VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO ONLINE VÝUKU MATEMATIKY

SYSTEM FOR ONLINE TEACHING OF MATHEMATICS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **LUKÁŠ DOHNAL**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. RADEK BURGET, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Dohnal Lukáš**
Program: Informační technologie
Název: **Systém pro online výuku matematiky**
System for Online Teaching of Mathematics
Kategorie: Web

Zadání:

1. Seznamte se s požadavky na online výuku matematiky prostřednictvím webového systému. Prozkoumejte existující řešení v této oblasti.
2. Prostudujte současné technologie pro tvorbu webových aplikací. Zaměřte se na zobrazení matematických vzorců, vektorové grafiky a interaktivních prvků na straně klienta.
3. Navrhněte architekturu systému pro online výuku matematiky prostřednictvím interaktivních úloh.
4. Po dohodě s vedoucím implementujte navržený systém na vhodné softwarové platformě.
5. Vytvořte v systému sadu vhodných úloh a otestujte funkčnost řešení.
6. Zhodnoťte dosažené výsledky.

Literatura:

- Gutmans, A., Rethans, D., Bakken, S.: Mistrovství v PHP 5, Computer Press, 2012
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Burget Radek, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 22. října 2019

Abstrakt

Cílem této bakalářské práce je navrhnout a implementovat webovou aplikaci pro online výuku matematiky. Systém je primárně určen pro studenty a učitele na středních školách. Aplikace je implementována v jazycích PHP a Javascript. Pro backendovou část je zvolen framework Symfony a pro frontendovou část byl použit framework React.

Abstract

The aim of this bachelor thesis is to design and implement a web application for online teaching of mathematics. The system is primarily intended for students and teachers at secondary schools. The application is implemented in PHP and Javascript languages. The Symfony framework is chosen on the backend and the React framework was used for the frontend implementation.

Klíčová slova

webová aplikace, výuka matematiky, PHP, Symfony, MySQL, Javascript, React

Keywords

web application, math education, PHP, Symfony, MySQL, Javascript, React

Citace

DOHNAL, Lukáš. *Systém pro online výuku matematiky.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Burget, Ph.D.

Systém pro online výuku matematiky

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Radka Burgeta, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Lukáš Dohnal
26. května 2020

Poděkování

Rád bych poděkoval panu Ing. Radku Burgetovi, Ph.D. za vedení mé bakalářské práce, čas strávený konzultacemi a poskytnutím zpětné vazby k práci.

Obsah

Kapitola 1

Úvod

Cílem této bakalářské práce je vytvořit aplikaci pro výuku matematiky. Aplikace je primárně určena pro pomocnou výuku matematiky na středních školách, ale lze ji využít i pro výuku na základních a vysokých školách.

Práce je implementována jako webová aplikace z důvodu možnosti využití z jakéhokoliv zařízení a možnosti jednoduché rozšiřitelnosti funkcionality.

Cílem návrhu aplikace bylo, aby učitelé na školách mohli jednoduše zadávat úkoly z oblasti matematiky svým studentům, kteří pak za jejich úspěšné splnění dostávají odměnu ve formě bodů. Pro zadávání příkladů z oblasti matematiky, je zde možnost využívat jazyku LaTeX pro jednoduché zapsání složitějších příkladů. Pro vyhodnocení výsledků studentů aplikace poskytuje statistické přehledy.

Obsah práce je rozdělen do několika kapitol a podkapitol. Kapitola 2 popisuje základní principy webových aplikací a představuje základní programovací jazyky, které se často využívají při tvorbě webových aplikací.

V kapitole 3 popisuji již existující řešení na podobné téma. Na konci kapitoly je podkapitola věnující se zhodnocení již existujících řešení a následně z tohoto hodnocení vytváří specifikace pro novou aplikaci.

Kapitola 4 se zaměřuje na samotný návrh výsledné aplikace. Je zde uveden diagram případů užití, návrh databázové struktury společně s vysvětlením jednotlivých entit a pro návrh uživatelského rozhraní je uveden drátěný model.

V kapitole 5 popisuji technologie, které byly pro vývoj webové aplikace vybrány a následně použity.

Kapitola 6 se věnuje již samotné implementaci webového systému. Kapitola obsahuje několik podkapitol věnujících se popisu implementace nejdůležitějších částí aplikace.

Následující kapitola 7 se zabývá testováním výsledné aplikace. Jsou zde uvedeny výsledky jednotlivých testů a následné kroky pro vylepšení aplikace na základě výsledků těchto testů.

Kapitola 2

Principy a možnosti tvorby webových aplikací

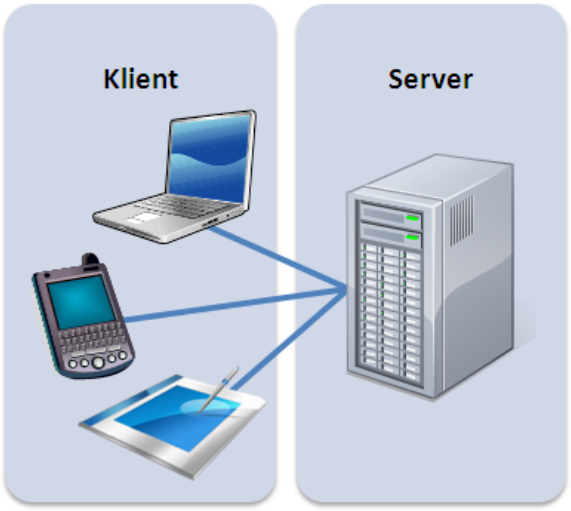
V této kapitole se zaměříme na to, co to vlastně je webová aplikace a jak vypadá její architektura. Následně se zaměříme na možnosti využití technologií, které lze využít k vytvoření webových aplikací.

2.1 Webová aplikace

Webové aplikace jsou aplikace, které jsou uživatelům poskytovány přes internet. Lze k nim přistoupit odkudkoliv, kde je dostupný internet, což zároveň patří mezi jejich největší výhodu. Na druhou stranu, mezi jejich největší nevýhodu patří to, že nemáme skutečnou kontrolu nad daty, mnohokrát ani nevíme, kde se ve skutečnosti data nacházejí a zdali jsou dostatečně zabezpečena před napadením a potencionálním únikem.

2.1.1 Architektura klient-server

Architektura klient-server je dvouvrstvý typ architektury, který využívá dva druhy výpočetních systémů - klienta a serveru. Zatímco klient obsahuje uživatelské rozhraní a logiku aplikace, server se specializuje na databázové dotazy. Mezi klientem a serverem se tedy přenášejí pouze dotazy a výsledky. Schéma architektury klient-server je zobrazeno na obrázku 2.1.

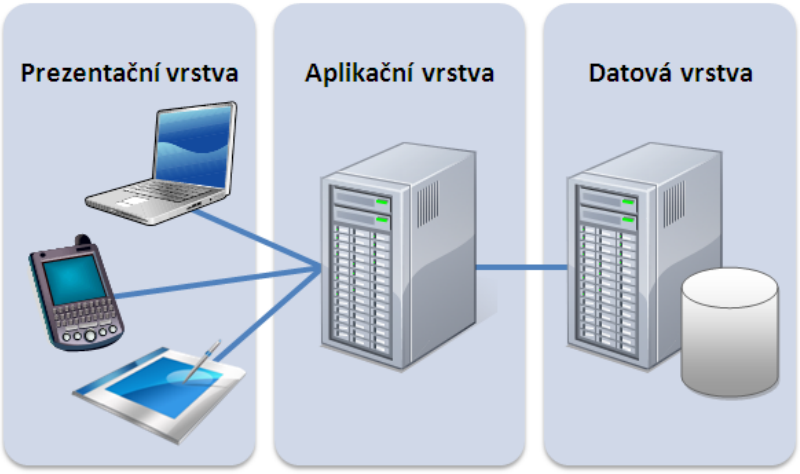


Obrázek 2.1: Architektura klient-server (zdroj: [11])

2.1.2 Třívrstvá architektura

V dnešní době jsou webové aplikace nejčastěji strukturované jako třívrstvé. První vrstva architektury - prezentační - má nejblíže k uživateli, zobrazuje mu informace nejčastěji formou grafického uživatelského rozhraní a provádí validace zadávaných vstupů. Vrstva aplikační, která tvoří mezivrstvu mezi vrstvou prezentační a datovou, obsahuje samotnou logiku aplikace a zpracovávání dat. Tato vrstva je nejčastěji tvořena nástroji pro dynamické generování stránek, např. PHP. Poslední vrstvu - datovou - tvoří databáze.

Samotná komunikace probíhá tak, že prohlížeč (prezentační vrstva) komunikuje pomocí protokolu HTTP s aplikační vrstvou, která zpracuje data a provádí dotazy nad datovou vrstvou (databází). Následně získaná a zpracovaná data předává zpět prezentační vrstvě. Schéma třívrstvé architektury je zobrazeno na obrázku 2.2.



Obrázek 2.2: Třívrstvá architektura (zdroj: [10])

2.2 Možnosti využití technologií

Při vývoji webových aplikací využíváme velké množství technologií, které se stále vyvíjejí. Mnoho z těchto technologií se vzájemně doplňuje. Tyto technologie můžeme zjednodušeně rozdělit na serverové a klientské.

Klientské technologie fungují na principu spouštění kódu přímo v prohlížeči uživatele. Tento princip má výhodu v tom, že pro některé operace nemusíme zatěžovat servery požadavky, protože se využívá výkon klientského počítače. Serverové technologie se využívají naopak na straně serveru. Ve chvíli, kdy server obdrží požadavek od klienta, zpracuje jej a zpět již odesílá pouze výsledná data.

2.2.1 HTML

HTML neboli HyperText Markup Language je značkovací jazyk používaný pro tvorbu webových stránek. Tento jazyk vyvinul v roce 1990 Tim Berners-Lee společně s jeho kolegy v CERNu - evropském výzkumném centru částicové fyziky. Tento nápad vycházel z toho, že se libovolný text může odkazovat na jiný libovolný text. [13] Nejnovější verze jazyka HTML je verze 5.3, která byla publikována 18. října 2018 organizací W3C. [20]

HTML jazyk popisuje strukturu webu pomocí značek, které nazýváme *tagy*. Mezi tyto značky se poté vkládají části textu dokumentu a tím se určí sémantika obsaženého textu. Názvy jednotlivých značek se vkládají mezi značky < a >. Značky mohou obsahovat také vlastnosti, které upřesňují význam značek, jako například identifikátory pro oddíly dokumentu nebo odkazy na zdroj obrázku. Značky mohou být párové a nebo nepárové, dle toho, jestli mají obsahovat část textu. Mezi párové značky patří například značky pro odstavce `p` a nadpisy `h1`, naopak mezi nepárové patří třeba značka pro obrázek `img`.

HTML jazyk prodělal velkou změnu v roce 2014, kdy byl vydán standard HTML5. Ten přinesl velké množství změn, mezi které patří mnohem lepší podpora přehrávání médií v prohlížeči či zavedení nových značek pro přesnější určení sémantiky dokumentu. Abychom mohli využívat nový standard HTML5, musíme na začátku dokumentu uvést značku `<!DOCTYPE html>`.

2.2.2 CSS

CSS (*Cascading Style Sheets*) slouží pro definici vzhledu webové stránky. První návrh na vytvoření kaskádových stylů, jehož autorem byl Håkon Wium Lie, vznikl v roce 1994 ve standardizační organizací W3C. [2] Hlavní důvod vzniku CSS bylo oddělení definice vzhledu stránky od jejího obsahu.

Hlavní problém při vytváření CSS je jejich podpora v jednotlivých prohlížečích. Každý prohlížeč může jednotlivá pravidla CSS interpretovat trochu jinak nebo dokonce vůbec. Tento problém přetrvává bohužel dodnes, i když již ne v tak velké míře jako dříve.

Důvodů pro využívání kaskádových stylů, kromě oddělení definice vzhledu stránky od jejího obsahu, je mnoho. Mezi hlavní výhody patří: [6]

- Možnost nastavení stylů dokumentu pro různá média - různé rozlišení obrazovky, tisk.
- Kaskádové styly umožňují používat jeden seznam pravidel pro více dokumentů. Pokud chceme udělat změnu, stačí změnit dané pravidlo v seznamu a změna se promítne do všech dokumentů, ke kterým je soubor s pravidly připojen.

