywords: VueJS, web application, reward system, student and teacher, math for 1st grade

Poděkování

Tímto bych rád poděkoval vedoucímu mé bakalářské práce Ing. Janu Hybšovi, jak za odborné rady v textové části, tak i za poskytnutí cenných rad v části praktické. Díky vám jsem získal mnoho nových zkušeností, které mi pomohli při tvorbě této práce, ale také posunuli mé dovednosti v tématice webových aplikaci.

Velké díky patří také mému příteli Jaroslavu Špačkovi, který se věnuje tvorbě herních assetů, pod pseudonymem DonSpaghetti. Děkuji za poskytnutí originálních herních obrázků, které aplikaci dodaly kvalitní design.

Chtěl bych poděkovat také své rodině, která mě podporovala, nejen při psaní této práce, ale i v celém průběhu studia.

Seznam obrázků

Seznam tabulek

Obsah

1 Úvod

Vzdělávání v dnešní době stále více přechází do elektronické podoby a žáci postupně mění své sešity za tablety a notebooky. Velké změny se však dějí v samotné metodě výuky, kde se pedagogové snaží hledat, ten nejefektivnější způsob, jak své studenty vzdělávat. Často se tak stává, že se žáci učí a rozvíjejí své dovednosti prostřednictvím naučných aplikací. Právě webové aplikace jsou v dnešní době stále více populární, především díky své přístupnosti, kde není potřeba nic stahovat, ale pouze do prohlížeče zadat požadovanou adresu.

I přes možnosti dnešní technologie je ale těžké najít kvalitní naučnou webovou aplikaci, která by vyhovovala těm studentům, kterým chybí motivace k učení, což je taky jeden z problémů dnešní mladé generace. Webové aplikace, které mají kvalitní a zábavný obsah je většinou nutné stáhnout do svého zařízení. Aplikace, které je možné využívat přímo v prohlížeči zase nedisponují zábavnou formou. Tato práce byla navrhnuta za účelem vytvořit naučnou webovou aplikaci, která bude kombinací výuky a hry.

Bakalářská práce pojednává o tvorbě webové naučné aplikace, zaměřené na matematiku pro 1.stupeň základní školy. Myšlenkou pro vznik této aplikace bylo motivovat děti k počítání příkladů a řešení slovních úloh takovým způsobem, aby to pro ně bylo zábavné. Aplikace se tímto způsobem rozděluje na dvě části.

V první části žáci počítají příklady a slovní úlohy. Příklady se generují náhodně a vyskytují se zde převážně všechny typy příkladů z matematiky pro 1.stupeň, ovšem kromě geometrie. Slovní úlohy žákům mohou zadávat přímo učitelé, tudíž obsah a náročnost slovních úloh je pouze na nich. Obtížnost učiva je navíc rozdělená podle ročníků žáků. Do dalšího ročníku může žák postoupit pokud úspěšně absolvuje závěrečný test, ke kterému se, ale může dostat až při splnění daných podmínek, kterými jsou: spočítání dostatečného počtu cvičení a požadovanou procentuální úspěšností. Za správné výsledky jak příkladů tak i slovních úloh, jsou žáci odměněni ve formě stříbrných či zlatých mincí, podle toho jak úspěšně počítali.

Tyto mince pak mohou využít ve druhé části, která obsahuje hru s příběhem. Do hry se však mohou žáci přihlásit až od 3.ročníku, z toho důvodu, že hra požaduje, aby žáci měli určité matematické i finanční znalosti. Mají zde svoji pohádkovou postavu a mince mohou utratit, právě za vylepšování této postavy, kupováním předmětů a nebo přímým zlepšením dovedností. V poslední řadě, je tu sekce příběh, kde se v jeho průběhu postava žáka musí vypořádat se zlými pohádkovými bytostmi, než dosáhne šťastného konce.

2 Naučné webové aplikace

2.1 Pojem webová aplikace

Pojem webová aplikace [1] je označení pro software zprostředkovaný internetovým prohlížečem. Oproti desktopovému softwaru jej není potřeba instalovat, stačí když je uživatel připojen k internetu a zadá do prohlížeče požadovanou adresu.

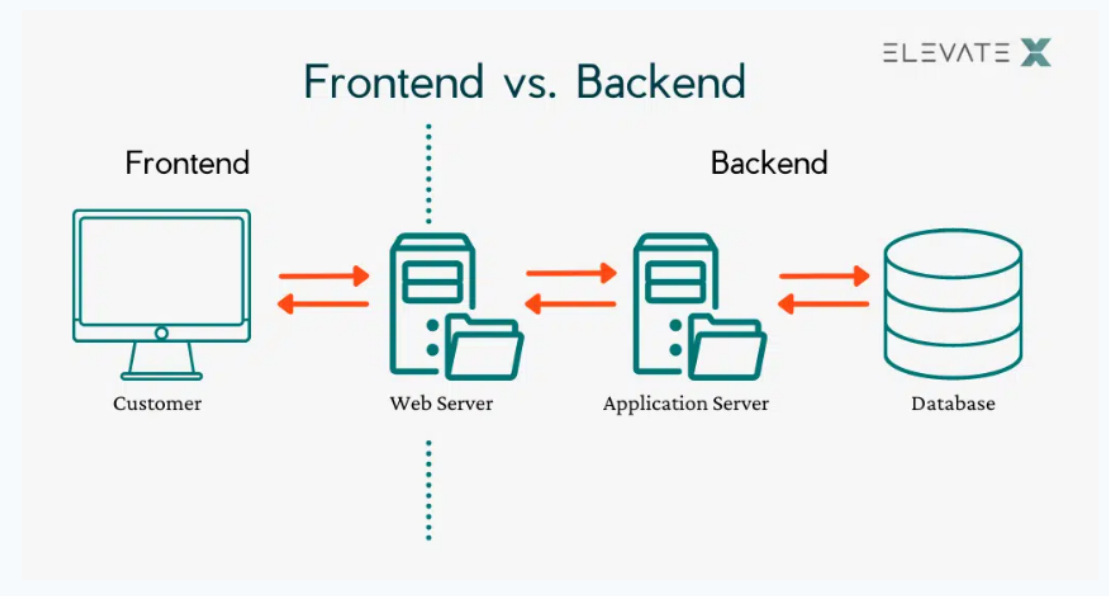
Webové aplikace díky své přístupnosti, ale i multiplatformosti, zaznamenaly v posledních letech obrovský vzestup [1]. Na rozdíl od ostatních aplikací, nejsou vyvíjeny pro konkrétní operační systém nebo hardware. Pro přístup tak stačí pouze připojení k internetu z jakéhokoli zařízení, které má v sobě prohlížeč.

2.2 Vývoj webové aplikace

Vývoj webové aplikace je proces, při kterém je navržena, vytvořena, uvedena do provozu a následně provozována a podporována webová aplikace. Toho lze docílit v první řadě výběrem správné technologie a ověřených postupů. Dnes nejčastějším způsobem postupu při tvorbě webové aplikace je výběr správných frameworků a následné rozdělení projektu do dvou částí: Frontend a Backend [1].

2.2.1 Framework

Framework je softwarová struktura [2], která slouží jako podpora při programování a organizaci projektu, čímž se snaží zjednodušit práci pro programátora. Může obsahovat podpůrné programy, knihovny, doporučené postupy při vývoji apod. Kritici tvrdí, že použitím frameworku bude kód pomalejší či jinak neefektivní a že čas, který se ušetří použitím cizího kódu, se musí věnovat nastudování frameworku. Nicméně při opakovaném nasazení nebo ve velkém projektu dojde k výrazné úspoře času. V dnešní době existuje pro programovací jazyky několik frameworků. V této práci byly použity frameworky: Vue pro frontend a Express pro backend. Oba tyto frameworky jsou psané a používané v programovacím jazyce JavaScript.



Obrázek 2.1: Frontend vs. Backend
[8]

2.2.2 Frontend

Pojem frontend pochází z oblasti programování webových aplikací, kde slouží k označení části webu viditelné běžným uživatelům. Např. U internetového obchodu frontend zahrnuje katalog zboží, nákupní košík a objednávkový formulář. V tomto projektu byl pro Frontend vybrán framework Vue. Dalšími velmi populárními frameworky pro frontend jsou například Angular nebo React [3].

Vue

Vue [4] je progresivní JavaScriptový framework, což znamená, že umožňuje aplikaci rozdělit na části, které lze vyvíjet nezávisle. Při založení projektu ve vue se vytvoří celkový ekosystém, který mimo jiné obsahuje i tzv. komponenty reprezentující jednotlivé webové stránky nebo pouze jejich části. Každá komponenta je v základu tvořena ze tří částí: HTML, CSS a JavaScriptu. S těmito komponentami a mezi nimi lze pracovat pomocí velkého množství balíku a knihoven, které vue poskytuje [4].

2.2.3 Backend

Backend je část webové aplikace, která slouží k administraci webu a zpracování dat. V případě internetového obchodu, který byl zmiňován v sekci Frontend, by backend sloužil ke vkládání nového zboží nebo k úpravám stávajícího zboží. V jiných případech se například stará o správu uživatelů (například přihlašovacích údajů). Pomocí backendu tedy přistupujeme k místu, kde jsou uložena data. V tomto projektu byl pro backend vybrán framework Express. Dalšími populárními frameworky jsou například Spring Boot (napsán v programovacím jazyce Java) nebo Lavarel (napsán v programovacím jazyce PHP) [3].

NodeJS

NodeJS je softwarový systém umožňující spouštět JavaScript mimo webový prohlížeč. Jeho primárním návrhem a účelem je tvorba serverové části webových aplikací. JavaScript se tedy díky tomuto prostředí dá používat i na serveru (Backend) a ne pouze u klienta (Frontend). NodeJS klade velký důraz na škálovatelnost, tzn. schopnost odlišit mnoho připojených klientů naráz [5].

Express

Express je velmi populární framework pro práci v NodeJS. Byl vytvořen za účelem snadnějšího vytváření webových aplikací. Jeho další výhodou je rychlé propojení s databázemi jako jsou MySQL nebo MongoDB.

MongoDB

MongoDB patří mezi NoSQL databáze. Je objektově orientovaná, jednoduchá, dynamická a dobře škálovatelná. Pro data místo řádků a sloupců tabulky, používá dokumenty a pole. Data se ukládají ve formátu JSON. Objekty mohou být do sebe vkládány, což je rozdíl oproti relačním databázím, kde se to řeší například několika propojenými tabulkami. Pro využívání této databáze je nejprve potřeba založit si účet na cloud.mongodb.com, pro využívání online úložiště. Lze, ale také využívat databázi lokálně a tudíž není potřeba si zakládat účet. V tomto projektu byla využita možnost online úložiště. Po založení účtu je potřeba založit i samotnou databázi, kde budou data uložená. V programu je pak zapotřebí definovat model dat a jeho atributy, případně i omezení a vlastnosti atribut. V poslední řadě je potřeba v programu vytvořit připojení k databázi a naprogramovat metody pro operace s daty. V tomto projektu byl pro testovací operace s daty využíván software "Postman" [6].

Postman

Postman bylo původně jen rozšíření prohlížeče, dnes je to aplikace pro Windows, OS X a Linux. Slouží vývojářům k navrhování, sestavování a testování jejich API (rozhraní pro programování aplikací). Po připojení na danou adresu, Postman obsahuje mnoho operací, které lze s daty provádět, především přidávat(POST) a mazat(DELETE), ale také data například prohlížet(GET) nebo upravovat již existující(PUT) a spoustu dalších operací. Než je však možné se na danou adresu přes tuto aplikaci připojit, je nejdříve potřeba vytvořit daný server a naprogramovat zmíněné metody, abychom je mohli v Postmanu používat [7].

2.3 Požadavky na webovou aplikaci

Při tvorbě webové aplikace je nutné dbát na to, aby byla jednoduchá, přehledná a dobře se ovládala. Dále je nutné brát v potaz, že někteří uživatelé budou aplikaci, používat i na jiných rozlišeních svých displejích nebo i na mobilních telefonech, čímž je potřeba zajistit, aby byla dobře responzivní a všem se správně a přehledně zobrazovala. Dále by měla být aplikace jednoduchá na ovládání, tzn. aby bylo jasné na co lze kliknout, například tím, že se při najetí myši na tlačítko zvýrazní či podtrhne text a podle textu tlačítka by bylo dobré, aby uživatel věděl, co se po kliknutí stane. V poslední řadě je i zapotřebí řešit vizuální stránku aplikace, aby uživatele neodrazovala od používání.

2.4 Rešerše naučných portálů

2.4.1 ABCya

ABCya.com je webový portál, který poskytuje vzdělávací hry pro děti. Tyto hry jsou rozděleny do kategorií, podle roku dítěte, ale i podle předmětů: Angličtina, Matematika, Přírodověda. Dále je tu možnost registrace, kde si uživatel založí profil a následně si může zakoupit prémiovou verzi. Má na výběr levnější verzi pro rodiče (podpora pro 5 zařízení) nebo dražší verzi pro učitele, kde je podpora až pro 30 zařízení. Učitelé si takto mohou sestavit plány pro celou třídu. Tento portál je kompletně v Anglickém jazyce.

Lze tedy o tomto portálu říci, že poskytuje sice hry a formu vzdělání z více okruhů, ale hry, ačkoli jich je tu spoustu, jsou spíše jednorázové a žák ani učitel tak nevidí celkový postup a prospěch žáka.

2.4.2 Adventure Academy

Vývojáři této aplikace zašli ještě dál než u ABCya.com a připravili tak pro žáky celý virtuální svět, plný vzdělávacích aktivit. Žáci si tak mohou vytvořit a postupně vylepšovat svoji virtuální postavu, se kterou cestují po tomto světě a plní vzdělávací aktivity a hry, téměř ze všech okruhů učiva základní školy. Aplikace poskytuje velmi interaktivní přístup vzdělávání, ale to s sebou přináší i různá omezení, kterými jsou: nutnost registrace, stažení a instalace, což obnáší i určité nároky na systém. Aplikace je navíc placená a pouze v Anglickém jazyce.

2.4.3 Matika.in

Matika.in je český webový vzdělávací portál zaměřený na matematiku pro žáky ze všech ročníků základní školy. Nabízí dětem počítat příklady, řešit obrázkové hádanky, slovní úlohy, ale i matematické hry. Odměnový systém pro žáky, zde funguje formou odznaků a skóre v žebříčku. Žáci mají bezplatný přístup bez registrace. Existuje tu však i speciální funkce pro rodiče a učitele, kteří se ale musí registrovat a mohou tak dohlížet na prospěch žáků.

2.4.4 Matemag

Aplikace Matemag je stejně jako Matika.in zaměřena pouze na matematiku a také je kompletně v českém jazyce. Oproti předchozí uvedené aplikaci, nabízí zcela jinou formu výuky. Tou je příběh, během kterého dítě řeší matematické úlohy a za každé splnění této úlohy se posouvá v příběhu dál. Je zde možnost zapojení rodičů, kteří mohou sledovat postup a úspěšnost dítěte. Aplikace má výborný grafický design, český dabing postav a je velmi interaktivní. Nevýhodou je však nutné stažení a instalace a poskytuje pouze placený přístup.

2.4.5 Celkové zhodnocení

Z této rešerše lze odvodit to, že vzdělávací aplikace, které jsou více interaktivní, jako například, že mají vlastní příběh nebo volnou hratelnost, je nutné ve většině případech koupit a nainstalovat do zařízení. Oproti tomu vzdělávací aplikace, které lze hrát přímo v prohlížeči, nejsou tolik hratelně příznivé a jsou zaměřené spíše na samotné vzdělávání, než na formu hry.

2.5 Rešerše tématických okruhů matematiky pro 1. stupeň základní školy

Pro sestavení aplikace bylo nutné podrobně prostudovat tématické okruhy matematiky pro 1.stupeň. Pro tuto potřebu posloužilo hned několik internetových zdrojů:

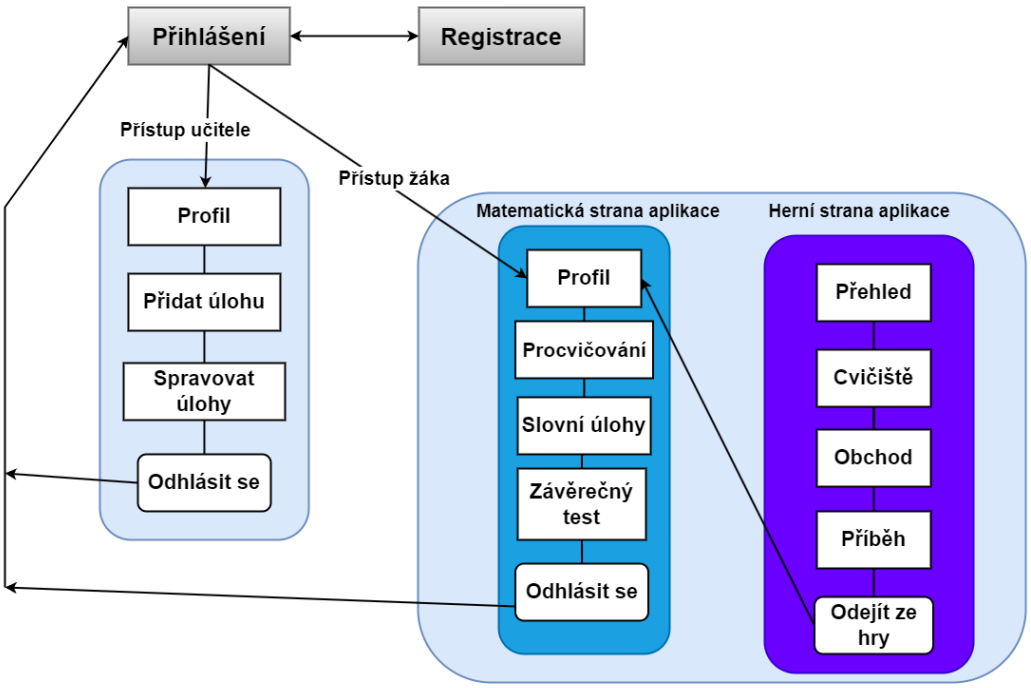
- (zskomenskeho-skutec.cz) ⇒ Stránka školy obsahuje kromě všeobecných informací o škole, jejích aktivitách, rozvrzích, žácích apod., také přehledné kompletní učební plány všech předmětů z 1.stupně základní školy. Pro potřeby aplikace tak byla využívána pro stanovení obsahu učiva matematiky pro dané ročníky.
- (onlinecviceni.cz) ⇒ Tato stránka obsahuje online cvičení pro všechny ročníky základních škol z Českého jazyka a matematiky. Pro každý ročník je zde velké množství cvičení ze všech okruhů daného ročníku. V této práci poskytla inspiraci pro sestavování cvičení z náhodně generovaných příkladů.
- (skolajakubcovice.cz) ⇒ Tato stránka také obsahuje všeobecné informace o Základní ale i Mateřské škole v Jakubčovicích nad Odrou. Mimo to, ale obsahuje v učebních plánech i několik cvičení pro každý ročník ze základní školy. Přínosem byla právě z toho důvodu, že mezi cvičeními jsou i vzorové závěrečné testy, které sloužily jako inspirace pro sestavování závěrečných testů v aplikaci.

3 Návrh aplikace

3.1 První návrh

Před samotným programováním i navrhováním byl nápad vytvořit aplikaci, která bude dětem rozvíjet matematické znalosti, takovým způsobem, aby se k počítání sami rádi vraceli. Zde se naskytla myšlenka žáky za úspěšně vypočítané příklady odměňovat. Také bylo nutné vyřešit jakým způsobem je odměňovat a do čeho mohou své odměny investovat. Bylo tedy nutné vymyslet aplikaci tak, aby žák svou odměnu potřeboval, tudíž aby ji investoval do něčeho, co musí stále rozvíjet. Nejdříve se naskytla myšlenka, že žák bude správcem virtuální osady, kterou bude muset rozšiřovat a bránit před útoky nepřátel. Kvůli jistým komplikacím a pro zjednodušení, jak ze strany programátora tak i pro samotného žáka, bylo později navrhnuto vylepšovat si pouze samotnou postavu. Ale pouze samotné vylepšování této postavy by postupně žáka jistě přestalo bavit, neboť by se ztrácel význam žákovy snahy a samotná potřeba tuto postavu vylepšovat. A tak vznikla myšlenka přidat příběh, ve kterém se bude moci žák posouvat, tím, že bude postupně porážet své nepřátele. Aby toho však mohl dosáhnout, je nucen svoji postavu vylepšovat.

Další myšlenkou bylo, zapojení učitelů do aplikace. Už v této fázi bylo jasné, že bude potřeba vytvořit systém, který bude umět rozdělit uživatele na učitele a žáky a že žák i učitel budou mít rozdílné možnosti v aplikaci, tedy i rozdílné prostředí. Jedna z možností, jak uživatele odlišit bylo vytvoření registračního a přihlašovacího systému, kde se podle své identifikace uživatel dostane na příslušnou stranu aplikace. Zde se ještě naskytla možnost přístupu do aplikace bez přihlášení (pokračovat jako host). Tento návrh byl, ale později zamítnut, především z toho důvodu, že host by měl velmi málo možností v aplikaci a jeho výsledky by nebylo možné nikde zaznamenat. Dalším krokem tedy bylo definovat možnosti přístupu žáka a přístupu učitele.



Obrázek 3.1: Diagram přístupu do aplikace

3.2 Obecný přístup do aplikace

Každý uživatel se musí do systému nejprve zaregistrovat, což může být problém obzvlášť pro žáky prvního ročníku a bude zde tedy zapotřebí, aby jim s vyplněním údajů případně pomohli jejich učitelé. Při registraci bylo nutné promyslet, jaké údaje bude muset uživatel zadávat. Dále bylo potřeba vybrat prvek jedinečnosti, kde se nejdříve nabízelo spojení křestního jména a příjmení. Avšak může se stát, že dva uživatelé mají totožné křestní jméno i příjmení a pravě zde by nastal problém, který by se jistě dal nějakým způsobem vyřešit. Nicméně mnohem jednoduší cesta byla, vybrat jako prvek jedinečnosti emailovou adresu, kterou uživatel musí zadat také. Další údaj, který musí uživatel zadat je heslo, to musí ještě potvrdit tím, že ho napíše znovu. Zde je další podmínkou, aby se hesla shodovala. Každý uživatel tedy musí zadat jedinečnou emailovou adresu, své křestní jméno, příjmení a následně dvakrát napsat heslo. V poslední řadě je ještě potřeba uživatele už při registraci rozeznat, jestli je žák nebo učitel, protože podle toho se mu v systému

vytvoří potřebné vlastnosti. Uživatel je tedy ještě nucen vybrat si roli. V případě, že si vybere žáka, musí si ještě vybrat ročník. Zde již nebylo potřeba nic dalšího řešit a uživatel se všemi výše uvedenými údaji se může registrovat.