- CSS soubory si webový prohlížeč ukládá do paměti cache. To znamená, že načítání stránek je rychlejší.

CSS je v dnešní době nejčastěji definováno jako seznam pravidel v externím souboru, který je následně propojen s dokumentem HTML. Samotné pravidlo je tvořeno ze selektoru a bloku deklarací. Selektor určuje, na které elementy HTML má být styl aplikován. V bloku deklarací jsou pak následně uvedena samotná pravidla pro nastavení vzhledu daného elementu. Nastavení vzhledu elementů pomocí CSS skrývá mnoho možností. Mezi základní vlastnosti vzhledu patří vlastnosti jako `color` pro nastavení barvy textu, `background-color` pro nastavení barvy pozadí elementu či `font-size` pro nastavení velikosti písma fontu.

2.2.3 Javascript

Javascript je programovací jazyk, jehož autorem je Brendan Eich ze společnosti Netscape. Zpočátku byl tento jazyk nazýván *LiveScript*, ale jakmile se ve světe začal ve velkém používat jazyk *Java*, společnost Netscape se z marketingového hlediska rozhodla přejmenovat jazyk na *Javascript* (přestože tyto dva jazyky nemají téměř nic společného).[21]

Kvůli zachování bezpečnosti autoři programovacího jazyka Javascript do něj nezabudovali funkce, které by umožňovaly pracovat se soubory. Pro uživatele je toto ovšem pozitivum, protože se nemusí obávat toho, že by někdo pomocí tohoto jazyka mohl získat přístup k informacím na jeho disku.

Dříve byl jazyk využíván hlavně pro psaní jednoduchých skriptů, jako například pro různé animace prvků a kontroly formulářů před jejich odesláním, dnes se využívá i pro tvorbu celých aplikací.

Pro jednodušší práci s Javascriptem existuje několik knihoven. Mezi nejznámější patří knihovna *jQuery*, která zjednodušuje práci s elementy. Naopak na serverové straně je nejznámější projekt *Node.js*.

2.2.4 PHP

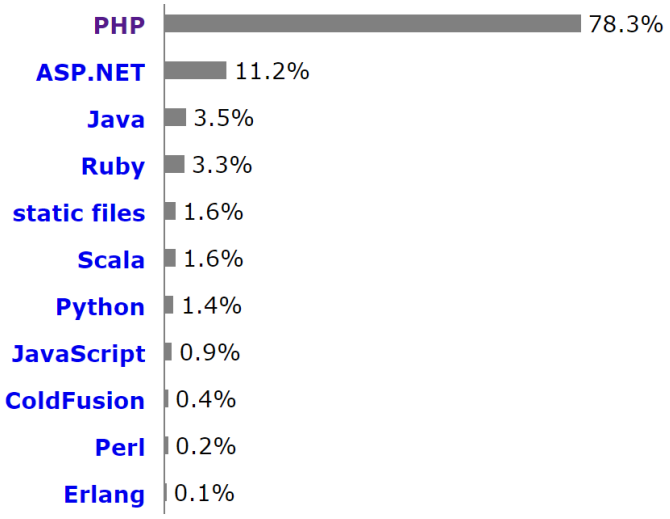
PHP je skriptovací programovací jazyk určený pro tvorbu dynamických webových stránek. První verzi PHP vytvořil Rasmus Lerdorf v roce 1994. [5]

Vytvořené skripty v jazyce PHP jsou vykonávány na straně serveru. To znamená, že uživateli se již odesílá pouze výsledný vygenerovaný obsah stránky. Jazyk PHP je při tvorbě webových stránek velmi populární. Využívají jej i velmi známé webové stránky, mezi které patří Facebook, Wikipedia či Flickr. Jak můžeme vidět na obrázku 2.3, který ukazuje aktuální využívání jednotlivých technologií na straně serveru, PHP jim výrazně dominuje.

Popularitě PHP také přispívá to, že jej podporuje velké množství webových hostingů. Poslední verze PHP je 7.4, která byla publikována 28. listopadu 2019. Hlavní novinkou, kterou tato verze přinesla, je možnost typování atributů. [7] Velká změna ovšem přišla ve verzi 7.0, která vyšla 11.6.2015 a přinesla nový engine pod označením PHP#NG, který zajistil dvojnásobnou rychlost vykonávání příkazů oproti starému engine Zend Engine II. [16]

2.2.5 Java

První náznaky jazyku Java začali v roce 1991, tehdy jazyk nazývali *Oak*. První verze vydaná jako veřejná - Java 1.0, spatřila světlo světa v roce 1995. Specifikace Javy pro web - Java Server Pages 1.0, vyšla v roce 1999 jako odpověď na ASP.NET a PHP. [9]



Obrázek 2.3: Porovnání používaných technologií na straně serveru(zdroj: [18])

Edice Javy, která se soustředí na vytváření webových aplikací, se nazývá *Java Enterprise Edition*. Název je poněkud zavádějící, protože tato verze Javy není nijak vázaná na komerční sféru, ale můžeme v ní naprogramovat třeba i osobní blog. JEE není úplně jiná verze samotné Javy, pouze přidává sadu knihoven do Javy SE (standardní edice). Přestože v Javě EE lze vytvořit i malé weby, převážně se používá ve velkých firmách k tvorbě náročných aplikací, jako jsou státní registry nebo bankovní aplikace, protože je to velmi robustní řešení, které již vyžaduje od programátora mít určité zkušenosti.

2.2.6 ASP.NET

ASP.NET vznikl v roce 2002 jako součást .NET frameworku. Stal se nástupcem Active Server Pages (ASP) technologie, od které se ale hodně liší. [4] ASP.NET díky tomu, že je součástí .NET frameworku, umožňuje využívat technologie CLR (Common Language Runtime). To umožňuje psát projekty v jakémkoliv jazyce podporujícím CLR, jako je C#, Visual Basic .NET nebo J#. [15]

2.2.7 Python

Python je multiplatformní jazyk, který má široké využití. Lze v něm tvořit jak drobné skripty, tak desktopové nebo webové aplikace. Zároveň se doporučuje začínajícím programátorům, protože je dobře čitelný a má jednoduchou syntaxi.

V Pythonu lze programovat jak v procedurálním, tak v objektovém či v menší míře ve funkcionálním stylu, přestože v jádru je Python objektově orientovaným jazykem. Python je zároveň velmi expresivní jazyk, což znamená, že obvykle stačí napsat mnohem méně řádků kódu v Pythonu, než kolik by bylo zapotřebí napsat pro vytvoření ekvivalentní aplikace v některém jiném jazyku, jako třeba C++ nebo Java. [19] Pro programování webových aplikací se používá v Pythonu nejčastěji framework Django či Flask.

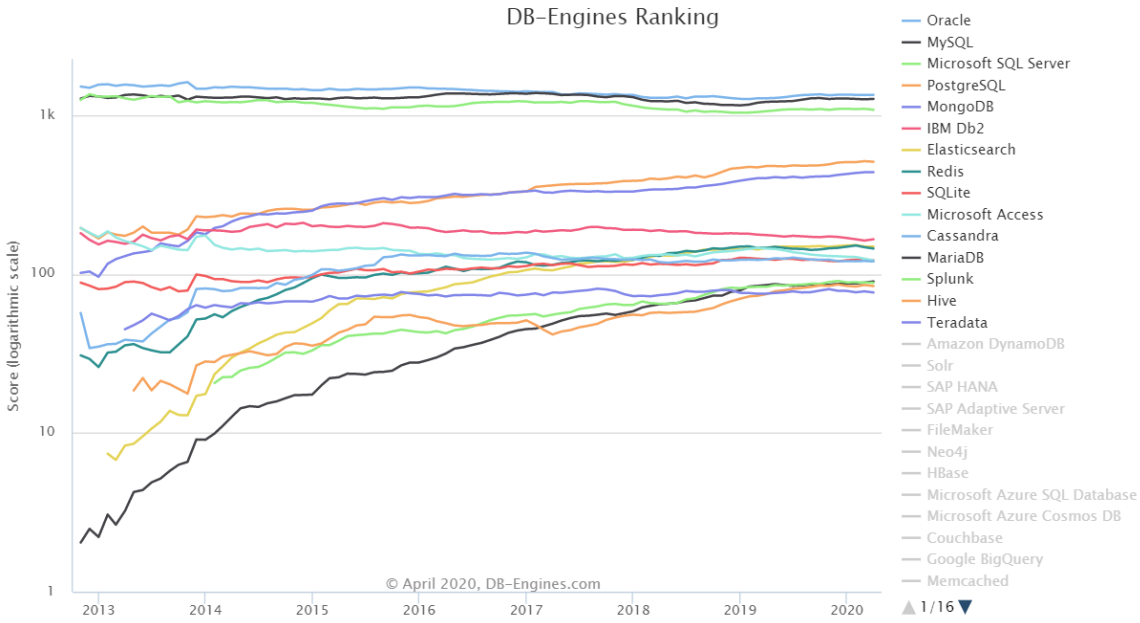
2.2.8 MySQL

MySQL (*My Structured Query Language*) je multiplatformní relační databázový systém, který je vyvíjen švédskou firmou MySQL AB. Relační databáze je databázový systém, který je založen na relačním modelu dat a relační algebře. Data v relační databázi jsou uložena ve více tabulkách a tyto tabulky jsou navzájem propojené. Pro manipulaci s daty, uloženými v relační databázi, se používá jazyk SQL (Structured Query Language). [12]

MySQL díky multiplatformnosti, výkonu a nulové ceně, je široce využíván na velkém množství webhostingů. Jak můžete vidět v grafu na obrázku 2.4, MySQL společně s Oracle a Microsoft SQL Server, které se využívají spíše v komerční sféře, je jednou z nejpoužívanějších databází.

MySQL bylo od začátku vývoje optimalizováno především pro vysoký výkon, na druhou stranu daní za to byla chybějící podpora některých funkcionalit. Ty ale byly kvůli rostoucím nárokům tvůrců webových aplikací postupem času doplňovány. Mezi funkcionality, které byly časem přidávány, patří například podpora cizích klíčů či podpora uložených procedur, triggerů a pohledů. [1]

MySQL podporuje více typů tabulek. Mezi ty nejznámější patří InnoDB a MyISAM. Hlavní výhoda InnoDB je v tom, že podporuje transakce a cizí klíče. MyISAM naopak podporuje fulltextové indexy a je optimalizováno především na dotazy typu `SELECT`.



Obrázek 2.4: Graf popularity jednotlivých databází(zdroj: [3])

Kapitola 3

Analýza existujících řešení

V této kapitole provedu analýzu již existujících řešení pro on-line výuku matematiky. Následně zhodnotím nedostatky jednotlivých řešení a z nich vyplývající návrh řešení pro nový systém, který budu implementovat. U každého existujícího řešení je uveden souhrn vlastností v bodech.

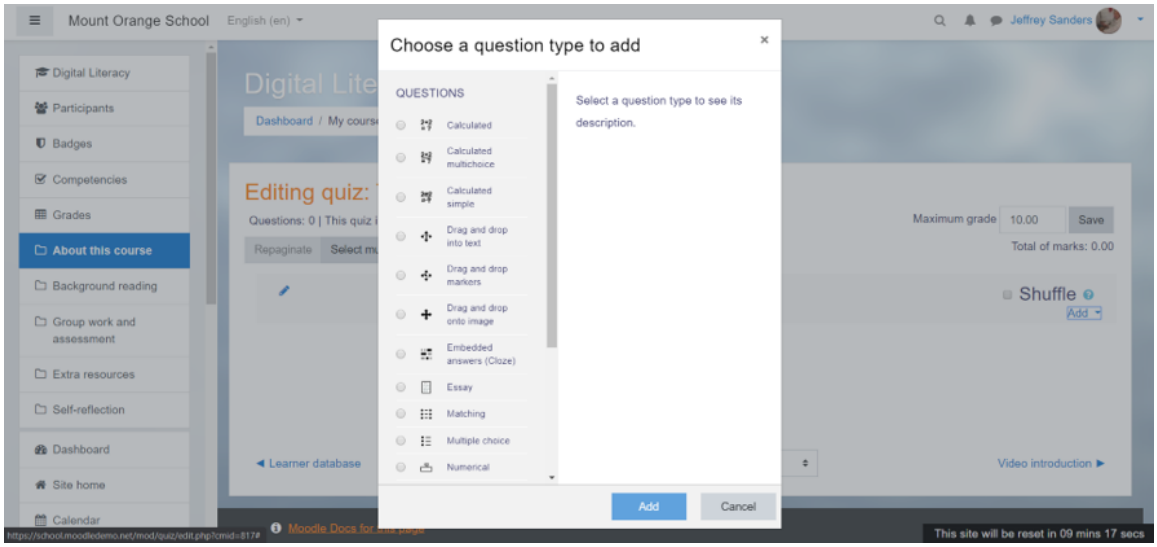
3.1 Moodle

Moodle je komplexní systém pro tvorbu výukových systémů a elektronických kurzů na internetu. Je poskytován jako open-source pod licencí GNU. Autorem projektu Moodle je Martin Dougiamas a jeho první prototypy byly napsány v Pythonu. Poté byl projekt přepsán do jazyku PHP a jeho první verze byla vydána v roce 2002. [17] Hned o rok později již Moodle začal podporovat českou lokalizaci.