Při přihlášení bylo potřeba, aby se uživatel prokázal svým identifikátorem (tedy emailovou adresou) a svým heslem. Pokud správně zadá emailovou adresu a heslo, které je k této adrese připojené, může se úspěšně přihlásit. Aplikace už sama pozná, jestli se jedná o žáka nebo učitele a podle toho mu zpřístupní dané funkce.

3.3 Přístup žáka

3.3.1 Matematická část

Po úspěšné registraci a následném přihlášení byly pro režim žáka plánovány možnosti: počítání příkladu, řešení slovních úloh, závěrečný test pro postup do dalšího ročníku a v poslední řadě stránka s přehledem jeho údajů a vlastností. Stránka s přehledem se nabízela zpřístupnit jako první strana, kterou uživatel uvidí, jakmile se úspěšně přihlásí. Pořadí zbylých stránek bylo navrženo chronologicky. U stránky s počítáním příkladů bylo jasné, že je potřeba navrhnout systém, který bude příklady generovat náhodně, pouze je zobrazovat a nikam neukládat, což také částečně vylučuje opakování příkladů. Také je nutné, aby příklady odpovídali okruhům ročníku žáka. Na stránce se slovními úlohami oproti tomu bylo potřeba někde trvale ukládat jednotlivé slovní úlohy, především proto, že nebyly plánovány pro náhodné generování, ale pro zadávání učiteli. V poslední řadě je tu stránka se závěrečným testem. Tento test obsahuje kombinaci generovaných příkladů a slovních úloh od učitelů, oboje v souladu ročníku žáka. Už zde bylo potřeba zamezit tomu, aby žák mohl rovnou postoupit do dalšího ročníku a tak bylo navrženo omezení, které žákovi zpřístupní test, až když vypočítá dostatek příkladů s určitou úspěšností.

3.3.2 Herní část

Jak již bylo řečeno, aplikace byla navržena tak, aby žáka bavila. Byla tedy navržena možnost vstupu do hry, která ale byla zamýšlena až pro žáka od 3.ročníku, z toho důvodu, že bylo potřeba, aby dítě dosáhlo věku, kdy je nějakým způsobem matematicky i finančně (ačkoliv jsou to virtuální peníze) gramotné a dokáže si tak promyslet hodnoty jednotlivých věcí ve hře. Pokud tedy žák dosáhne 3.ročníku zpřístupní se mu možnost vstupu do hry, kde si může vylepšovat vlastnosti postavy jménem Jakub, buď přímo nebo kupováním předmětů. Poslední stránka ve hře byla plánována pro vstup do příběhu, kde bylo zamýšleno realizovat knihu, ve

které by se žák listováním posouval a jakmile by narazil na nepřítele, příběh by se na tom místě zastavil dokud by žák nepřítele neporazil.

3.4 Přístup učitele

Po úspěšné registraci a následném přihlášení byla učiteli plánována možnost zadávat slovní úlohy. Učiteli bylo umožněno zadat veškeré atributy úloh, včetně odměny kterou za ni žák dostane, pokud ji správně vypočítá. Dále má učitel možnost spravovat všechny slovní úlohy, tedy přepisovat veškeré atributy dokonce i úlohy z databáze mazat.

3.5 Postupy použité při návrhu

3.5.1 Routing a navigace

V první řadě bylo potřeba vyřešit přechod mezi stránkami. Jelikož žák i učitel budou mít odlišené možnosti v aplikaci, bylo nutné jim vytvořit odlišné cesty. Zde se nabízela možnost vytvořit navigační lištu, kterou budou mít obě strany odlišnou. Pro žáka však bylo potřeba vytvořit další navigační lištu pro režim hry. V první verzi aplikace, byly vytvořeny hned tři navigační lišty - jedna pro učitele, druhá pro žáka v režimu matematiky a třetí pro žáka v režimu hry. Později se naskytlo vylepšení předělat navigační lišty do jedné a jednotlivým cestám vytvořit atributy, které určí, zda se jedná o stránku pro učitele nebo žáka a v případě žáka, zda jde o cestu v herní nebo v matematické části. V poslední řadě zde bylo ještě potřeba ošetřit to, aby se nepřihlášený uživatel nemohl dostat do aplikace, ale také, aby se žák přes adresu nemohl dostat do učitelských funkcí a naopak.

3.5.2 Propojení s databází

Už v počátcích návrhu aplikace bylo zřejmé, že některá data bude potřeba uchovávat v databázi trvale. Bylo tedy nutné aplikaci s databází propojit. Dále bylo potřeba vytvořit strukturu a modely dat a jak budou v databázi vypadat. Aplikace tedy potřebovala vést záznamy o svých uživatelích, o slovních úlohách a o položkách ve hře. Zde byly vytvořené hned tři různé databáze a aplikace byla propojená se všemi, bohužel nebylo možné spojení se všemi třemi najednou. Aplikace se tak při požadavku na data musela vždy připojit do dané databáze, kde se provedl požadavek a zase se odpojit, pro možnost připojení do jiné databáze. Tento postup sice fungoval, ale ne příliš efektivně. Velké množství dat tak muselo být uloženo v lokáním úložišti a navíc neustálé připojování a odpojování snižovalo čas pro operaci s daty. Později však bylo implementováno vylepšení v podobě

vytvoření různých kolekcí v jedné databázi. Každá kolekce tak obsahovala data, podle již vytvořených modelů. V tomto bodě již nebylo potřeba řešit připojování a odpojování databází, protože aplikace bylo připojena hned při spuštění do jedné databáze, ve které bylo možné data rozdělit podle kolekcí. Jedním z posledních vylepšení bylo, odstranění kolekce s položkami do hry. Jelikož u položek nebylo nutné provádět žádné změny, ale pouze je vytvořit a tak nebylo potřeba je mít uložené v databázi, ale pouze v úložišti aplikace.

3.5.3 Generování příkladů

Příklady pro žáky, bylo zapotřebí připravit tak, aby se, co nejméně opakovaly a aby byly co nejvíce náhodné. Proto byl hned na začátku zamítnut návrh ukládat je do databáze. Postup, který byl navrhnut spočíval v tom, že příklady se budou generovat náhodně a vygeneruje se jich hned daný počet, který bude muset žák spočítat jako cvičení, aby dostal odměnu. Pro každý ročník bylo však nutné definovat různé typy příkladů. Například pro 2.ročník příklady na sčítaní, odčítání, násobení, zaokrouhlování čísel apod. U každého typu příkladů bylo ještě zapotřebí zajistit to, aby nepřesahoval znalosti daného ročníku, např. abychom se u odčítání nedostali do záporných čísel, u sčítání a odčítání abychom nepřešli přes desítku atd. Ve chvíli, kdy cvičení dosáhne požadovaného počtu příkladů, je ještě potřeba zkontrolovat, jestli cvičení neobsahuje dva stejné příklady a případně jeden nahradit. Nakonec bylo dobré, ještě příklady zamíchat, aby se například nezobrazovaly nejdříve příklady na počítání, pak na porovnávání apod. V tomto momentě je cvičení připravené a žák může počítat. Výsledky, které zadá, se uloží a po posledním příkladu se porovnají se správnými výsledky a podle počtu správných výsledků dostane žák příslušnou odměnu.

3.5.4 Slovní úlohy

U slovních úloh bylo hned na začátku jasné, že je nelze generovat náhodně a pokud ano je možné, že by nedávaly smysl. Zde se nabídl nápad zapojení učitelů do aplikace. Učitelé se tedy po registraci mohou přihlásit a vytvářet, upravovat či mazat slovní úlohy. Učitelé mohou navíc i stanovit odměnu za správně vyřešenou úlohu, poté co zváží složitost dané úlohy. Tyto úlohy bylo rozhodně nutné mít uložené v databázi, především proto, že jsou předávány mezi oběma přístupy do aplikace. Navíc tak stejnou úlohu mohou řešit různí žáci. Každá úloha má parametr ročník, podle kterého se zobrazí odpovídajícím žákům.

4 Implementace aplikace

4.1 Založení projektu

Prvním krokem při tvorbě této aplikace bylo založení projektu a v něm nastavit všechny potřebné náležitosti. Založení projektu proběhlo tak, že v příslušné složce, která byla vybrána pro projekt, byl v příkazovém řádku vývojového prostředí zavolán příkaz *npm init*. Po zadání příkazu je potřeba v příkazovém řádku odpovědět na několik věcí, jako jsou: název projektu, verze aplikace, popis projektu, spouštěcí skripty, jméno autora atd. Tyto věci je nutné přímo při zadání vyplnit. Následně se vytvoří soubor "package.json", který obsahuje všechny výše zmíněné informace. Tento soubor je důležitou součástí aplikace, protože také obsahuje pomocné moduly a jejich verze, které je později potřeba do projektu přidat. Dalším krokem bylo vytvoření aplikace ve frameworku Vue. To bylo realizováno příkazem *vue create "název projektu"*. I v případě tohoto příkazu bylo potřeba vyplnit určité informace, jako jsou například: verze vue a další pomocné moduly, které je, ale možné přidat i po vytvoření. Po dokončení tohoto procesu, byla vytvořena komplexní struktura pro frontendovou část, také se ale vytvoří kompletní základní aplikace, kterou vue poskytuje. Aby bylo možné oba výše uvedené příkazy provést, je zapotřebí mít v zařízení nainstalované moduly nodeJS a vue. V zařízení, kde byla aplikace tvořena, ale byly tyto moduly již nainstalované a připravené z předchozích projektů.

4.2 Struktura projektu

Ještě před samotným programováním aplikace, bylo zapotřebí nastavit si přehledné a jednoduché prostředí, především pro snazší orientaci v projektu.

4.2.1 Frontendová část

Ve frontedové části se nachází adresář **src**, ve kterém se objevují následující adresáře:

- **assets** ⇒ pro ukládání všech potřebných obrázků

- **components** ⇒ obsahuje tzv. komponenty. Jednotlivé komponenty mohou tvořit část webové stránky. Mají však stejnou strukturu, jako celá webová stránka vytvořená ve Vue. Avšak často fungují tak, že se webová stránka skládá ze dvou nebo více komponent, což nám dává značnou výhodu, v případě, že komponenta bude použitá na více místech v programu. Každý soubor komponenty s příponou ".vue"obsahuje 3 části: **template** (kde programátor tvoří html), **script** (zde se tvoří funkcionalita těchto komponent, uchovávají se zde např: data a algoritmy.) a **style** (zde je prostor pro tvorbu CSS, tedy designu stránek). V tomto projektu byly vytvořeny komponenty například pro zobrazení stříbrné i zlaté mince nebo pro navigační lištu, která se vyskytuje na každé stránce v aplikaci.

- **store** ⇒ obsahuje soubory s veškerými daty, se kterými se v aplikaci pracuje, například: data uživatelů, položek nebo nepřátel. Pro přesnost je potřeba zmínit, že data uživatelů zde nejsou trvale uložena, jen se načtou při přihlášení z databáze a v případě změny se znovu odešlou do databáze. Dále složka obsahuje ale i metody pro veškeré změny v těchto datech, jako je například metoda pro přidání nové slovní úlohy.

- **router** ⇒ obsahuje soubor, který uchovává data o všech cestách v aplikaci. Více bude popsán v sekci Navigace a přechod mezi stranami v aplikaci

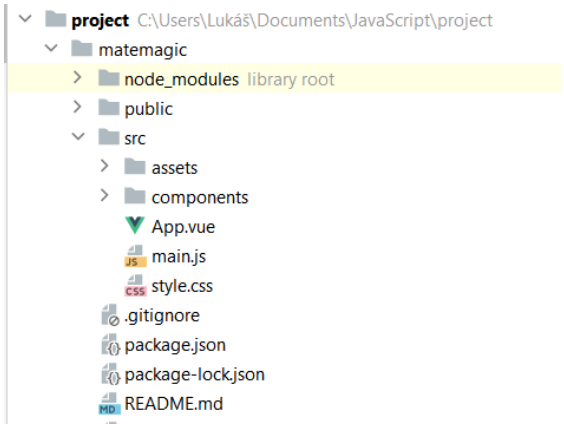
- **views** ⇒ tvoří jednotlivé kompletní webovové strany aplikace. Právě zde se volají komponenty.

Dále se ve frontendové části vyskytuje soubor "App.vue", který je spouštěcí často i úvodní stranou aplikace. Zpravidla se zde skládají ostatní komponenty. Soubor "main.js"slouží pro importování moduly, vytváření proměnných a podobných věcí, vše však globálně, tudíž všechny závislosti jsou dostupné ve všech ostatních částí aplikace. V poslední řadě soubor "style.css"je zde pro tvoření designu dostupného všem komponentám. Také se zde nachází pohádkový obrázek, který jako pozadí doprovází každou stránku aplikace. Později byly přidány ještě další podsložky, například pro rozdělení stran pro učitele a žáky.

4.2.2 Backendová část

v backendové části, tedy ve složce "server"se nacházejí následující adresáře a soubor "index.js":

- **config** ⇒ obsahuje pouze adresu databáze a funkci pro porovnání hesel u přihlášení. Celková problematika však bude více vysvětlena v sekci Registrace a přihlašování



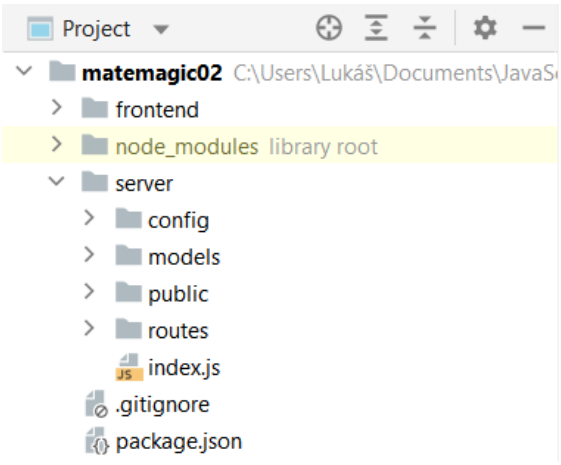
Obrázek 4.1: Organizace frontendové části v projektu

- **models** ⇒ obsahuje struktury dat, které jsou uložené v databázi, tedy uživatelů a slovních úloh.
- **routes** ⇒ obsahuje metody pro práci s daty, které jsou uchované v databázi. Pro práci s uživateli je na tomto místě i realizován celý registrační a přihlašovací systém.
- soubor **index.js** ⇒ obsahuje kód potřebný pro spuštění serveru. Také obsahuje všechny potřebné moduly pro spuštění serveru. V poslední řadě je zde realizováno připojení do databáze.

Obsah a funkce backendu budou podrobněji vysvětleny v sekci Popis backendové části.

4.3 Navigace a přechod mezi stranami v aplikaci

V tomto bodě, když je projekt založený a je v něm vytvořena přehledná a jasná struktura, bylo možné začít s programováním. Prvním úkolem v programování bylo zajistit přechod mezi stránkami, což s sebou přináší i různá omezení, jako například to, že na veškeré stránky (kromě stránky s přihlašováním a registrací), se lze dostat až po ověření uživatele. Ověření uživatele v tomto momentě ještě nebylo potřeba zcela řešit. Nicméně prvním krokem pro správu cest bylo vytvoření tzv. ”routeru”, který uchovává vlastnosti jednotlivých cest. Všechny cesty na stránky aplikace se vyskytují právě v souboru routeru. Dále bylo zapotřebí vyřešit jakým způsobem bude uživatel mezi stránkami přecházet. Zde se nabídl nápad vytvoření navigační lišty a tak byly vytvořeny hned 3 navigační lišty (jedna pro učitele,



Obrázek 4.2: Organizace backendové části v projektu

druhá pro žáky a třetí pro žáky v režimu hry). Nicméně později se naskytlo vylepšení převést navigační lištu pro učitele i pro žáka do jedné a jednotlivým cestám přidat atributy *role*, pomocí kterých se dané cesty zobrazí příslušným uživatelům. V poslední řadě byl ještě přidán atribut, který rozliší, zda se jedná o cestu ve hře nebo v matematické části.

```
{
    path:"/gameShop", component: gameShop,
    title: "Obchod",
    inGame:true,
    roles: ["student"],
    meta: {
        requiresUser: true,
        roles: ["student"],
    }
},

{
    path:"/addWordTask", component: addWordTask,
    title: "Přidat úlohu",
    inGame:false,
    roles: ["teacher"],
    meta: {
        requiresUser: true,
        roles: ["teacher"],
    }
},
```

Obrázek 4.3: Zobrazení vlastností cest

Ve chvíli, kdy byl už realizován systém přihlašování, byl po přihlášení uživatel uložen do tzv. ”local storage”. Local storage umožňuje data ukládat do webového prohlížeče pro konkretní zařízení. Pomocí toho byl načten uživatel, který už uchovává informaci o jeho roli. Pomocí funkce *filter* mu byly pak zpřístupněny dané cesty. Jinak řečeno v navigační liště byly vždy všechny cesty, ale uživatel viděl pouze ty, ke kterým má podle své role přístup. V poslední řadě bylo ještě zapotřebí zabránit tomu, aby se do aplikace nemohl dostat nepřihlášený uživatel nebo naopak, aby přihlášený uživatel nemohl odejít bez odhlášení. Toto zabezpečení bylo implementováno pomocí funkce, která podle atributy cesty *meta* rozpozná, zda je uživatel přihlášený či nikoliv. Pokud je tedy uživatel přihlášený a pokusí se dostat na stránku pro přihlášení či registraci, tím že zadá adresu stránky ručně, automaticky ho aplikace odkáže na stránku svého profilu. V opačném případě, pokud uživatel přihlášený není a pokusí se zadat některou z adres aplikace, bude odkázán na stránku s přihlášením.

4.4 Procvičování a počítání příkladů

Tato stránka v první řadě řeší, kolik cvičení žák v tento den absolvoval. Existuje zde totiž omezení, že každý žák může za den absolvovat pouze 5 cvičení. To je z toho důvodu, aby žák pouze neseděl u počítače a nenaháněl si mince. Pokud tedy žák ještě v tento den neabsolvoval 5 cvičení, zobrazí se hned okno s prvním příkladem. Nad ním je ještě uveden počet, kolik cvičení žák již v tento den absolvoval a pod ním je uveden počet příkladů v tomto cvičení, který se pro každý ročník liší. Žák tedy vidí příklad a pod ním prostor, kam může zadat výsledek. Poté musí kliknout na tlačítko *Další příklad* a stejným způsobem se hned zobrazí další příklad. Až žák takto vypočítá daný počet příkladů, cvičení se uzavře a kliknutím na tlačítko *Vyhodnotit* se žák může podívat jak úspěšně počítal, tedy zobrazí se další okno s příklady, které měl špatně, pokud takové byly a jejich správné výsledky. Dále se podle toho i zobrazí odměna, jakou žák dostane. V případě, že žák měl alespoň 90% příkladů správně, tato odměna odpovídá vzorci:

$$odmena(stribrneMince) = rocnikStudenta * 3.$$

Pokud měl správně alespoň 75% ročník se násobí pouze dvěma. Pokud měl správně alespoň 60% ročník se násobí pouze jedničkou. V poslední řadě se může stát že žák měl více než 60% příkladů špatně, v takovém případě nedostává odměnu žádnou. Jakmile si žák prohlédne své špatně vypočítané příklady, může přejít na další cvičení, které se zobrazí stejným způsobem jako to předchozí, pokud tedy ještě nedosáhl pátého cvičení v tento den.

I tato komponenta se skládá z více komponent i částí pro zobrazení příkladů. Při vstupu na tuto stránku je binární atribut, který určuje zda je cvičení kompletní, nastaven na *false*. O zadávání příkladů se stará komponenta "math-problem". Typů příkladů je hodně a každý se zobrazuje trochu jiným způsobem, například: příklad na porovnání bude mít jiné atributy než příklad na počítání apod. Přesněji řečeno příklad na porovnání obsahuje dvě čísla a prostor pro výsledek je tedy pro znaménko '>', '<' nebo '=' se. Oproti tomu příklad na počítání obsahuje dvě čísla, mezi nimi operátor, tedy '+', '-', '·' nebo '÷' a prostor pro výsledek je až pod ním. Proto by bylo složité a nepřehledné všechny typy příkladů uchovávat v jedné komponentě. Atribut, který určuje, zda je cvičení kompletní se změní na *true* v momentě, kdy žák klikne na tlačítko pro vyhodnocení výsledků a podle toho se také zobrazí příslušný text, v tomto případě příklady, které byly špatně. O generování příkladů se stará samostatná třída "ExampleGenerator", která obsahuje všechny algoritmy pro generování příkladů a která je v této komponentě také volána. Generování funguje tak, že pro každý ročník byla napsána samostatná metoda, která si jako parametr bere počet příkladů. Dále jsou v ní volány metody na generování daných typů příkladů, které se generují dokud není dosažen limit počtu příkladů. Nakonec jsou příklady ještě promíchány, aby se například nezobrazovaly nejdříve příklady na sčítání, pak na odčítání apod. Při generování příkladů se ještě generují náhodná čísla, odpovídající rozsahu v daném ročníku. Například tedy pokud je žák v prvním ročníku, počet příkladů pro jeho cvičení je 15 a příklady jsou zde na: sčítání, odčítání a porovnávání. Konkrétní metoda pro první ročník obsahuje *while cyklus*, ve kterém se vygeneruje první příklad na sčítání, druhý na odčítání, třetí na porovnávání a čtvrtý zase na sčítání. Tímto způsobem cyklus pokračuje, než dosáhne patnácti příkladů a příklady se uloží do pole. Pak se provede ještě zamíchání příkladů v poli. Každý příklad v poli je uložen jako objekt, který obsahuje atributy, které se však liší podle typů příkladu. Příklad na sčítání pro první ročník by mohl vypadat, například takto: *message:"Vypočítej"*, *type:"calculate"*, *first_number:"1"*, *second_number:"7"*, *operator:"+"*, *result:"8"*, *your_result:"0"*, *completed:"false"*. Do atributu *your_result* se uloží výsledek žáka, který zadá při počítání a atribut *completed* se tím nastaví na *true*. Stejným způsobem fungují metody i pro vyšší ročníky, kde se liší pouze typy a počet příkladů. Kompletní typy příkladů pro dané ročníky je zobrazen v tabulce v sekci 4.4.

4.5 Operace se slovními úlohami

Kromě počítání příkladů mají žáci možnost řešit slovní úlohy. Tyto slovní úlohy jsou zadávány učiteli. Aby žák měl k dispozici slovní úlohu, musí ji nejdříve učitel zadat.