Moodle má velmi rozsáhlé možnosti přizpůsobení. Obsahuje velké množství modulů, kterými si uživatel systém přizpůsobí. Uživatel si může přidat moduly například pro přidání diskuzního fóra do kurzu, přidání ankety či automaticky vyhodnocovaného testu. I nastavení samotných testů je velmi rozsáhlé. Na druhou stranu, je až tak rozsáhlé, že pro některé uživatele se tímto stává nepřehledným a některé nastavení uživatelé ani nevědí, co znamená. Moodle také umožňuje používat metodu přihlašování LDAP, kde se přihlašovací údaje uživatele kontrolují proti serveru LDAP.

Projekt Moodle je využíván na velkém množství škol v České republice, a to ať na základních, tak středních i vysokých školách. Pro přehled ho využívají školy jako VUT, Univerzita Karlova, Masarykova Univerzita či Univerzita Tomáše Bati.

- Studenti mají všechny materiály a testy na jednom místě.
- Nastavení kurzů a testů je velmi rozsáhlé.
- Pro zadávání testů matematiky ne příliš vhodné, kvůli možnosti zadání otázky i odpovědi pouze ve formě textu.
- Z důvodu velké komplexnosti systému místy nepřehlednost.



Obrázek 3.1: Ukázka projektu Moodle

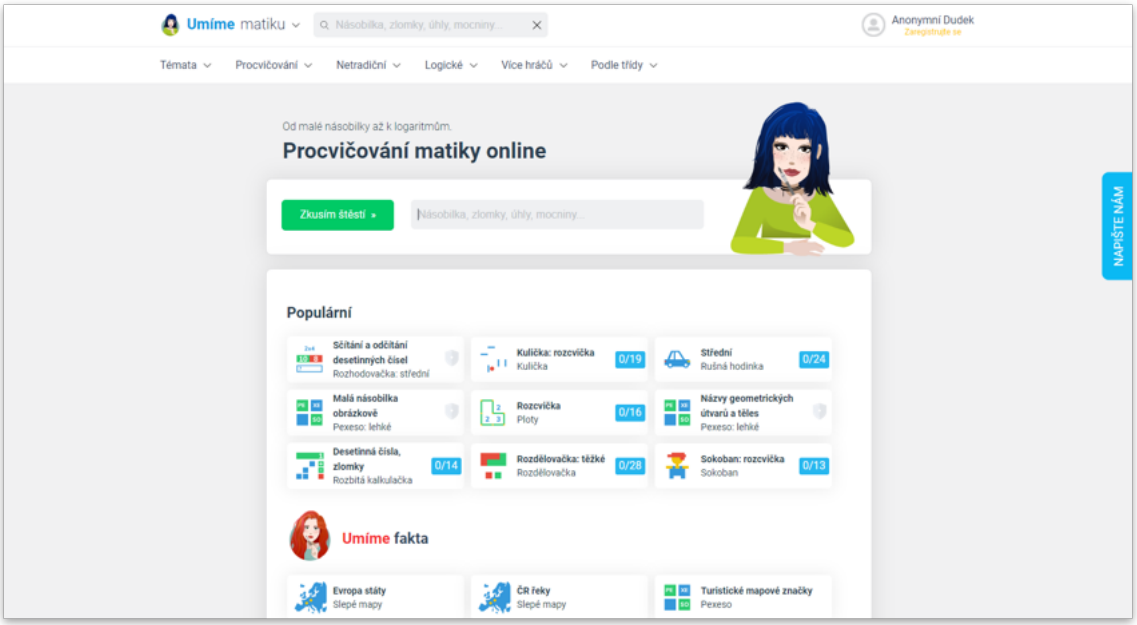
3.2 Umíme matiku.cz

Web umimematiku.cz nabízí několik témat pro procvičování matematiky. Témata jsou procvičovány jak klasickými příklady, tak formou her - ve formě sudoku či pexesa. Témata pro procvičování jsou připravena pro základní a střední školy. Web nabízí registraci pro studenty, jejich rodiče a učitele.

Studentský mód nabízí klasické vybrání látky a její procvičení, rodičovský mód umožňuje dohled nad dodržováním procvičování dětí. Učitelský mód nabízí zadávání domácích úkolů jednotlivým třídám a kontrolu jejich úspěšnosti v jednotlivých látkách. Web je moderní, přehledný, nabízí zajímavé možnosti procvičování látky pomocí různých her. Bohužel, učitelé nemají možnost jakkoliv upravit látku k procvičení ani přidávat novou.

Projekt Umíme matiku.cz vznikl na fakultě informatiky Masarykovy univerzity ve spolupráci s výzkumnou skupinou Adaptive Learning.

- Existují zde různé možnosti procvičení látky - pomocí her, klasické úkoly.
- Rozdělení rolí na studenty, rodiče a učitele - možnost zadávání domácích úkolů učiteli či kontroly plnění úkolů rodiči.
- Moderní, přehledný vzhled webu.
- Nemožnost přidání vlastního obsahu.



Obrázek 3.2: Ukázka projektu Umíme matiku.cz

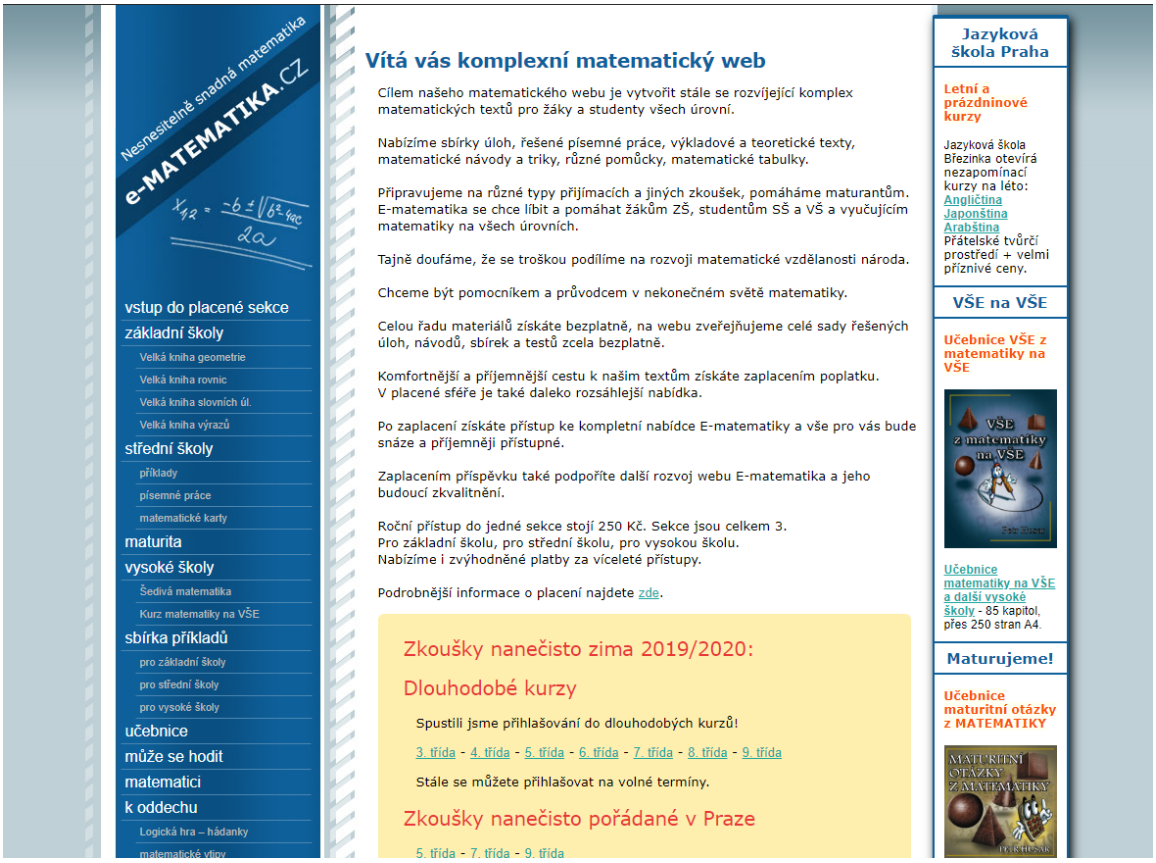
3.3 E-matematika.cz

Web e-matematika.cz je velmi komplexní web pro výuku matematiky. Web vznikl jako jeden z projektů zaměřených na matematiku od Petra Husara a jeho matematického týmu. První projekt tohoto týmu *www.zkousky-nanecisto.cz* vznikl kolem roku 1990 a připravuje studenty na přijímací zkoušky. [8]

Web e-matematika.cz se vydává trochu jiným směrem. Snaží se vytvořit web věnující se matematice ze všech možných úhlů pohledu. Obsahuje materiály jak pro základní, tak pro střední i vysoké školy. Učivo rozděluje na několik kategorií, z nichž každá obsahuje zhruba desítku příkladů s postupem řešení. Web bohužel obsahuje jen ukázkové řešení, ale žádné testy, které by mohli studenti vyplňovat.

Pokud má uživatel zájem o větší obsah, může zaplatit drobný poplatek a získá tím přístup ke kompletní nabídce učiva na webu.

- Web obsahuje velké množství učiva - pro základní, ale i střední a vysoké školy.
- K příkladům jsou přiloženy i postupy řešení.
- Nejsou zde žádné testy k procvičení.
- Poměrně zastaralý a místy nepřehledný vzhled webu.



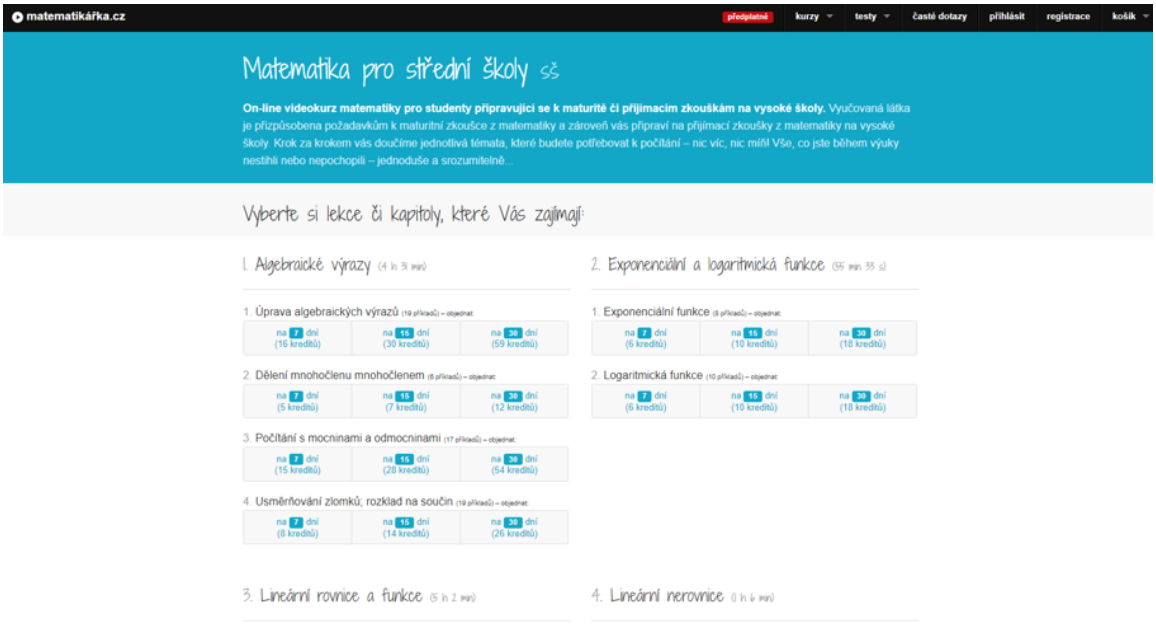
Obrázek 3.3: Ukázka projektu Ematematika.cz

3.4 Matematikářka.cz

Tento web se zaměřuje na výuku matematiky pomocí on-line videokurzů. Videokurzy jsou zaměřeny pouze na matematiku na středních a vysokých školách, a to dle příslušných vysokých škol, nebo pro střední školy obecně. Jednotlivé videokurzy se zakupují pomocí takzvaných kreditů, které se nakupují za peníze.