Typ příkladů	I.	II.	III.	IV.	V.
Sčítání	✓	✓	✓	✓	×
Odčítání	✓	✓	✓	✓	×
Násobení	×	✓	✓	✓	×
Dělení	×	×	✓	✓	×
Porovnávání	✓	×	×	×	×
Zaokrouhlování	×	✓	✓	✓	×
Dělení se zbytkem	×	×	×	✓	×
Převod arabských čísel na římská čísla	×	×	×	×	✓
Převod římských čísel na arabská čísla	×	×	×	×	✓
Sčítání desetinných čísel	×	×	×	×	✓
Odčítání desetinných čísel	×	×	×	×	✓
Násobení desetinných čísel	×	×	×	×	✓
Dělení desetinných čísel	×	×	×	×	✓

Tabulka 4.1: Přehled typů příkladů, které se generují pro každý ročník

4.5.1 Možnosti učitele

Pokud se do aplikace přihlásí učitel, aplikace ho odkáže na stránku pro zadávání úloh. Aby mohl učitel přidat novou slovní úlohu musí zadat následující vlastnosti:

1. **Text slovní úlohy**. Sem je nutné napsat celé znění slovní úlohy, do příslušného textového pole.

2. **Ročník žáka, pro jaký je úloha určena**. To je umožněno, pomocí tzv. html tagu *select*. Ten umožňuje vybrat si jednu z nabízených možností. V tomto případě nabízí ročník 1 až 5.

3. **Správný výsledek úlohy**. Zde je vstupní element pro text, ale zadat lze pouze čísla, buď tím způsobem, že učitel číslo zapíše ručně nebo vybere pomocí šipek (nahoru nebo dolu). Žák tedy bude muset do výsledku zadat stejné číslo, které určí učitel jako výsledek.

4. **Odměna za úlohu**. Odměnu učitel zadá také jako číslo, stejným způsobem jako výsledek. Odměna je zde ve formě stříbrných mincí, tudíž pokud učitel stanoví, že za úlohu bude odměna 23. Tato odměna bude odpovídat dvěma zlatým a třem stříbrným mincím. Zde bylo ještě přidáno omezení takové, že učitel může zadat maximální velikost odměny desetkrát větší než je ročník žáka, pro který je daná úloha určená. Například pokud učitel zadává úlohu pro 1.ročník, její maximální odměna je 10 mincí.

V momentě, kdy učitel vyplní všechna příslušná data úlohy, musí kliknout na tlačítko *Přidat úlohu*. Poté se zavolá metoda, ve které se vytvoří objekt, do kterého se uloží všechny zadané atributy. Dále se ještě vytvoří atribut výsledek žáka, do kterého se zatím uloží nulová hodnota a také se přidá do tohoto objektu. Tento atribut bude později důležitý, protože se do něho uloží výsledek, který zadá žák při řešení úlohy. Po vytvoření objektu se objekt přidá do lokálního úložiště projektu, aby učitel mohl úlohu vidět a případně ji upravit či vymazat. V poslední řadě se objekt odešle do databáze.

Kromě přidávání nových úloh má učitel možnost spravovat již existující úlohy, které dříve zadal on nebo jiní učitelé. Pokud v navigační liště klikne na *spravovat úlohy*, vstoupí na stránku, kde vidí všechny slovní úlohy pro všechny ročníky, včetně všech atribut u každé úlohy. Tyto úlohy jsou číslovány a seřazeny podle ročníků. Zde se vyskytují tlačítka *Upravit* a *Smazat*. Při kliknutí na tlačítko *Upravit* se přesune do podobné stránky jako při zadávání nové slovní úlohy. Tlačítko ho ve skutečnosti nepřesune na jinou stránku, ale pouze se změní binární atribut, který určuje zda je uživatel v režimu úprav, či ne a podle toho se zobrazí příslušný obsah stránky. I zde se vyskytují stejné možnosti jako při pro zadávání nové úlohy s tím rozdílem, že v každém vstupním prvku jsou aktuální data úlohy, které lze přepsat. Po upravení jedné nebo více vlastností, musí svou úpravu potvrdit kliknutím na tlačítko *Dokončit úpravy*. Poté se přesune zpět na stránku, kde lze opět vidět všechny úlohy, tedy binární atribut pro zobrazení obsahu se změní zpět. Při kliknutí tlačítko *Smazat* se úloha okamžitě smaže jak z lokálního úložiště projektu tak i z databáze.

4.5.2 Možnosti žáka

Pokud žák přejde na stránku slovních úloh, zobrazí se mu okno se slovní úlohou odpovídající jeho ročníku. V okně se nejdříve zobrazí zadání této úlohy a pod ním je prostor, kam žák zapíše svůj výsledek, který následně musí potvrdit zeleným tlačítkem hned vedle. Tento výsledek se uloží do již zmíněné hodnoty, která se vytvořila při zadání této slovní úlohy a do této chvíle měla nulovou hodnotu. Tato komponenta obsahuje dvě binární proměnné, které určují, zda se jedná o fázi řešení úlohy či odevzdání úlohy. Jakmile žák klikne na tlačítko pro potvrzení svého výsledku, je žákův výsledek porovnán s výsledkem, který zadal učitel a v případě, že je výsledek správný, respektive stejný jako zadal učitel, žákovi se přičtou peníze. Zároveň se změní hodnota binární proměnné a podle toho se změní i obsah na této stránce, který oznámí zda je řešení správně nebo špatně. Následně se uživatele zeptá, zda chce zadat další úlohu. Kliknutí na tlačítko pro další úlohu se opět změní hodnota binární proměnné a zobrazí se stejným způsobem další úloha. Ve spodní části okna je ještě vypsaná odměna za úlohu.

4.6 Závěrečný test

K závěrečnému testu může žák vstoupit, až pokud absolvuje alespoň 40 cvičení v minimální úspěšností 85 %. To je zde realizováno pomocí kontrolní komponenty, která je zde volána, podle toho, zda má či nemá splněné podmínky. Pokud tedy nemá splněné tyto podmínky a vstoupí na tuto stránku, zobrazí se karta, informující ho, o této podmínce, ale i počet cvičení, které splnil a jeho úspěšnost. Pokud však vstoupí na tuto stranu a podmínky má splněné. Ihned se mu zobrazí test. Test je složený z náhodně generovaných příkladů, stejným způsobem jako cvičení, s tím rozdílem, že příkladů je zde padesát. Kromě příkladů se zde vyskytují ještě 2 slovní úlohy od učitelů. Procentuální hodnocení je následující: váhá všech příkladů je dohromady 80% a váha slovních úloh je 20%. Aby žák test splnil a mohl tak postoupit do dalšího ročníku, musí mít, úspěšnost alespoň 90%. Pokud ji mít nebude, může test absolvovat znovu. Pokud však odejde od rozepsaného testu na jinou stranu, test se uloží "local storage"a může se k němu vrátit. Pokud se však odhlásit, musí celý test absolvovat znovu.

4.7 Herní část

Do této části se může žák dostat až od 3.ročníku. Pokud je tedy žák ve třetím a vyšším ročníku na hlavní straně aplikace, tedy na straně svého profilu se zobrazí tlačítko *Vstoupit do hry*. Jakmile vstoupí do hry, přejde do úplně nové strany aplikace s novou navigační lištou. Žák se ocitá ve hře, kde mu je přidělena postava Jakuba, kterého musí vylepšovat, aby se mohl posouvat v příběhu dál. Žák má tedy následující možnosti:

- **Přehled** ⇒ na této stránce může vidět celkové informace o své postavě, včetně schopností a předmětů postavy Jakuba.
- **Cvičiště** ⇒ zde si může vylepšovat schopnosti postavy, kterými jsou: Síla, Útok, Obrana a Výdrž
- **Obchod** ⇒ zde si může kupovat předměty pro svoji postavu z kategorií: Zbraně, Helmy a Brnění
- **Příběh** ⇒ zde je realizován příběh, v podobě otevřené knihy
- **Odejít ze hry** ⇒ slouží pro návrat zpět do části s matematikou.

4.7.1 Přehled

Přehled je první strana, kterou žák vidí, jakmile vstoupí do hry. Hned nahoře v levém rohu je znázorněna karta Jakuba. Nejdříve tedy obrázek Jakuba, který má ve třetí ročníku podobu sedláka a v dalších ročnících už podobu rytíře, což odpovídá tomu, jak žák v příběhu postupuje. Pod obrázkem jsou vypsané schopnosti této postavy, tedy: Síla, Útok, Obrana a Výdrž, které odpovídají tomu, jak je žák vylepšuje v sekci *Cvičiště*, ale také tomu, jaké má vybavené předměty. Vybavené předměty, lze vidět v další kartě, hned vedle této karty. Jednotlivé předměty se zobrazují v dalších samostatných kartách, tím způsobem, že nejdříve je uveden Název předmětu. Pod názvem je obrázek, odpovídající tomuto předmětu a v poslední řadě pod obrázkem je tlačítko pro odvybavení předmětu. Při kliknutí na toto tlačítko se předmět odvybaví a přesune se do inventáře, který je na další kartě, ve spodní části této stránky. Položky v inventáři jsou zobrazeny úplně stejným způsobem jako položky vybavené, s tím rozdílem, že místo tlačítka pro odvybavení, je zde tlačítko po vybavení předmětu.

Princip vybavování předmětu funguje tak, že každá položka má atribut *equip*, který je při zakoupení nastaven na *false*. Jakmile tedy žák klikne na tlačítko pro vybavení, tento atribut se nastaví na *true*. Tato změna se provede jak v "local storage", tak i v databázi. Položky v inventáři jsou tedy všechny, které mají tento atribut nastaven na *false*. Při opětovném vstupu na tuto stránku tedy nejdříve proběhne kontrola, jestli nějaká položka není nastavená na *true*. Z každé kategorie helm, zbraní a brnění, může být aktivovaná vždy pouze jedna položka. Pokud tedy postava má například již aktivovanou helmu a aktivuje jinou helmu, bez toho, aby tu předchozí odvybavil, položka se automaticky změní.

4.7.2 Trénink schopností

Na této stránce si žák může vylepšovat své schopnosti za získané mince. Tyto schopnosti jsou zde vypsány pod sebou v pořadí: Síla, Útok, Obrana a Výdrž. Přičemž vedle názvu schopnosti je vždy její hodnota, vedle ní tlačítko '+' pro vylepšení a vedle je uvedena cena. Cena je vždy pětinásobkem hodnoty dané schopnosti ve stříbrných mincích, bez započítání aktivovaného předmětu. Při zobrazení je však proveden přepočet na zlaté a stříbrné mince. Tedy pokud má například útok hodnotu 5 je zobrazena cena dvou zlatých a pěti stříbrných mincí.

Hodnoty jednotlivých schopností jsou načteny z "local storage", ale před samotným zobrazením je ještě provedena kontrola, jestli není aktivovaná některá položka. Tedy, aby cena vylepšení odpovídala pětinásobné hodnotě schopnosti bez vylepšení u předmětu. Po kliknutí na tlačítko plus, se provede změna jak v "local storage"tak

Obrázek 4.4: Ukázka přehledu postavy

i v databázi. Stejným způsobem se mu odečtou i peníze. Každou schopnost je možné vždy při jednom kliknutí vylepšit pouze o jednu jednotku.

4.7.3 Obchod

Na této stránce si žák může kupovat nové předměty, ale i prodávat ty, které už vlastní, avšak za předmět, který žák prodá, dostane pouze poloviční cenu, než za jakou ho kupoval. Předměty v obchodě jsou rozdělené do tří kategorií, kterými jsou: zbraně, helmy a brnění. Jednotlivé předměty jsou zobrazeny stejným způsobem jako předměty v inventáři na stránce profilu. Tato stránka obsahuje v horní části čtyři tlačítka. První je tlačítko *Prodat předměty* a pokud na něj hráč klikne, stav binární proměnné *buyuing* se změní na *false* a na stránce se objeví pouze předměty, které hráč vlastní. Zároveň se změní text tlačítka na *Koupit předměty*. Při opětovném kliknutí se objeví zase všechny předměty v obchodě. Zobrazené jsou vždy jen předměty dané kategorie a v základním režimu jsou zobrazené pouze zbraně. Pokud tedy bude chtít hráč koupit předmět z jiné kategorie, například helmu, musí kliknout na jedno ze tří tlačítek pro zobrazení předmětů z jiné kategorie, například pokud si bude chtít koupit helmu, musí kliknout na tlačítko *Helmy* a na stránce se zobrazí pouze helmy. Pokud u daného předmětu klikne na tlačítko *Koupit* program v první řadě zkontroluje, zda má hráč dostatek peněz, ale také pokud tento předmět už nevlastní. Podobný způsob funguje i pokud bude chtít nějaký předmět prodat, zde se však neřeší to jestli má dostatek peněz, ani to, jestli již neprodal, protože každý předmět může mít hráč pouze jednou. V pravém

horním rohu je ještě navíc zobrazen počet peněz, který hráč má.

Obrázek 4.5: Ukázka ze stránky obchodu

4.7.4 Strana příběhu

Při vstupu na tuto stránku se zobrazí 2 okna, které reprezentují stránky otevřené knihy. Na pravé straně knihy je napsán text příběhu. Aby byl zobrazen příběh odpovídající ročníku žáka, musí žák kliknout na jedno ze tří tlačítek, které vyjadřují ročník, tedy třetí až pátý ročník. Pro každý ročník, je totiž napsána jiná část příběhu. Tato tlačítka se nacházejí nad touto otevřenou knihou. Při prvním vstupu je na druhé straně napsána nápověda, aby si žák vybral část příběhu, nejlépe podle svého ročníku. Jakmile žák klikne na tlačítko ročníku, na stránce knihy se objeví příběh a pomocí šipek může listovat v příběhu dál. Jakmile se však objeví část, kde je nucen porazit nepřítele, aby mohl pokračovat v příběhu dál, příběh se v tomto bodě zastaví a zmizí i šipka doprava. Žák tedy musí porazit nepřítele, aby mohl pokračovat. Tento proces je realizován pomocí speciálních znaků v textu, kde znak ';' vymezuje odstavce v příběhu, ve kterém se příběh zastaví, než žák znovu

klikne na jednu ze šipek. Dále znak '@' v textu označuje nepřátele a po porážce daného nepřítele se tento znak v textu nahradí, aby s nim žák nemohl bojovat znovu. Nepřítel se tak uloží do pole poražených nepřátel, které je uchováno u každého uživatele. Při dalším listováním v příběhu tedy proběhne kontrola, kolik nepřátel již žák porazil a podle toho se nahradí příslušný počet těchto znaků. Pokud tedy žák během příběhu narazí na nepřítele, příběh se zde zastaví a pod knihou se zobrazí karta nepřítele s tlačítkem *Útok*. Při kliknutí na toto tlačítko je žák přesunut do jiné komponenty, ve které může vidět dvě karty se schopnostmi svých a nepřítele, ale také zda zvítězil nebo prohrál a může se vrátit zpět k příběhu. V případě, že žák prohraje, nezbývá mu nic jiného, než si vylepšit své schopnosti a utkat se s ním znovu. Nepřátelé jsou uloženy v úložišti aplikace a text je uložen v samostatných textových souborech a jakmile žák klikne na tlačítko ročníku, načtou se nepřátelé z daného ročníku, stejně tak se načte i příslušný textový soubor pro příběh.

4.7.5 Tvorba nepřátel

Data všech nepřátel jsou uloženy v již zmíněném "store". Jsou zde rozděleny do tří polí, podle toho, v jakém ročníku se vyskytují. Každý nepřítel je v aplikaci uložen jako objekt, obsahující následující parametry: *name*, *strength*, *attack*, *defense*, *hp*, *year*, *money_reward*, *link* viz 4.6. Hodnoty atributů *strength*, *attack*, *defense*, (síla, útok, obrana a životy), jsou generovány tak, že každému nepříteli je přidělená určitá hodnota, která by měla odpovídat jeho kvalitě. Tato hodnota však není pevná a při volání nepřítele v programu, se nepříteli přidělí hodnota v rozmezí -20% až +20%, k jejímu atributu. Pokud je tedy například určeno, že nepřítel bude mít hodnotu síly 10, tak skutečná hodnota jeho síly může být minimálně 8 a maximálně 12.

```
enemies_from_4:[

    {
        "name":"Nepřátelský rytíř",
        "strength": randomize_enemy_abilities( number: 11),
        "attack": randomize_enemy_abilities( number: 15),
        "defense": randomize_enemy_abilities( number: 15),
        "hp": randomize_enemy_abilities( number: 12),
        "year":4,
        "money_reward":50,
        "link":'assets/imgs/enemies/enemies_4/rytir.png'
    },

    {
        "name":"Nepřátelský lučišník",
        "strength": randomize_enemy_abilities( number: 13),
```

Obrázek 4.6: Ukázka uložení nepřátel v aplikaci

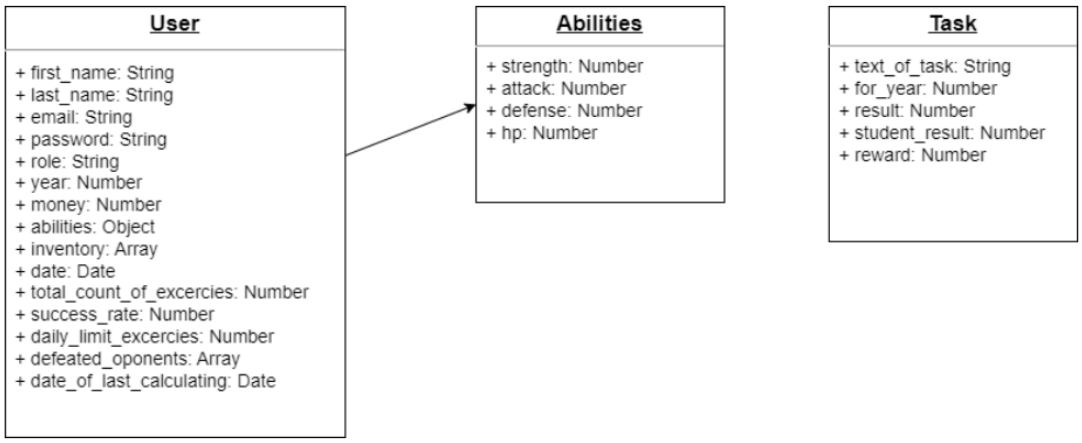
4.8 Popis backendové části

V následující sekci bude více popsána funkčnost backendové části. Tato sekce bude obsahovat, přehled struktury dat, popis metod pro práci s nimi a problematiku hesel, s čímž je spojená registrace, přihlašování a připojení k databázi. Propojení backendové a frontedové části bude popsáno až v sekci Propojení backendu a frontendu.

4.8.1 Příprava dat

Pro připojení k databázi a pro práci s ní bylo nejdříve potřeba rozvrhnout, co všechno bude potřeba uchovávat. Nejdříve byly uchovávány v databázi 3 typy objektů, kterými byly: Uživatel, Slovní úloha a Položka. Avšak položku nebylo nutné uchovávat v databázi, protože po jejím vytvoření už se její data nijak nemění. A tak byly vytvořeny modely pouze dva. Modely jsou v aplikaci definovány stejným způsobem jako na obrázku níže. U objektu *User* jsou atributy: year, money, abilities,

inventory, date, total_count_of_excercies, success_rate, daily_limit_excercies, defeated_oponents, completed_word_tasks, date_of_last_calculating, nastavené jako nepovinné. Nepovinné jsou z toho důvodu, že pokud se uživatel zaregistruje jako učitel. V tu chvíli tyto atributy nepotřebuje a tudíž se ani nevytvářejí. V poslední řadě je každý model zařazen do kolekce pro přípravu odeslání do databáze.



Obrázek 4.7: Ukázka struktury dat (Uživatel, uživatelovi hráčské dovednosti a slovní úloha)

4.8.2 Vytváření kolekcí

V minulé sekci byly popsány modely a struktury dat. V této sekci budou popsány jednotlivé metody, které bylo potřeba vytvořit pro práci se těmito daty v databázi. Pro práci s každým modelem dat tj. Uživatel a Slovní úloha, byla vytvořena tzv. "routa". V každé této routě se odkazuje na model dat a dále jsou zde naprogramovány metody pro práci s těmito daty. V poslední řadě jsou tyto modely připraveny pro další využití na dalším místě programu.

Kolekce slovních úloh

V této routě byly vytvořeny následující metody:

- **Metoda GET** ⇒ v této metodě se pouze řeší, zda dokáže najít všechna data, daného typu, v tomto případě slovních úloh. Pokud je najde předá je

programu pro další využití s návratovým kódem 200, který znamená, že vše proběhlo v pořádku. V opačném případě je odeslán chybový kód 500 a zpráva o chybě

- **Metoda POST** ⇒ Zde je načten požadavek pro poslání, tedy přidání nové slovní úlohy. Úloha je dále poslána mezi ostatní slovní úlohy. V případě, že vše proběhne v pořádku, tedy že se úloha odešle do databáze, je opět odeslán i návratový kód 200. V opačném případě je odeslán znovu kód 500.

- **Metoda PUT** ⇒ Tato metoda složí pro úpravu dat v databázi, v tomto případě pro úpravu slovní úlohy a kódy zde fungují stejným způsobem, jako v předešlých dvou metodách. Metoda PUT se liší pouze v tom, že je zde navíc parametr *id*, který určuje pro jakou slovní úlohu se musí data změnit. Tedy kromě parametru *id*, je zde ještě načten nový obsah těchto dat.

- **Metoda DELETE** ⇒ Metoda DELETE funguje stejně jako předchozí metoda PUT, ale není zde potřeba posílat data pro změnu. Úloha najde odpovídající úlohu podle parametru *id* a celou ji smaže z databáze.

Kolekce uživatelů

V této routě se vyskytují metody pro práci s uživateli. Kromě metod: PUT (pro úpravu dat o uživateli) a GET (pro získání všech uživatelů), které fungují stejným způsobem jako u slovních úloh, se zde vyskytují ještě další metody, které bylo potřeba naprogramovat.