Ke kurzům jsou přiřazeny také testy obsahující příklady z příslušných látek, avšak tyto testy obsahují jen pár příkladů a k nim příslušné řešení. Pokud by student látku nepochopil, nebo se chtěl dozvědět o daném tématu víc, je možno si s majitelkou webu domluvit on-line doučování.

- Zaměření na určitou VŠ či SŠ - dle toho zaměřená látka.
- Web obsahuje velké množství videokurzů.
- K procvičování jednotlivých látek je malý počet testů.
- Jednotlivé kurzy se zakupují pomocí kreditů.
- Možnost individuálního on-line doučování majitelkou webu.



Obrázek 3.4: Ukázka projektu Matematikářka.cz

3.5 Zhodnocení existujících řešení a návrh specifikací nové aplikace

Do výběru existujících řešení jsem vybral jedny z nejvyužívanějších webových aplikací pro výuku matematiky on-line. Vybrané aplikace jsou určené jak pro správu samotnými školami (Moodle, Umíme matiku.cz), tak pro individuální výuku samotných studentů. Jednotlivé aplikace mají poněkud odlišné zaměření, ovšem několik nevýhod mají společných.

Mezi hlavní nevýhody bych zařadil nepřehlednost jednotlivých webových aplikací (kromě poměrně nově vytvořeného projektu Umíme matiku.cz) a nemožnost přidání vlastního obsahu pro výuku.

Z tohoto zhodnocení jsem následně vytvořil několik klíčových bodů, ke kterým bych chtěl při návrhu a tvorbě aplikace směřovat:

1. Moderní a přehledný vzhled.
2. Aplikace bude spravována jednotlivými školami - učitelé budou mít možnost přizpůsobit si obsah dle jejich představ.
3. Jednoduché rozhraní pro vytváření otázek a jejich odpovídání.
4. Možnost zobrazení přehledných statistik pro vyhodnocení výsledků studentů.

Kapitola 4

Návrh webové aplikace

Aplikace **Matikář** bude sloužit k vytváření a následnému vyplňování úkolů z oblasti matematiky. Otázky budou rozčleněny do kategorií, které budou přiřazeny jednotlivým třídám. Za správně vyplněné úkoly následně studenti dostanou body, pomocí kterých učitel vyhodnotí nejlepší studenty třídy a může je bonusově odměnit.

4.1 Případy užití

Diagramy případu užití jsou součástí modelovacího jazyka UML. Účel diagramu případu užití je popsat funkcionalitu systému. Tedy co má systém umět, ale jak to má systém dělat, to už neříká. I proto tento diagram bývá první, který se při návrhu aplikace vytváří. Diagram případů užití se skládá z případů užití, aktérů a vztahů mezi nimi.

Aktér Aktér komunikuje s jednotlivými případy užití. Aktér může být buď uživatel (uživatelská role), nebo externí systém jako například čas. Aktér se znázorňuje jako postava s názvem napsaným pod ní.

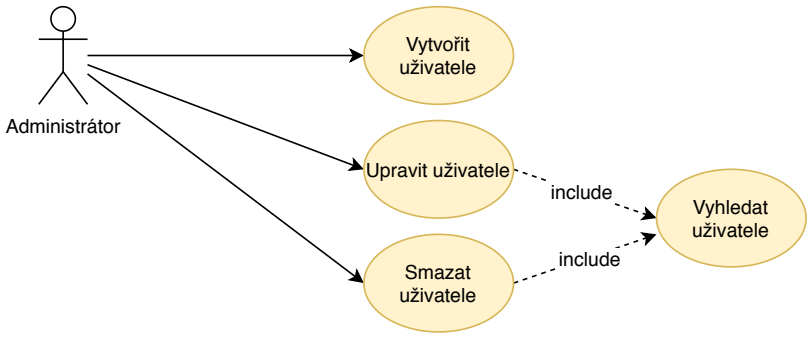
Případ užití Případ užití je akce či sada akcí, které vedou k dosažení daného cíle. Definuje funkcionalitu, kterou by měl navrhovaný systém obsahovat. Případ užití může být například vytvoření uživatele, přidání článku či tisk dokumentu. Akce, které celý proces obsahuje (u vytvoření uživatele validace, přidání do databáze), jsou skryty. Jednotlivé případy užití se nejčastěji vytváří ze zadání zákazníka. Vytváříme tak jednotlivé případy užití dle jednotlivých požadavků zákazníka. Případy užití se zakreslují nejčastěji jako elipsy s názvy akcí uvnitř.

Vazby V diagramu případu užití máme několik typů vazeb. Jednoduchá čára či šipka znamená asociaci - propojení aktérů s jednotlivými případy užití. Šipka s uzavřenou šipkou směrem k jinému aktérovi znamená generalizaci - aktér umí to stejné co jeho předek. Ještě existují dvě speciální vazby, a to `include` a `extend`. Vazba `include` se spustí vždy, když je spuštěn případ užití, na který je napojena. Například pokud chceme editovat uživatele, musíme ho vždy předtím vyhledat. Druhý druh vazby `extend` se používá velmi zřídka. Používá se v případě, že nějaký případ užití rozšiřuje možnosti hlavního případu užití.

Ve webové aplikaci budou tři uživatelské role - administrátor, učitel a student. Veškeré diagramy jsou vytvořeny v on-line webové aplikaci `draw.io`.

4.1.1 Administrátor

Administrátor má nejvyšší oprávnění v celé webové aplikaci. Má všechna oprávnění jako učitel, a navíc jako jediný může vytvářet, editovat či mazat uživatele. Při editaci uživatele může editovat údaje jako je jméno či příjmení, dále mu přiřadit třídy do kterých má přístup a měnit jeho roli. Administrátorů může být v systému více. Administrátor tím, že má navíc všechna oprávnění jako učitel, může vytvářet otázky, třídy a kategorie. Další rozdíl mezi administrátorem a učitelem je ten, že administrátor má přístup ke statistikám všech tříd.



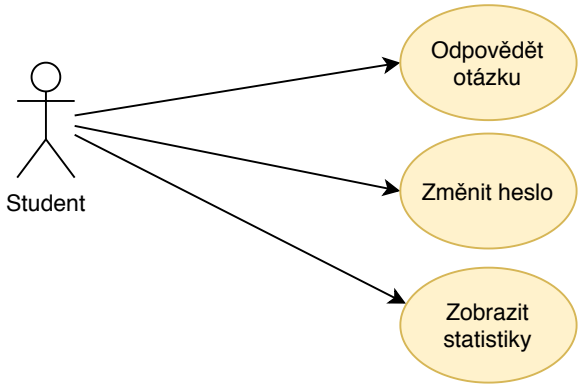
Obrázek 4.1: Diagram případu užití administrátora

4.1.2 Učitel

Učitelská role má přístup ke statistikám studentů v těch třídách, ke kterým má nastaven přístup. Dále může pracovat s otázkami i kategoriemi, nastavovat pořadí otázek v kategoriích a nastavit kategorie pro jednotlivé třídy. Diagram případů užití uživatele s rolí učitel je zobrazen na obrázku 4.3.

4.1.3 Student

Student si bude moct zobrazit statistiky v jeho třídách a kategoriích a odpovídat na samotné otázky. Student také jako jediný si může v aplikaci sám vytvořit účet pomocí registračního formuláře. Tam si zvolí svou primární třídu a vyplní uživatelské údaje. Diagram případu užití studenta můžete vidět na obrázku 4.2.



Obrázek 4.2: Diagram případu užití studenta

Obrázek 4.3: Diagram případu užití učitele

4.2 Návrh struktury databáze

Návrh struktury databáze se nejčastěji vytváří pomocí ER diagramu. ER diagram modeluje data, které budeme v systému uchovávat a vztahy mezi těmito daty. Před tvorbou ER diagramu je nutno definovat si několik pojmů s ním souvisejících.

Entita Entita je objekt, o kterém má být uchovávána informace. Každá taková entita musí být rozlišitelná od ostatních entit a musí být identifikovatelná - mít množinu atributů, které entitu jednoznačně identifikují. Za entitu můžeme považovat například zákazníka obchodu nebo zboží.

Množina entity Množinou entit rozumíme množinu, která je tvořena entitami stejného typu neboli entitami, které mají stejné atributy. Jako příklad můžeme uvést množinu zákazníků obchodu či množinu zboží.

Atribut Atribut je vlastnost entity. Tedy entita je reprezentována množinou atributů, které popisují její vlastnosti. Máme několik typů atributů:

- Jednoduché a složené atributy - jednoduchý atribut je reprezentován primitivním datovým typem, jako je řetězec či číslo, zatímco složený atribut je složen z několika jednoduchých atributů.
- Jednohodnotové a vícehodnotové atributy - vícehodnotový atribut může mít více hodnot současně - například *telefonni_cisla*.
- Odvozené atributy - dají se vypočítat z hodnot jiných atributů. Například věk se dá vypočítat, pokud je známo datum narození.
- Prázdné atributy - mohou nabývat hodnoty `NULL`. Taková hodnota zastupuje neznámou či prozatím chybějící hodnotu.

Vztah Vztah definuje propojení dvou či více entit.

Množina vztahů Množina vztahů je množina propojení mezi dvěma či více množinami entit.

Stupeň vztahu Stupeň vztahu označuje počet entit, které do vztahu vstupují. Nejčastěji se používají stupně unární (vztah mezi dvěma entitami stejné entitní množiny), binární a ternární.

Kardinalita vztahů Kardinalita určuje počet prvků množiny entit, přidružené prostřednictví množiny vztahů. Příklady k jednotlivým kardinalitám vztahu:

- 1:1 - Tento vztah se používá velmi zřídka, protože většinou se místo použití této kardinality data umístí do jedné společné tabulky. Příklad této kardinality vztahu může být vztah mezi entitami člověk a rodné číslo.
- 1:N - Firma má více zaměstnanců.
- M:N - Více spisovatelů může napsat více knih.

Databáze se skládá z celkem 9 tabulek, které jsou zobrazeny ve formě ER diagramu na obrázku 4.4. Diagram byl vytvořen v programu *MySQL Workbench*.

4.2.1 Tabulka `user`

Tato tabulka obsahuje všechny registrované uživatele. Tabulka obsahuje celkem 9 sloupců. Primárním klíčem tabulky je sloupec `id`, který je typu *int(11)*. Následují sloupce pro identifikaci uživatelů, a to `email`, `password`, `first_name` a `last_name`. Do těch se zapisuje email uživatele, pomocí kterého se bude přihlašovat, jeho heslo a dále jméno a příjmení uživatele. Všechny tyto sloupce jsou typu *varchar(255)*. Následuje sloupec `role`, který je typu řetězec o délce 100 znaků a označuje roli uživatele.

Poslední tři sloupce - `activation_code`, `token` a `token_expire`, slouží pro aktivaci účtu a obnovení zapomenutého hesla. Po registraci uživatele se do sloupce `activation_code` vygeneruje řetězec, který slouží pro aktivaci uživatele. Pokud tento sloupec neobsahuje hodnotu `NULL`, uživatel se nemůže přihlásit k účtu. Dále sloupce `token` typu *varchar(255)* a `token_expire` typu *datetime* slouží pro obnovení hesla k uživatelskému účtu. Pokud si uživatel zažádá o obnovu uživatelského hesla, do sloupce `token` se vygeneruje řetězec, který se následně při změně zapomenutého hesla ověřuje a do sloupce `token_expire` se zapíše datum a čas vypršení platnosti tokenu.

4.2.2 Tabulka `class`

Tabulka `class` obsahuje jednotlivé studentské třídy. Tato tabulka je velmi jednoduchá a obsahuje pouze dva sloupce. Prvním sloupcem je sloupec `id`, který je primárním klíčem tabulky a je typu *int(11)*. Druhým sloupcem tabulky je sloupec `name`, který je typu *varchar(255)* a označuje samotný název studentské třídy.

4.2.3 Tabulka `user_class_user`

Tabulka `user_class_user` je vytvořena jako pomocná tabulka pro vazbu M:N mezi tabulkami `user` a `class`. Tabulka obsahuje pouze dva sloupce s cizími klíči - sloupce s hodnotami `id` z tabulek `user` a `class`.

4.2.4 Tabulka `category`

Tato tabulka obsahuje jednotlivé kategorie, do kterých se pak zařazují otázky. Tabulka obsahuje pouze dva sloupce, a to primární klíč `id`, který je typu *int(11)* a sloupec `name`, který je typu *varchar(255)* a vyjadřuje název dané kategorie.

4.2.5 Tabulka `category_in_class`

Tabulka `category_in_class` je vytvořena jako pomocná tabulka pro vazbu M:N mezi tabulkami `category` a `class`. Tabulka obsahuje pouze dva sloupce s cizími klíči - sloupce s hodnotami `id` z tabulek `category` a `class`.

4.2.6 Tabulka `question`

Tabulka `question` obsahuje otázky a má 8 sloupců. Primárním klíčem v tabulce je sloupec `id` typu *int(11)*. Tabulka dále obsahuje sloupec `description`, který je typu *varchar(255)* a označuje krátký název vystihující otázku. Dále sloupec `reward` typu *int(11)*, který obsahuje počet bodů za úspěšné splnění otázky a sloupec `type`, který je typu *smallint* a označuje typ otázky.

Pro obsah samotné otázky jsou tu dva sloupce. Sloupec `text`, který je typu *longtext* a obsahuje samotný text otázky a sloupec `file`, který v případě, že je otázka zadána formou obrázku, tak obsahuje název souboru. Předposlední sloupec `answer_type` je typu *smallint* a označuje typ odpovědi. Poslední sloupec tabulky je sloupec `auto_control`, je typu *tinyint*, a označuje, jestli má být odpověď otázky automaticky zkontrolována a vyhodnocena.

4.2.7 Tabulka `question_order_in_category`

Tabulka `question_order_in_category` určuje otázky v jednotlivých kategoriích a jejich pořadí. Tabulka obsahuje primární klíč `id`, který je typu *int(11)*. Dále tabulka obsahuje dva cizí klíče do tabulek `question` a `category`. Poslední sloupec tabulky je sloupec `question_order`, který je typu *int(11)* a určuje pořadí otázky v dané kategorii.

4.2.8 Tabulka `answer`

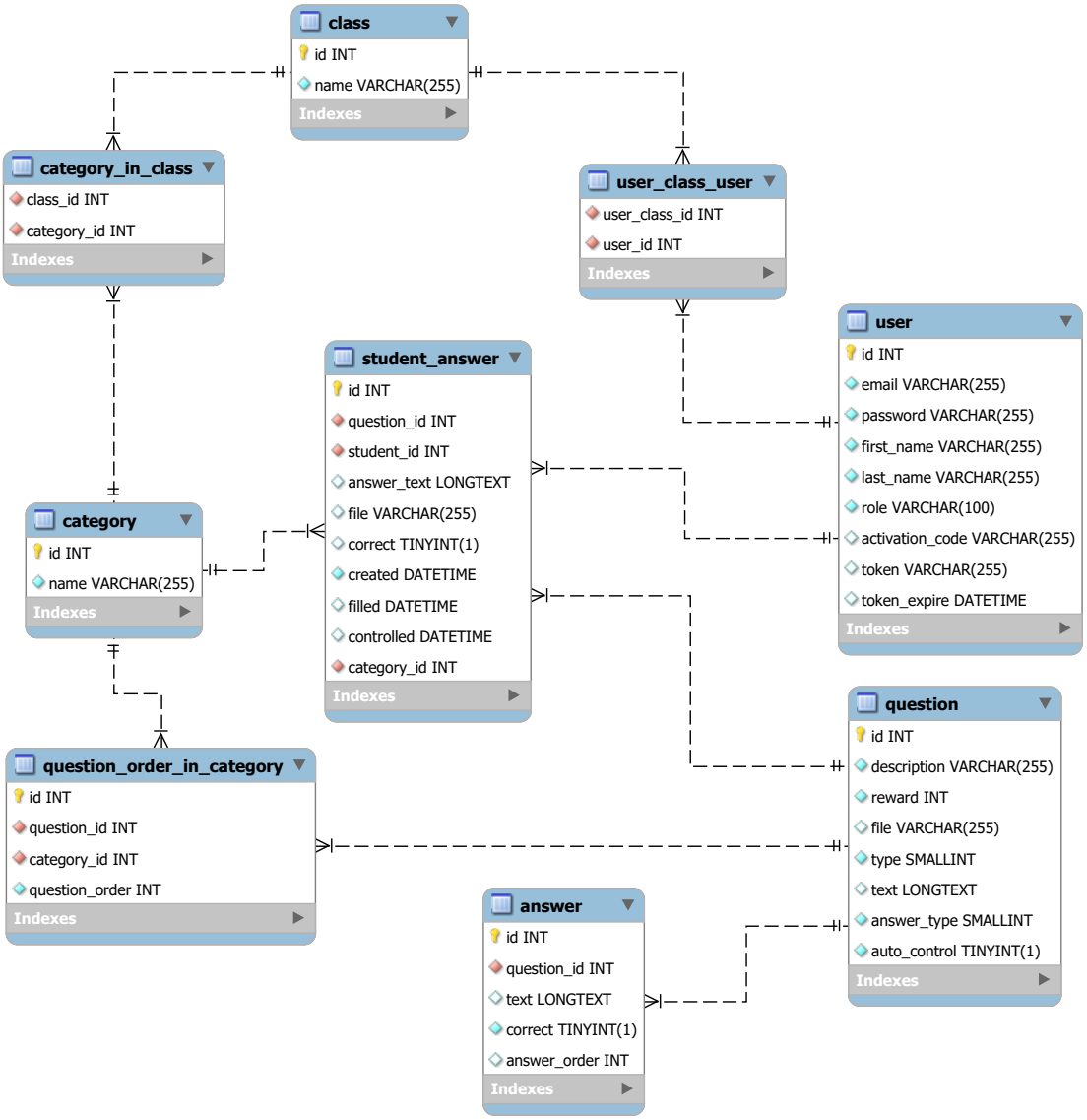
Tato tabulka obsahuje odpovědi k otázkám, které se automaticky vyhodnocují a mají textovou odpověď. Tabulka má celkem 5 sloupců. Primárním klíčem tabulky je sloupec `id`, který je typu *int(11)*. Dále tabulka obsahuje cizí klíč do tabulky `question`, který označuje, ke které otázce daná odpověď patří. Dalším sloupcem je sloupec `text`, který je typu *longtext* a obsahuje samotnou odpověď na otázku, která se porovnává při automatickém vyhodnocení otázky s odpovědí studenta.

4.2.9 Tabulka `student_answer`

Poslední tabulka v databázi má název `student_answer` a obsahuje studentské odpovědi na otázky. Má celkem 10 sloupců. Primárním klíčem tabulky je sloupec `id`, který je typu *int(11)*. Dále tabulka obsahuje tři cizí klíče do tabulek `question`, `user` a `category`. Cizí klíče do tabulek `question` a `category` označují, ke které otázce a kategorii odpověď patří a cizí klíč do tabulky `user` označuje, ke kterému studentovi odpověď patří.

Dále má tabulka dva sloupce určující samotnou odpověď - `answer_text` a `file`. Sloupec `answer_text` je typu *longtext*, obsahuje samotný text odpovědi studenta a využívá se při textovém typu odpovědi. Druhý sloupec `file` je typu *varchar(255)*, označuje název souboru s odpovědí a využívá se při typu odpovědi, kdy student nahrává jako odpověď obrázek. Tabulka dále obsahuje sloupec `correct`, který je typu *tinyint* a obsahuje příznak, jestli je odpověď správná. Tento příznak se vyplní automaticky ihned, pokud má otázka nastaveno automatické vyhodnocení odpovědi. Jinak má hodnotu `NULL` do doby, dokud odpověď nezkontroluje učitel.

Poslední tři sloupce tabulky jsou typu *datetime* a jsou to sloupce `created`, `filled` a `controlled`. První sloupec `created` označuje, kdy byl záznam přidán do tabulky. Následující sloupec `filled` označuje datum a čas, kdy student vyplnil otázku. Poslední sloupec `controlled` označuje, kdy byla odpověď zkontrolována. To je buď čas, kdy byla otázka zkontrolována automaticky, pokud má nastavenou automatickou kontrolu odpovědi, nebo čas, kdy odpověď zkontroloval učitel.



Obrázek 4.4: ER diagram databáze

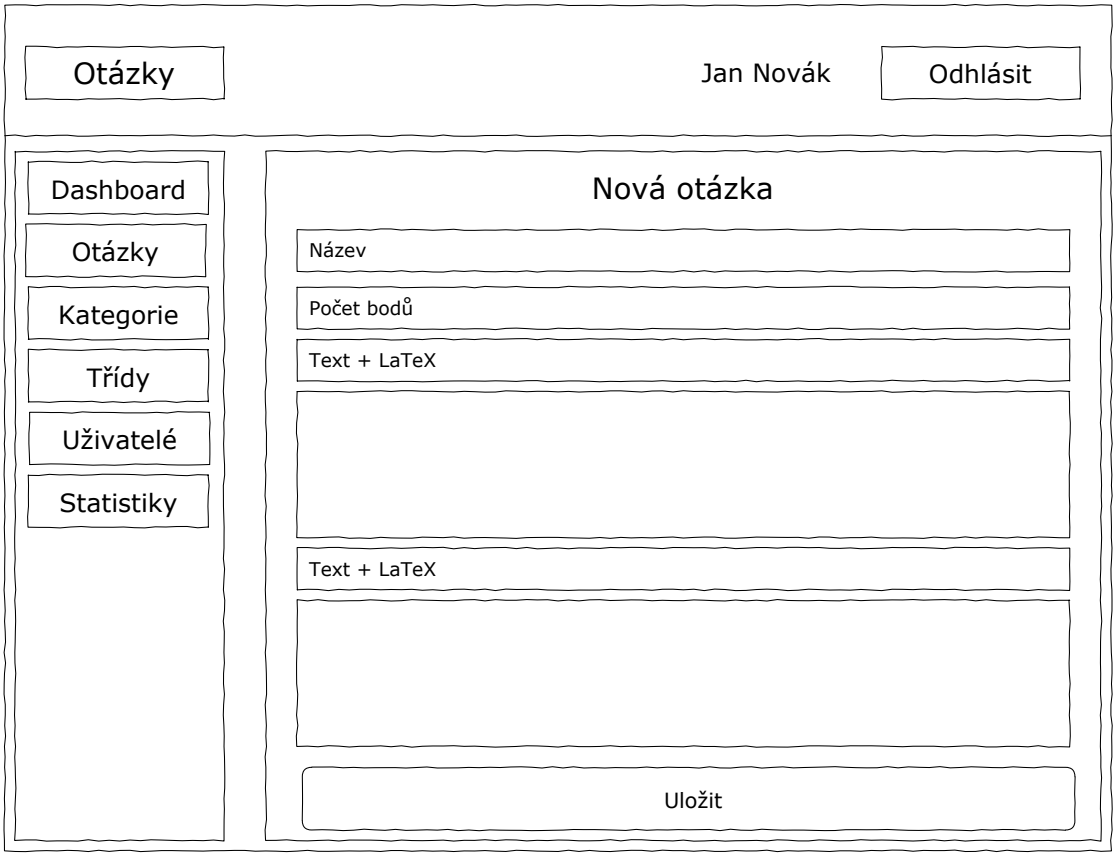
4.3 Uživatelské rozhraní

Návrh uživatelského rozhraní je velmi důležitá část při návrhu aplikací. Pokud aplikace obsahuje velké množství funkcí, ale má špatně navrženo uživatelské rozhraní, uživatel tyto funkce mnohokrát ani nenajde a tedy nepoužije. I proto jsem do návrhu specifikací aplikace zařadil bod, aby aplikace měla moderní a přehledný vzhled.