- **Metoda POST s parametrem cesty register** ⇒ v této metodě jsou v první řadě načteny všechny údaje o uživateli, včetně těch, které uživatel při registraci nezadává, ale program je nastavuje sám, jako například uživatelovy peníze. Peníze by uživatel při registraci rozhodně zadávat neměl. Dále už se jen řeší validace těchto dat. V první řadě se porovná heslo, které uživatel zadá s potvrzením hesla, které při registraci zadává také. Tato hesla by měla být napsaná uživatelem totožně. V případě, že nejsou totožná program odešle chybový kód 400 a zprávu o chybě. Tento kód funguje stejně jako kód 500, ale navíc definuje, že se chyba stala na straně uživatele, tedy některý údaj byl zadán špatně. Dále pokud jsou hesla totožná, metoda řeší, jestli byly skutečně zadány všechny údaje potřebné pro registraci těmi jsou: jméno, příjmení, mail, heslo a role (v případě role žáka musí být zadán ještě ročník žáka). Pokud je některý údaj nezadaný, znovu se odešle kód 400, včetně chybové zprávy. V opačném případě program pokračuje dál a řeší už pouze to, zda mail již není používaný. Chybový stav v tomto bodě je zpracován stejným způsobem, jako v předchozích dvou bodech. Pokud je i tato

podmínka splněna, metoda vytvoří nového uživatele do databáze. Ještě před uložením se, ale zašifruje jeho heslo, pomocí tzv. funkce *bcrypt*, která bude více popsána v sekci Registrace a přihlašování

- **Metoda POST s parametrem cesty login** ⇒ v této metodě jsou opět načteny údaje, které uživatel zadal, těmi jsou zde jen mail a heslo. V první řadě se tedy ověří pokud je mail správný a případně se opět odešle chybový kód 400, tedy chyba na straně uživatele. Pokud kontrola mailu projde v pořádku, proběhne pomocí funkce *bcrypt* kontrola zašifrovaného hesla s heslem, které uživatel právě zadal. Pokud je i tato kontrola v pořádku, uživateli se nastaví přístupový token a je mu umožněn přístup do aplikace. Tento token mimo jiné také určuje dobu přístupu, v případě této aplikace, má každý uživatel po přihlášení expiraci dvou dnů nebo dokud se sám neodhlásí. Expirace je zde však určena v sekundách.

Parametry cesty *register* a *login*, pouze určují to, že tyto procesy probíhají na určitých adresách. Uživatel k nim však nemá přístup.

4.8.3 Spuštění serveru

Celý proces spuštění serveru probíhá v souboru "index.js", který je uložen v backendové části programu. V tomto souboru jsou v první řadě importovány veškeré knihovny a moduly potřebné pro všechny operaci na straně backendu, jako například moduly pro korektní připojení k databázi, korektní zpracování dat nebo moduly pro zahrnutí datových modelů pro databázi. Dále je tu realizován postup připojení do databáze. Pro připojení do databáze bylo nejprve potřeba vytvořit si účet na account.mongodb.com. Dále bylo potřeba si zde vytvořit konkrétní databázi, do které se aplikace připojí. To bylo umožněno tím, že se vygeneroval odkaz, který byl přidán do programu. Dále pak bylo ještě potřeba vytvořit si heslo, kterým pak byla nahrazena část v tomto odkazu. Bez hesla tak není možné se k databázi dostat. V poslední řadě je v tomto souboru už pouze část se spuštěním serveru. Zde bylo potřeba odlišit pouze to, zda je aplikace zveřejněna a pokud ne, adresa backendu se nastaví na *http://localhost:5000*. V opačném případě je odkaz určen službou, kde se aplikace zveřejní.

4.9 Propojení backendu a frontendu

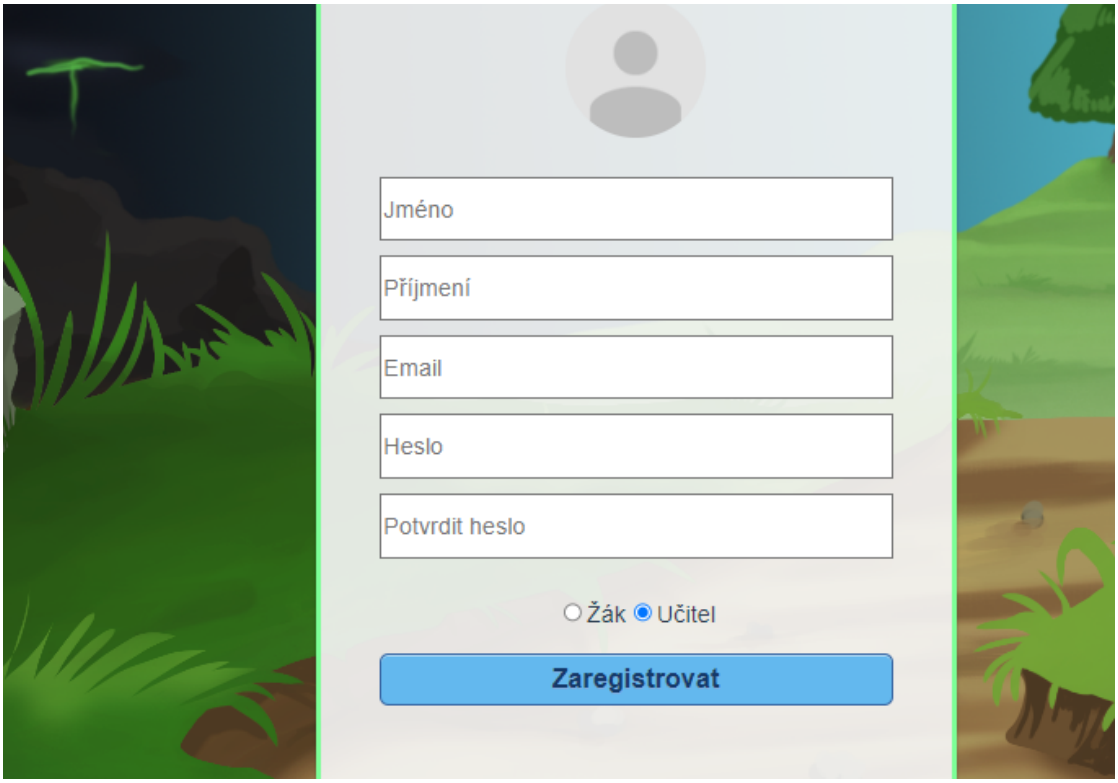
Hlavní roli v propojení backendu s frontendem hraje tzv. "store", který v podstatě uchovává stav aplikace. Lze říci, že v této aplikaci funguje jako prostředník pro práci s daty mezi jednotlivými komponenty a backendem. V tomto souboru jsou definovány bloky:

- **state** ⇒ Slouží pro uchovávání stavů, pro celou aplikaci. V této aplikaci bylo potřeba uchovávat zejména uživatele, který je po přihlášení uložen do ”local storage”.
- **getters** ⇒ Slouží pro získávání atributů. Lze tak získávat například konkrétní atributy z tohoto obchodu. V této aplikaci se ve store určuje například, jestli je uživatel přihlášen.
- **mutations** ⇒ Jediným způsobem jak provést změnu atributů v tomto store, je provedením tzv. mutace. Tímto způsobem tak můžeme měnit stavy všech atributů v celé aplikaci, včetně uživatelů nebo slovních úloh. V jednotlivých komponentách se tak aplikace odkazuje pravě na tyto mutace zde uložené. V těchto mutacích se provede změna, která se pak rovnou i na tomto místě odešle do databáze, v případě změny v údajích uživatele se uloží i do ”local storage”. Například pokud si ve hře žák zakoupí předmět, komponenta se odkáže právě na jednu z těchto mutací, kde se pak provedou změny. Tímto způsobem jsou zde uloženy mutace pro všechny změny atributů v celé aplikaci, které je potřeba měnit i v databázi.
- **actions** ⇒ Akce jsou velmi podobné mutacím. Avšak lze se v nich odkazovat i na mutace. V opačném případě to nelze, ani mutace se nemohou odkazovat na jiné mutace. V této aplikaci tak existují akce pro registraci nebo pro přihlášení, kde se do ”local storage”uloží token i přihlášený uživatel.

4.9.1 Registrace a přihlašování

Proces registrace

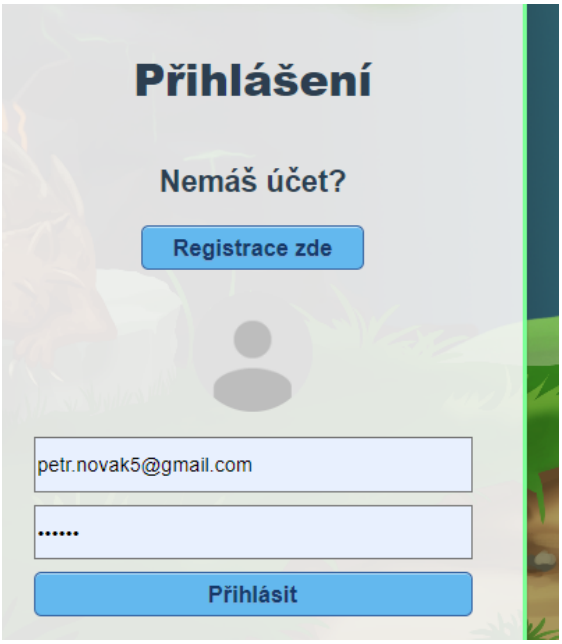
Proces registrace začíná v komponentě ”registerPage”, kde uživatel musí vyplnit všechny potřebné údaje. Podle zvolené role se zde vytvoří buď učitel nebo žák a podle toho i všechny ostatní potřebné údaje. Zde je volána akce právě ze ”store”. která posílá data metodou POST, s parametrem cesty *register*, což je metoda routy, která byla zmíněná v sekci Kolekce uživatelů. V této metodě POST, po úspěšné kontrole hesel, vyplnění všech údajů uživatelem a zadáním mailu, který ještě není registrovaný, proběhne šifrování hesel. To probíhá pomocí knihovny *brcypt*, která mimo jiných funkcí obsahuje právě funkce *genSalt* a *hash*. V první řadě se tedy zavolá funkce *genSalt*, která bere jako parametr číslo. Toto číslo určuje dobu šifrovacího algoritmu a zároveň stupeň zabezpečení hesla. Vytvoří se tzv. sůl, která se pak využívá ve funkci *hash*, která pomocí této soli šifruje heslo. Jakmile je uživatel zaregistrován, zobrazí se zpráva oznamující úspěšnou registraci a přesune se na stránku pro přihlášení.



Obrázek 4.8: Okno s registrací

Proces přihlášení

Proces přihlášení začíná v komponentě ”loginPage”, kam je uživatel přesunut po úspěšné registraci, ale také je to první strana, která se zobrazí při vstupu do aplikace. K úspěšnému přihlášení zde stačí zadat správný mail a heslo. Zadávání je umožněno přímo v této komponentě, ale je zde volaná akce *login*, pomocí, které se data přesouvají do ”store”. Tam se pak dále posílají pomocí metody POST s parametrem cesty ”login” na routu, která se stejně jako při registraci nachází na backendu. V této metodě, po úspěšné kontrole mailu, který zadal uživatel, se provede metoda *compare*, kterou také obsahuje knihovna *bcrypt*. V této metodě se pouze porovná heslo, které uživatel zadal v komponentě ”loginPage”se zašifrovaným heslem v databázi. Metoda si sama heslo rozšifruje. Pokud se hesla shodují, aplikace se vrací zpět do ”store”, kde se do ”local storage”uloží daný uživatel a ověřovací token. Uživatel se v této chvíli přesouvá na stránku svého profilu.



Obrázek 4.9: Okno pro přihlášení

4.10 Doplňky aplikace

V aplikaci byly vytvořeny některé doplňky a vylepšení, zejména proto, aby uživatel měl přehled o stavu v aplikaci, ale i o jeho možnostech. Byla tak vytvořena okna s nápovědou a okna s chybovými stavy.

4.10.1 Okno s nápovědou

Okno s nápovědou slouží k tomu, aby uživatel vždy věděl, co vše na dané stránce může dělat a jakým způsobem. Například pokud žák přejde na stránku pro slovní úlohy, zobrazí se v horní části okno, které popisuje jak cvičení i získávání odměn fungují. Toto okno je vždy možné zavřít. Tato okna jsou vytvořena jako samostatné komponenty a volány v jiných komponentách, s tím, že v každé komponentě je do nich přesunut jiný příslušný text s nápovědou. Takto jsou vytvořena okna s nápovědou téměř pro každou stránku v aplikaci. Okna mají zelené pozadí, v horní části se vyskytuje křížek pro zavření okna, pod ním nadpis *nápověda* a dole text nápovědy.