4.3.1 Drátěný model

Pro návrh uživatelského rozhraní jsem využil drátěný model. Drátěný model neboli *wireframe* je skica, která se používá pro rozvržení základní struktury webové aplikace. Slouží pro návrh rozmístění funkčních prvků na stránce. Nejedná se však o grafický návrh, neboť je tvořen pouze pomocí čar a textu. Taktéž by neměl obsahovat rozdílné barvy, až na výjimečné případy, kdy je potřeba některé prvky odlišit.

Na obrázku 4.5 můžeme vidět drátěný model obrazovky pro přidání otázky. Tento model byl vytvořen v on-line webové aplikaci `draw.io`.



Obrázek 4.5: Drátěný model přidání otázky

Jak můžeme vidět na návrhu výše, vzhled aplikace jsem rozdělil na 3 základní části. Nahoře je informační panel, ve kterém se v levé části zobrazuje název části aplikace, ve které se uživatel aktuálně nachází. Vpravo v tomto panelu je pak zobrazeno jméno přihlášeného uživatele a vedle něj je tlačítko pro odhlášení.

V levé části se nachází navigační panel s odkazy do jednotlivých částí aplikace. Ve středu aplikace se následně nachází panel, který zobrazuje samotný obsah aktuálně vybrané části. Na mobilních zařízeních se navigační panel skryje a je zobrazen pouze horní informační panel společně s panelem pro samotný obsah.

4.3.2 Práce s otázkami a odpověďmi

Práce s otázkami a zadávání odpovědí bude nejvyužívanější část aplikace. Proto jsem se snažil vymyslet co nejpohodlnější rozhraní pro uživatele. Zvolil jsem zadávání otázek a odpovědí pomocí WYSIWYG editoru s možností využití LaTeX pro zápis matematiky.

Uživatel tak může jednoduše formátovat svůj text, ale zároveň lze efektivně zadat i složitější zápisy matematiky. Tato varianta přináší velkou výhodu v podobě kvalitního zobrazení příkladů a odpovědí.

Kapitola 5

Použité technologie

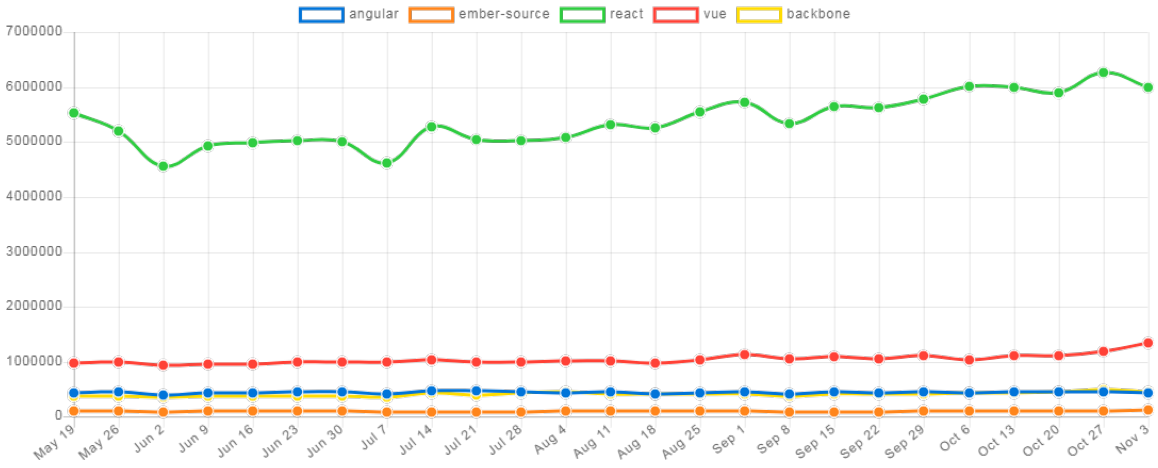
Po návrhu aplikace je potřeba zvolit vhodné technologie pro implementaci. Rozhodl jsem se webovou aplikaci rozdělit na dvě části. Zatímco front-endová část bude implementována v Javascriptu, back-endová část bude napsána v jazyku PHP.

5.1 Front-end

Front-endová část aplikace bude mít na starost obsluhu uživatelských požadavků, je tedy nutné navrhnout vhodné uživatelské prostředí. Níže uvedu technologie, které budu používat pro implementaci front-endu.

5.1.1 React

React je Javascriptový framework, který vydala společnost Facebook v květnu 2013. Tento framework slouží pro tvorbu uživatelského rozhraní. Z hlediska MVC architektury řeší tedy *view* část, která prezentuje data uživateli. Mezi nejznámější weby, které tento framework používají, patří Facebook, Instagram či Netflix. Na obrázku 5.1 můžeme vidět, že aktuálně je React nejpoužívanějším Javascriptovým frameworkem.



Obrázek 5.1: Popularita JS frameworků v roce 2019 (zdroj: [14])

Základním elementem frameworku React jsou takzvané komponenty. Komponenty jsou znovupoužitelné HTML elementy se zapouzdřenou funkcionalitou. Skládáním těchto komponent následně vzniká komplexní UI aplikace. Komponenty mají své vlastnosti (props) a svůj vnitřní stav (state). Vlastnosti komponent (props) slouží pro vzájemnou komunikaci mezi komponenty. Můžeme tak předávat data mezi komponenty pomocí parametrů, stejně jako bychom zapisovali atributy HTML elementů. Stavy komponent se využívají pro data, která se při běhu aplikace mění. Pokud se stav komponenty změní, automaticky se celá komponenta překreslí.

React využívá takzvaný virtuální DOM. To znamená, že si React sestaví svůj virtuální DOM a pokud se nějaká komponenta pozmění, porovnává rozdíly se skutečným DOMem a co nejefektivnějším způsobem ho aktualizuje. To nám dává velkou výhodu, vzhledem k tomu, že práce se samotným DOMem je velmi pomalá a tím, že React umožňuje aktualizovat pouze jeho části, se jeho výkonnost rapidně zvyšuje.

Další důležitá vlastnost při používání Reactu je životní cyklus komponent. Každá komponenta prochází vlastním životním cyklem. Životní cyklus komponenty začíná její inicializací a končí jejím odstraněním. Životní cyklus komponenty má tři fáze - *mounting*, *updating* a *unmounting*. Každá tato fáze má dané metody, kterými je reprezentována. Mezi nejpoužívanější metody patří:

- `componentDidMount` - provede se, jakmile se komponenta vytvoří a vloží do DOMu.
- `render` - metoda, kterou musí obsahovat každá komponenta a vrací to, co se má vykreslit.
- `componentWillUnmount` - provede se, než se komponenta odstraní.

React se nejčastěji používá pro tvorbu *single-page* aplikací. Single-page aplikace znamená, že webová aplikace zobrazuje pouze jednu webovou stránku, která následně svůj obsah mění pouze s pomocí Javascriptu. Další možností využití je použití Reactu pro vykreslení webových stránek na straně serveru (v kombinaci s knihovnou Next.js v serverovém prostředí Node.js). Zajímavým projektem, týkající se Reactu, je projekt React Native, který slouží pro tvorbu mobilních aplikací pomocí Reactu.

Verze Reactu 16.8 přinesla zásadní novinku, kterou byla takzvané React Hooks. Ta vyřešila hned několik problémů, které dříve nastávaly. Jeden z hlavních problémů, který byl díky React Hooks vyřešen, je ten, že do této verze platilo, pokud uživatel chtěl vytvořit komponentu, která měla vnitřní stav či životní cyklus, musel volit třídní komponentu. Pokud tyto věci nepotřeboval, mohl zvolit funkcionální komponentu, která je jednodušší pro zápis a nemusíme zápasit se slovem `this`, které dělalo v Javascriptu často vývojářům problémy.

Hooks však tento problém odstraňují, a to díky primitivům, které přidává. Nyní tak vývojáři mohou tvořit funkcionální komponenty a využívat v nich vnitřní stavy či životní cyklus a neřešit tak problémy, které nastávaly při používání třídních komponent. Mezi nejvýznamnější přidané Hooks patří:

- `useState` - práce se stavem.
- `useEffect` - práce s životním cyklem komponenty (pokrývá hned tři okamžiky - zavedení komponenty do DOMu, překreslení komponenty a odebrání komponenty z DOMu).
- `useRef` - rychlý a jednoduchý přístup k DOM elementu.

5.1.2 Redux

Redux je open-source knihovna napsaná v Javascriptu určená pro správu stavu aplikace. Redux se používá s knihovnami, které vykreslují uživatelské rozhraní jako Angular nebo právě React.

Základem Reduxu je takzvaný `store`, který obsahuje veškerý stav aplikace. V celé aplikaci se nachází vždy pouze jeden `store`. Další částí Reduxu jsou `akce`. `Akce` popisují změnu, která má nastat. Neboli, pokud chceme změnit nějaká data ve `store`, musíme to vždy provést pomocí `akce`. `Akce` má vždy atribut *type* a další atributy jsou vyplněny dle potřeby. Poslední důležitou částí Reduxu je `reducer`. Ten provádí samotnou změnu dat ve `store`. `Reducer` je volán ve chvíli, kdy se zavolá `akce`, `store` zavolá samotný `reducer` a předá mu stav aplikace a akci. `Reducer` má v sobě následně kód, který dle identifikace akce provede požadovanou změnu dat a vrátí nový stav aplikace.

5.1.3 React Router

React Router je knihovna, která slouží k nastavení směrování v React aplikacích. Vzhledem k tomu, že se v Reactu nejčastěji tvoří *single-page aplikace*, kde webovou aplikaci tvoří jedna webová stránka, na které se dynamicky mění obsah, je potřeba přidat pro podporu změn URL či vracení se zpět, tak jako na klasických webech, knihovnu pro směrování.

Základem knihovny React Router je `router`, do kterého následně umísťujeme jednotlivé `routy`. Tyto `routy` nám definují, jaká URL má vést k jaké komponentě. React Router také podporuje přesměrování, vnořené `routy` či práci s URL parametry. Pokud používáme React verze 16.8 či novější, React Router obsahuje několik Hooks pro přístup ke stavu `routeru` a provádění přesměrování uvnitř komponent.

5.1.4 Material-UI

Material-UI je jedna z nejpopulárnějších knihoven pro rychlejší tvorbu designu React aplikací. Knihovna má velké množství předpřipravených React komponent, které respektují zásady Material Designu. Material-UI knihovna má předpřipravené jak základní komponenty pro tvorbu rozložení stránky či nadpisy, tak i pokročilé komponenty pro autocomplete či stránkování. Pokud vývojář chce změnit vzhled některých komponent, může využít podporu knihoven jako *CSS modules* či *styled-components*.

5.1.5 Styled components

Styled components je knihovna pro React, která umožňuje stylování jednotlivých komponent pomocí kombinace Javascriptu a CSS. Je to jedna z takzvaných *CSS-in-JS* knihoven, které nám umožňují tvořit graficky upravené komponenty přímo v Javascriptovém kódu. Díky tomu můžeme mít všechny nastylované elementy přímo v komponentě. To znamená, že nemusíme hledat, kde se styly pro daný element vyskytují a zvyšuje se tím přehlednost zdrojového kódu. Navíc díky tomu můžeme jednoduše dynamicky měnit vlastnosti stylů díky možnosti zápisu Javascriptového kódu do stylování.

5.1.6 Material table

Material Table je knihovna pro zobrazení tabulek v Material Designu. Tato knihovna je doporučována pro použití s knihovnou Material-UI. Material Table má velmi široké mož-

nosti využití a umožňuje možnosti jako editování řádků přímo v tabulce, filtrování dat, seskupování dat, exportu dat z tabulky a mnoho dalších.

5.1.7 TinyMCE React

TinyMCE je WYSIWYG editor, který umožňuje jednoduché vytváření HTML kódu i pro lidi, kteří HTML neovládají. Tento editor má podobně jednoduché rozhraní, jako má například Microsoft Word, aby byl co nejjednodušší pro uživatelské použití. TinyMCE React je speciální verze TinyMCE určená pro React, kde je pro něj nachystaná komponenta pro TinyMCE.