4.10.2 Okno s chybovými stavy

Okna s chybovými stavy se v aplikaci vyskytují také jako samostatné komponenty, ale na rozdíl od nich se text uchovává přímo ve ”store”. Vyskytují se hlavně na stránkách pro přihlášení a registraci, kde uživatele informují v případě, že zadá některý atribut špatný. Oproti oknům s nápovědou mají červené pozadí bez nadpisu *nápověda*. I zde se vyskytuje křížek pro zavření okna.

4.11 Nasazení aplikace do provozu

První řadě při nasazení aplikace do provozu, bylo potřeba změnit adresu dat, uložených na backendu. Zjednodušeně by se dalo říct, že stačilo změnit adresu z *http://localhost:5000* na */api/...* (následující cesty se liší podle kolekcí v databázi). Dále bylo potřeba z terminálu zavolat příkaz *npm run build*, čímž se vytvořila složka pro ukládání statických souborů. Dále bylo potřeba tuto složku zahrnout v aplikaci. To bylo nastaveno v souboru ”index.js”, ve kterém ve vyskytuje i spouštění serveru. Pak bylo potřeba zvolit si nějak hostující službu, na které může být aplikace spuštěna. Hostující službou pro tento projekt bylo vybráno Heroku [1], kde bylo potřeba nejprve si založit účet a následně i založit projekt, do kterého byl uložen zdrojový kód aplikace. Přes příkazový řádek bylo provedeno přihlášení do Heroku a přes řadu příkazů, provedených také v příkazovém řádku, byl zdrojový kód odeslán a následně spuštěn v této službě.

[1] Heroku je hostující služba, která podporuje několik programovacích jazyků. Byla založena v roce 2007 a v té době podporovala pouze jazyk Ruby. Dnes je jednou z nejrozšířenějších služeb tohoto typu a v současné době podporuje jazyky: Java, Node.js, Scala, Clojure, Python, PHP, a Go.

5 Testování aplikace v provozu

Po dokončení byla aplikace zveřejněna a následně testována různými uživateli. Na základě testování byly získány následující poznatky. V první řadě byl problém s registrací a přihlašováním pro uživatele prvních a druhých ročníků. Zde by se nabízela možnost, při registraci si vytvořit určitý avatar a přihlašovat se prostřednictvím tohoto avataru. Tento způsob, by ale nebyl dostatečně bezpečný a tak by se musel navrhnout jiný kompromis. Uživatelé dále doporučovali interaktivnější příběh, například putováním po mapě, kde by se hráč posouval v rámci příběhu blíže k cíli. Dále by bylo vhodné zařadit do stránky obchodu možnosti filtrování například: podle ceny apod. V obchodě by také položky mohly být zobrazeny dvě vedle sebe, čímž by se zmenšil obsah stránky. Několik uživatelů také doporučovalo přidat průběh souboje, tak aby hráč neviděl pouze celkový výsledek, ale i výsledky jednotlivých kol.

5.1 Možnosti pro vylepšení od autora

Jedním z hlavních nedostatků, z pohledu autora je to, že uživatelé mohou vstoupit do režimu hry až od třetího ročníku. Nabízí se zde možnost vytvoření nějakého jednoduššího režimu pro první a druhý ročník, tak aby počítaní matematických příkladů bylo dostatečně motivující pro všechny ročníky. Dále by bylo vhodné přidat do hry zvuky a animace. Zvuky by mohly být zařazeny například do soubojů nebo i samotný text by mohl být vyprávěn za pomocí audia. Animace by bylo možné přidat například také do průběhu souboje, nebo při vybavování předmětů, kde by bylo možné jednotlivé položky přetáhnout přímo na svoji postavu.

6 Závěr

Cílem bakalářské práce bylo vytvoření naučné webové aplikace pro 1.stupeň základních škol. Pro splnění tohoto cíle bylo nezbytné seznámit se s moderními technologiemi a frameworky. To vyžadovalo provést rešerši naučných webových portálů, na základě, které bylo možné navrhnout a implementovat vlastní webovou aplikaci. Myšlenkou bylo vytvořit aplikaci, která bude jednak naučná, ale díky svému zasazení do hry, včetně příběhu, bude i žáky motivovat k učení.

Aplikace umožňuje přístup pro dva typy uživatelů. V první řadě se uživatel může zaregistrovat a následně přihlásit jako učitel. Každý učitel v aplikaci tak má možnost přidávat slovní úlohy, ale i odstraňovat a upravovat stávající. Mimo jiné má také možnost každé úloze přiřadit patřičnou odměnu, kterou žák dostane, za její správné řešení. Pokud se do aplikace přihlásí žák, má podstatně více možností. Jeho možnosti jsou rozděleny na matematickou a herní část. V matematické části má možnost počítat cvičení, přičemž každé cvičení se skládá z daného počtu příkladů, které pokrývají tématické okruhy daného ročníku žáka. Za každé úspěšně vyřešené cvičení je pak žák odměněn v podobě stříbrných mincí. Kromě cvičení, má také možnost řešit slovní úlohy, které zadává učitel. I v tomto případě je odměněn, pokud danou úlohu správně vyřeší. V poslední řadě této matematické části je závěrečný test. K testu může však žák přistoupit, až pokud splní daný počet cvičení s požadovanou úspěšností. Pokud tento test splní, nejen, že získá odměnu v podobě mincí, ale také postoupí do dalšího ročníku (pokud však není v posledním, tedy v pátém ročníku). Každý žák od třetího ročníků má také možnost vstoupit do herní části, kde využije své nasbírané mince. Tyto mince může uplatnit za nákup z pestré nabídky položek, kterými pak vybaví svoji pohádkovou postavu. Dále si za tyto mince může vylepšovat vlastnosti své postavy v sekci *cvičiště*. Způsob hry žáka vede k tomu, aby svoji postavu vylepšoval, neboť je zde realizován příběh, ve kterém se musí posouvat dál, tím že bude porážet nepřátele, kteří jsou stále silnější. Příběh je rozdělen do tří částí, podle ročníků žáka. V poslední části, se žák utká s hlavním nepřítelem a při jeho porážce tento příběh končí.

Výsledkem bakalářské práce je interaktivní naučná webová aplikace, která svým odměnovým systémem a formou hry, žáky motivuje k rozvoji svých dovedností. Aplikace je připravená pro zahrnutí do výuky na základních školách, kde se mohou zapojit i učitelé, kteří zde mají možnost vymýšlet a přidávat slovní úlohy, na míru svým žákům. Aplikaci je možné v budoucnu dále rozšiřovat a přidávat nové funkce, herní položky nebo nové nepřátele. Nabízí se zde, také rozšíření okruhů matematiky, včetně herní části pro druhý stupeň. Další možností je také zahrnutí dalších předmětu základní školy.

Literatura

[1] KOĎOUSKOVÁ Barbora. Web, webová stránka a webová aplikace, v čem je rozdíl? WEB & MOBILE DEVELOPMENT AGENCY | Rascasone [online]. Copyright © [cit. 14.02.2023]. Dostupné z: https://www.rascasone.com/cs/blog/web-webova-aplikace-rozdil

[2] Codecadamy: What Is a Framework?. [online]. [cit. 14.02.2023]. Dostupné z: https://www.codecademy.com/resources/blog/what-is-a-framework/

[3] THAPLIYAL Vimal. Difference Between Frontend and Backend MVC Joomlatuts [online]. [cit. 14.02.2023]. Dostupné z: http://joomlatuts.net/joomla-2-5/87-how-backend-model-view-controller-mvc-works-in-joomla/98-difference-between-frontend-and-backend-mvc

[4] Vuejs. The Progressive JavaScript Framework.[Online]. [cit. 14.02.2023] Dostupné z: https://vuejs.org.

[5] MÁCA, Jindřich. Úvod do NodeJS. Node.js [online]. [cit. 14.02.2023]. Dostupné z: https://www.itnetwork.cz/javascript/nodejs/uvod-do-nodejs.

[6] SEDLÁČEK, Petr. Úvod do MongoDB. Node.js [online]. [cit. 14.02.2023]. Dostupné z: https://www.itnetwork.cz/javascript/nodejs/uvod-do-mongodb?gclid=Cj0KCQjw9ZGYBhCEARIsAEUXITX6YwhSlVcltcV7-rMeBQn_jJdlv13VBnCNsLtL-Xf2YyRHMGaNE_0aAtMWEALw_wcB

[7] Postman. Kutáč: Weby a vše okolo [online]. [cit. 14.02.2023]. Dostupné z: https://www.kutac.cz/weby-a-vse-okolo/postman-1-cast

[8] Frontend vs. Backend vs. Full-Stack – The Difference Explained | ElevateX [online]. [cit.24.03.2023] Dostupné z: https://elevatex.de/blog/it-insights/frontend-vs-backend-vs-fullstack-differences

[9] Selzer, Michelle. Salt and Hash Passwords with bcrypt [online]. [cit. 24.03.2023]. Dostupné z: https://heynode.com/blog/2020-04/salt-and-hash-passwords-bcrypt

[10] RUSEV Konstantin. What is Heroku and What is it Used For? [online]. [cit. 24.03.2023]. Dostupné z: https://mentormate.com/blog/what-is-heroku-used-for-cloud-development

7 Seznam příloh

7.1 Odkaz na aplikaci

http://mate-magic.herokuapp.com

7.2 Odkaz na GitHub

https://github.com/LukasDufek/webEducationApp.git

7.3 Příběh

Příběh se odehrává v království, kde dlouhá léta panoval klid a mír. Jednoho dne se však objevil černokněžník, který unesl královu dceru na svém tříhlavém drakovi. Říše se tak ocitla v nebezpečí, protože si všichni uvědomili, že král nemá žádného dědice. Této situace využili loupežníci a nájezdníci usilující o králův trůn. V říši tak začali vládnout nepokoje.

Rázem se ocitáme na farmě, která je zpustošena loupežníky a mladý sedlák Jakub, za kterého žák hraje, přísahá otci, že se loupežníkům pomstí. Vydává se tak najít jejich útočiště a během své cesty se mu podaří i osvobodit stařík, který ho vyučuje v boji. Jakubovi se podaří porazit všechny loupežníky z tábora, kde najde i vzácné klenoty, které patří králi a rozhodne se tyto klenoty vrátit.

Další část příběhu se odehrává již ve čtvrtém ročníku, když Jakub dorazí na královský hrad. Po krátkém rozhovoru s králem, ale na hrad zaútočí nájezdníci a Jakub králi pomáhá s obranou hradu. Potom, co se jim podaří nájezdníky porazit a zajmout jejich velitele, Jakub zjistí, že tento velitel byl ve spojení s černokněžníkem. Král mu pak prozradí celou pravdu o únosu jeho dcery. Jejich rozhovor je však přerušen příletem draka i s jeho temným pánem. Drak po sobě zanechá několik zničených budov a černokněžník králi pohrozí dalším útokem. Jakub králi slíbí, že najde tohoto černokněžníka a pomstí se mu. Jakub se i přes královo varování,

vydává do drakovy jeskyně, kde se mu podaří draka porazit, ale černokněžníka zde nenalezne. Místo toho najde vchod do temné říše, ve které má černokněžník svůj hrad. Děj pátého ročníku se tak odehrává právě v této říši, když se Jakub vydává napříč touto temnou a nebezpečnou říší porazit samotného černokněžníka a zachránit princeznu.