5.1.8 @nteract/mathjax

@nteract/mathjax je knihovna od organizace *nteract* pro jednoduché vykreslování matematických vzorců, zadaných v LaTeX, pomocí knihovny MathJax.

5.1.9 Axios

Knihovna Axios je HTTP klient, pomocí kterého lze jednoduše zasílat všechny druhy HTTP požadavků.

5.2 Back-end

Back-endová část aplikace se stará o zpracování požadavků, které budou chodit od front-endové části pomocí REST rozhraní. Tato část bude tedy zpracovávat příchozí data, provádět dotazy do databáze a následně zpět pošle zpracovaná data. Níže uvedu technologie, které budu používat pro implementaci back-endu.

5.2.1 Symfony

Symfony je PHP framework, který je tvořen sadou PHP komponent, které společně výrazně zjednodušují a zrychlují vývoj webových aplikací. Symfony vychází z návrhového vzoru MVC, je open-source a jeho vývoj je sponzorován pařížskou firmou *Sensio Labs*. Tento framework byl z velké části inspirován frameworkem Spring, který se využívá pro vývoj webových aplikací v Javě. Symfony využívá velké množství projektů, mezi které nejznámější patří Drupal, Joomla či PrestaShop. Symfony je společně s frameworkem Laravel nejpoužívanějším PHP frameworkem na světě.

Vývojáři využívají frameworky, protože v sobě obsahují velké množství funkcí, které v PHP jako samotném chybí, nebo se s nimi špatně pracuje. Zdrojový kód aplikací, které využívají frameworky, díky tomu obvykle bývá přehlednější a lépe se udržuje.

Od listopadu 2017 a vydání verze Symfony 4 se stalo Symfony takzvaným mikroframeworkem. To znamená, že si můžeme nainstalovat pouze minimální kostru frameworku a ostatní komponenty případně doinstalovat dle potřeb. Díky tomu lze framework využívat na tvorbu konzolových či REST aplikací.

Symfony má velké množství částí na pokrytí potřeb webových aplikací. Mezi ty nejvyužívanější patří:

- Doctrine - ORM práce s databází
- Dotenv - pro práci s .env soubory

- Mailer - práce s emaily
- PHPUnit - unit testování
- Security - autorizace uživatelů
- Serializer - serializace a deserializace objektů (XML, JSON, Yaml, ...)
- Twig - tvorba šablon

Pro mou aplikaci jsem zvolil verzi 4.4 LTS (Long Term Support), což znamená, že verze bude podporována delší dobu než obvykle, přesněji u této verze je to do listopadu 2022.

5.2.2 Doctrine 2

Doctrine 2 je ORM (Object-Relational Mapping) framework, který nám umožňuje pracovat s daty jako s objekty. Doctrine 2 využívá pro definici vlastností jednotlivých entit komentářové anotace. Pro dokonalé odstínění aplikace od databáze lze využít DQL (Doctrine Query Language), kde se využívají názvy entit a jejich proměnných místo SQL dotazů. Doctrine 2 je již od základu stavěno s ohledem na co nejvyšší výkon. Proto se u něj využívá několik úrovní cachování. Framework je rozdělen do tří vrstev:

Common Definuje základní třídy, knihovny a rozhraní. Zahrnuje nástroje pro práci s anotacemi, cachováním, kolekcemi a podobně. Tyto nástroje jsou následně využívány vyššími vrstvami.

DBAL DBAL neboli (DataBase Abstraction Layer) abstrahuje aplikaci od konkrétního typu databáze. Rozšiřuje PDO, ale umí pracovat i s jinými databázovými ovladači. Tato vrstva má také na starost DQL.

ORM ORM je nejvyšší vrstva Doctrine 2, která zajišťuje samotné mapování objektů na relační databázi.

5.2.3 JWT

JWT neboli *JSON Web Token* je jeden ze způsobů pro bezpečnou výměnu informací mezi dvěma stranami. JWT jako takový je JSON objekt, který se skládá ze 3 části: hlavičky (header), dat (payload) a podpisu (signature). Hlavička obsahuje informace o použitém algoritmu, data obsahují informace o samotném uživateli a následuje podpis, který obsahuje kontrolní součet dat.

Pro jednodušší práci s JWT autentizací budu využívat knihovnu `LexikJWTAuthenticationBundle` určenou pro Symfony. Tato knihovna usnadňuje použití autentizace za pomocí JWT formou využití autorizační hlavičky či umístění tokenu v cookies.

5.2.4 Rest Bundle

Pro jednodušší práci s REST rozhraním na back-endu jsem používal *FOSRestBundle* určený pro Symfony. Ten pomáhá s jednoduchou definicí URL a HTTP metod, díky možnosti zapsání těchto informací pomocí anotací k jednotlivým metodám. Rovněž umožňuje přímé parsování dat z požadavků do objektů a jednoduché vytvoření `view` pro zaslání dat zpět s příslušným HTTP kódem.

5.3 Ostatní technologie použité pro implementaci

Níže uvádím technologie, které mi pomáhaly se samotnou implementací práce.

5.3.1 Docker

Docker je open source projekt, určený pro nasazení aplikací jako přenosných, soběstačných kontejnerů. Kontejner obsahuje pouze požadovanou aplikaci a pro ni potřebné soubory, ale neobsahuje operační systém. To je velká výhoda, protože je tím dosaženo menší velikosti a větší flexibility. Hlavní důvod mého využití při implementaci je častý přechod mezi počítači s operačními systémy Windows a Linux, což Docker velmi ulehčuje.

5.3.2 Composer

Composer je nástroj pro správu závislostí v PHP. Umožňuje nám deklarovat knihovny, které náš PHP projekt potřebuje a následně je instalovat a aktualizovat. Dále nám umožňuje definovat si zkratky ke skriptům či nadefinovat cesty pro automatické načítání tříd projektu. Pomocí Composeru se také zakládají projekty v několika PHP frameworcích, včetně projektů v Symfony. Repositář, který Composer využívá pro instalaci a aktualizaci knihoven, je *Packagist*.

5.3.3 NPM

NPM (Node Package Manager) je podobně jako výše zmíněný Composer správce balíčků, ale tentokrát pro Javascript. Umožňuje nám knihovny potřebné pro projekt deklarovat, nainstalovat a následně aktualizovat.

Kapitola 6

Implementace

V této kapitole budu popisovat vybrané části implementace webové aplikace. Implementace vychází z návrhu specifikací v kapitole 3.5 a návrhu webové aplikace v kapitole 4. Při implementaci budu využívat technologie z kapitoly 5.

6.1 Tvorba kontejnerů

První krok, který jsem při implementaci provedl, byl vytvoření a nastavení kontejnerů s aplikacemi, které budu využívat v Dockeru. Těchto kontejnerů vzniklo nakonec 7, zde je jejich výčet:

- Nginx - webový server
- Php:fpm - verze 7.3.6
- Mysql - databáze, verze 8
- Adminer - nástroj pro jednoduchou správu databáze (jednodušší verze PhpMyAdminu)
- Mailcatcher - zachytávání emailů na localhostu
- SFTP - pomocný kontejner pro rychlejší běh aplikace na Windows
- Nginx-proxy - pro běh více webových aplikací v Dockeru zároveň

Vytvoření těchto kontejnerů mi umožnilo jednoduchou přenositelnost back-endové části aplikace mezi operačními systémy při vývoji. Rovněž přidává možnost rychlého přenesení či otestování aplikace, bez nutnosti zdlouhavé instalace všech potřebných závislostí aplikace či nahrání aplikace na webový server.

6.2 Vytvoření databáze

Vzhledem k rozhodnutí využití ORM Doctrine jsem převedl schéma databáze z kapitoly 4.2 na jednotlivé entity, které Doctrine využívá. Využil jsem k tomu Symfony komponentu `maker-bundle`, která přidává několik konzolových příkazů pro urychlení procesů, mezi které například patří právě tvorba entit. Postupně jsem tedy jednotlivé tabulky z navrhnutého schéma databáze převedl pomocí příkazu `make:entity`.

Následně jsem pomocí Doctrine vytvořil migrace pro nahrání tabulek do databáze pomocí komponenty `maker-bundle` a jejího příkazu `make:migration` a spustil migraci příkazem `doctrine:migrations:migrate`.

6.3 Vytvoření kontrolerů

V aplikaci jsou celkem 4 kontrolery. Všechny kontrolery používají REST rozhraní a implementují základní CRUD operace (Create - vytvoření, Read - čtení, Update - editace a Delete - smazání). Cesty k jednotlivým metodám v kontroleru se řeší pomocí anotací.

Ke každému kontroleru níže přidám tabulku s přehledem cest pro REST rozhraní. Tabulka obsahuje cestu, HTTP metodu, požadovanou roli k použití funkce (administrátor může používat všechny metody jako učitel) a jednoduchý popis. Všechny cesty začínají prefixem */api*.

6.3.1 QuestionController

Kontroler `QuestionController`, jak již jeho název napovídá, slouží pro práci s otázkami. Obsahuje samozřejmě základní operace pro práci s otázkami, ale kromě těchto metod obsahuje metody pro nahrání souboru s otázkou či odpovědí, kontroly odpovědí, získání informací o odpovědi na danou otázku, či metodu pro ruční kontrolu odpovědí studentů.

Cesta	Metoda	Role	Popis
/question	GET	-	Získání všech otázek
/question/{id}	GET	Učitel	Informace o dané otázce
/question	POST	Učitel	Přidání otázky
/question/{id}	PUT	Učitel	Úprava otázky
/question/{id}	DELETE	Učitel	Smazání otázky
/question/upload	POST	Učitel	Nahrání souboru s otázkou
/question/upload-answer	POST	Student	Nahrání souboru s odpovědí
/question/{id}/complete	POST	Student	Odpovězení otázky
/answer/{id}	GET	Učitel	Získání informací o studentské odpovědi
/answer/{id}	POST	Učitel	Ruční vyhodnocení odpovědi studenta

6.3.2 CategoryController

Tento kontroler kromě základních CRUD operací s kategoriemi obsahuje metody pro získání studentských kategorií (kategorie ve třídách, ve kterých je student přiřazen), získání aktuální otázky v dané kategorii či nastavení pořadí otázek v kategorii.

Cesta	Metoda	Role	Popis
/category	GET	-	Získání všech kategorií
/category/{id}	GET	Učitel	Informace o dané kategorii
/category	POST	Učitel	Přidání kategorie
/category/{id}	PUT	Učitel	Úprava kategorie
/category/{id}	DELETE	Učitel	Smazání kategorie
/category/{id}/question-order	POST	Učitel	Nastavení pořadí otázek
/category-student	GET	Student	Získání kategorií pro přihlášeného studenta
/category/{id}/get-question	GET	Student	Získání otázky pro danou kategorii

6.3.3 UserController

`UserController` obsahuje CRUD operace pro uživatele, dále metody pro aktivaci uživatelského účtu, vyřízení požadavku na resetování zapomenutého hesla, změnu uživatelského hesla a získání statistik k danému uživateli.

Cesta	Metoda	Role	Popis
/users	GET	-	Získání všech uživatelů
/users/{id}	GET	-	Informace o daném uživateli
/users	POST	-	Registrace nového uživatele
/users/{id}	PUT	Admin	Úprava uživatele
/users/{id}	DELETE	Admin	Smazání uživatele
/users/activate/{code}	GET	-	Aktivace uživatelského účtu
/users/reset-password	POST	-	Žádost o obnovení zapomenutého hesla
/users/change-password	POST	-	Obnovení zapomenutého hesla
/users/dashboard	GET	-	Získání uživatelských statistik pro dashboard
/users/statistics	GET	-	Získání uživatelských statistik pro přehled
/users/logout	GET	-	Odhlášení uživatele

6.3.4 StudentClassController

`StudentClassController` slouží pro práci s třídami. Kromě základních operací s třídami tento kontroler obsahuje metody pro získání tříd pro daného studenta či nastavení kategorií přiřazených k dané třídě.

Cesta	Metoda	Role	Popis
/class	GET	-	Získání všech tříd
/class/{id}	GET	-	Informace o dané třídě
/class	POST	Učitel	Přidání třídy
/class/{id}	PUT	Učitel	Úprava třídy
/class/{id}	DELETE	Učitel	Smazání třídy
/class/{id}/category-order	POST	Učitel	Přiřazení kategorií k dané třídě
/class-student	GET	Student	Získání tříd přihlášeného studenta

6.4 Autentizace

Autentizace do aplikace probíhá pomocí JWT. Pro zjednodušení autentizace v aplikaci používám knihovnu `lexik/jwt-authentication-bundle`. V konfiguračním souboru `lexik_jwt_authentication.yaml` jsem nastavil expiraci tokenu na 1 den. Jako metodu autentizace nepoužívám autentizační hlavičku, ale cookie s názvem `BEARER`. Toto nastavení také lze změnit ve výše zmíněném konfiguračním souboru. Přihlášení do aplikace se provádí pomocí HTTP metody `POST` na URL `/api/login` a odhlášení uživatele probíhá na `/api/users/logout` pomocí HTTP metody `GET`.

S obsluhou autentizace pomocí JWT souvisí třída `JWTSubscriber`. Ta obsahuje 3 metody, které reagují na události:

onJwtAuthenticationSuccess Metoda reaguje na úspěšné přihlášení uživatele. Kontroluje, jestli má aktivovaný účet a v případě, že ano, vytvoří novou cookie s vygenerovaným JWT a nastaví ji uživateli pomocí *setCookie*.

onJwTAuthenticatedAccess Tato metoda reaguje na akci přihlášeného uživatele. Spouští se před danou akcí a nastavuje do proměnné `user` informace o přihlášeném uživateli, které získá z tokenu.

onJwtAuthenticatedResponse Metoda, která se spouští před odesláním odpovědi uživateli. Zajišťuje prodloužení expirace tokenu, pomocí vygenerování nového JWT a jeho nastavení pomocí *setCookie*.

6.5 Adresářová struktura back-endové části

Níže uvádím konečnou adresářovou strukturu back-endové části aplikace s jednoduchým popisem obsahu každé složky.

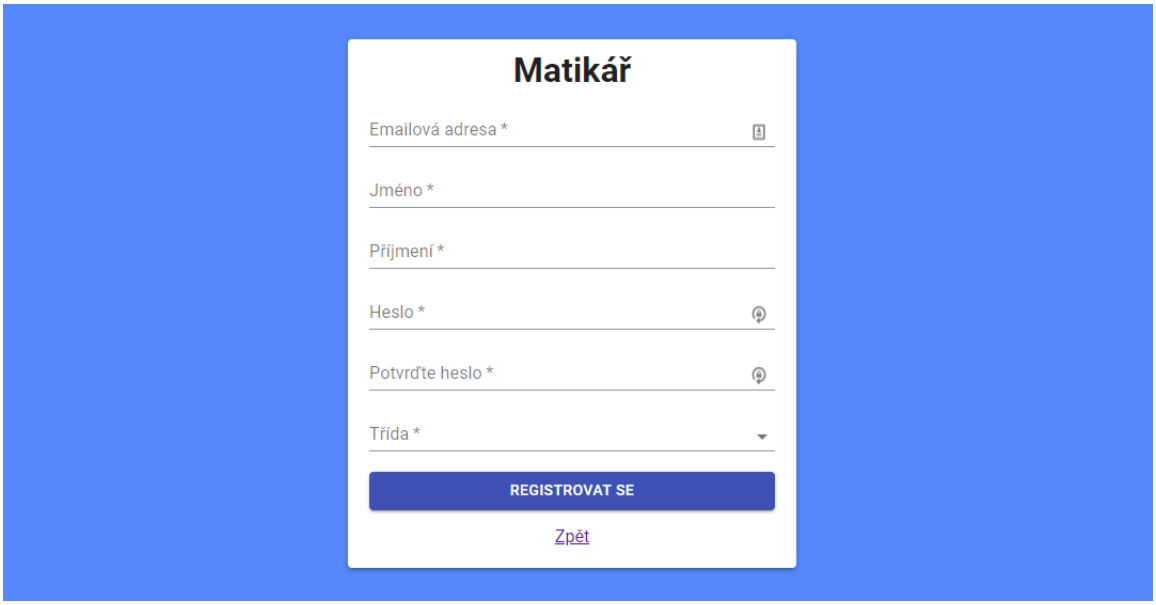
```
/
├── docker ..... Konfigurační soubory pro Docker
├── bin ..... Konzole pro Symfony
├── config ..... Konfigurační soubory pro Symfony
├── public ..... Veřejně přístupné soubory (nahrané otázky a odpovědi)
├── src ..... Zdrojové soubory Symfony
│   ├── Controller ..... Kontrolery aplikace
│   ├── Entity ..... Doctrine entity
│   ├── EventSubscriber ..... Posluchače událostí (JWT)
│   ├── Extensions ..... Vlastní funkce do DQL
│   ├── Migrations ..... Doctrine migrační skripty
│   ├── Model ..... Modely aplikace
│   └── Repository ..... Repozitáře pro jednotlivé entity
├── templates ..... Twig šablony (používáno pro emaily)
├── tmp ..... Dočasné soubory pro Symfony
├── var ..... Logy + cache
└── vendor ..... Složka se závislostmi - spravováno Composerem
```

6.6 Registrace uživatelů

Registrace uživatelů v aplikaci probíhá dvěma způsoby. První způsob, který můžeme vidět na obrázku 6.1, je registrace pomocí registračního formuláře na úvodní stránce aplikace. Tam uživatel vyplní základní údaje, jako je jeho email, jméno, příjmení, heslo a zvolí si jeho primární třídu. Tímto způsobem ovšem každý uživatel, který se zaregistruje, může nabývat pouze role student. Po vyplnění a odeslání formuláře, se studentovi zašle email s odkazem pro aktivaci jeho účtu. Po aktivaci účtu se již student může přihlásit do aplikace.

Druhý způsob, jakým probíhá registrace uživatelů je registrace, kterou provádí administrátor. Administrátor je jediný uživatel, který může registrovat uživatele a přiřadit mu libovolnou roli, nejenom studenta. Zároveň jediný může uživatele později upravovat, měnit mu roli, přiřazovat či odebírat třídy nebo uživatele smazat. Pokud uživatele registruje administrátor, musí administrátor vyplnit údaje o uživateli - emailovou adresu, jméno, příjmení a přiřadit mu roli. Pokud mu přiřadí roli student nebo učitel, tak mu musí ještě přiřadit příslušnou třídu.

Po registraci touto cestou, na vyplněný email ve formuláři přijde email opět s aktivačním odkazem, ovšem tentokrát bude v emailu i vygenerované uživatelské heslo. Toto heslo si může uživatel později po přihlášení kdykoliv změnit. Záměr při tvorbě aplikace byl takový, že administrátor primárně zaregistruje pouze učitele a studenti se budou registrovat sami pomocí registračního formuláře tak, aby tvorba nových uživatelských účtů byla co nejefektivnější.



Obrázek 6.1: Registrační formulář pro studenty

6.7 Správa studentských tříd

Studenti a učitelé jsou zařazeni do určitých tříd. U studentů tyto třídy ovlivňují, k jakým kategoriím budou mít přístup. Studenti mají přístup ke všem kategoriím, které jsou přiřazeny k jejich třídám. Student si při registraci přes registrační formulář volí jeho primární třídu, ovšem tříd může mít více. Ty mu mohou být dodatečně přiděleny administrátorem. Doda-

tečné přiřazení více tříd studentům se může využít například při semináři matematiky pro dobrovolné zájemce. V tomto případě může administrátor přiřadit studentům, kteří mají o tento seminář zájem třídu, kde později budou kategorie, týkající se pokročilého učiva.

Také učitelé mají přiřazeny třídy, tentokrát si je nevolí oni, ale jsou jim přiřazeny administrátorem, který jim vytváří uživatelský účet. U učitelů přiřazené třídy ovlivňují přístup k statistikám studentů pouze ve třídách, které jim jsou přiřazeny. Dále učitelé mohou upravovat kategorie pouze ve třídách, u kterých jsou přiřazeni. Uživatelé s administrátorskou rolí mají přístup ke všem třídám. To znamená, že i ve statistikách mohou vidět všechny studenty.

6.8 Otázky a odpovědi

Než se kategorie přiřadí třídám, je potřeba v nich vytvořit otázky. Otázky mohou být zadány textovou formou s možností využití LaTeX pro matematické vzorce nebo formou obrázku. Textová forma sebou samozřejmě přináší výhody v podobě vektorového zobrazení a většinou je to rychlejší způsob zadání otázky, než řešení různých skenování příkladů. Pokud jsou ovšem příklady složitější na zadání (obzvláště geometrické typy úkolů), je zde možnost využití obrázku.

Odpovědi mohou být trojího typu. Buď mohou být zadány opět textovou formou s možností využití LaTeX pro matematické vzorce nebo formou obrázku, stejně jako je u otázek. Jako poslední možnost je zde dát studentovi na výběr z odpovědí definovaných učitelem při vytváření otázky. Student pak následně pomocí přepínačů zvolí odpověď k otázce.

6.8.1 Studentské odpovědi

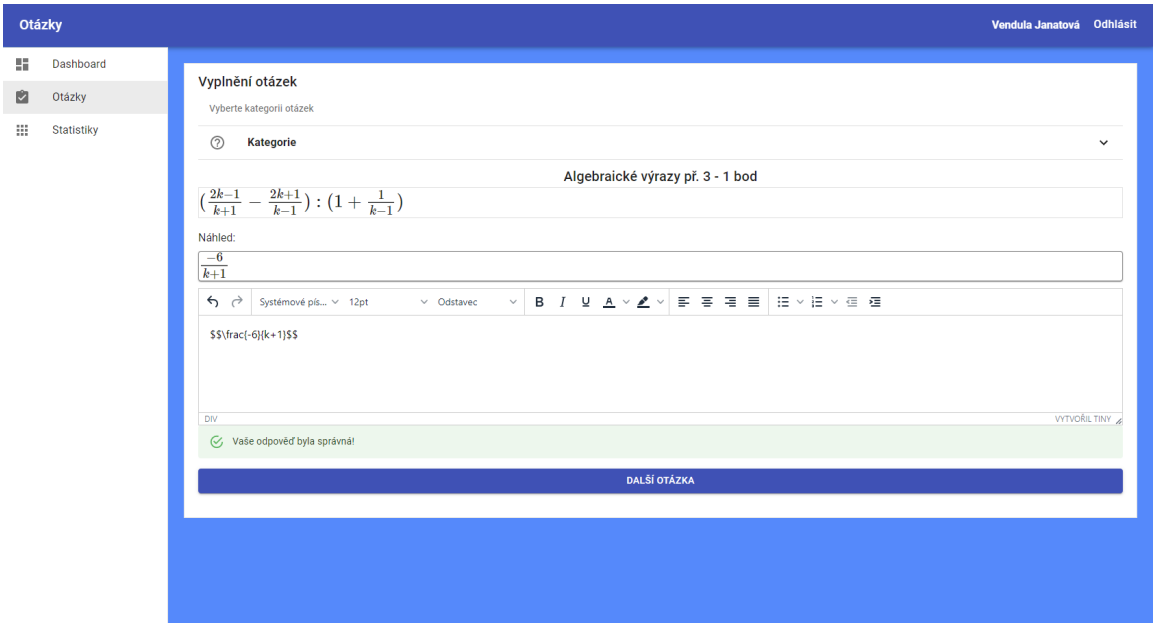
Po přihlášení k uživatelskému účtu si student zvolí kategorii ze seznamu kategorií, ke kterým má přístup (kategorie ve třídách, které má student přiřazeny) a následně je mu zobrazena otázka. Po zodpovězení otázky je studentovi okamžitě dána zpětná vazba o tom, jestli odpověď byla správná nebo ne, pokud učitel zvolil automatické vyhodnocení odpovědi při tvorbě otázky. Pokud tato možnost je vypnutá, odpověď bude čekat na ruční vyhodnocení učitelem, a poté budou případně přičteny studentovi body za správnou odpověď. Ukázku automatického vyhodnocení zodpovězené otázky studenta můžete vidět na obrázku 6.2.

Na obrázku 6.3 je zobrazen seznam kategorií s otázkami pro studenta. Seznam obsahuje několik základních informací, jako aktuální úroveň otázky v kategorii a jednoduché indikátory, jestli jsou v kategorii nějaké otázky, které čekají na vyhodnocení učitelem, nebo jestli naposled zodpovězená otázka v kategorii nebyla poslední.

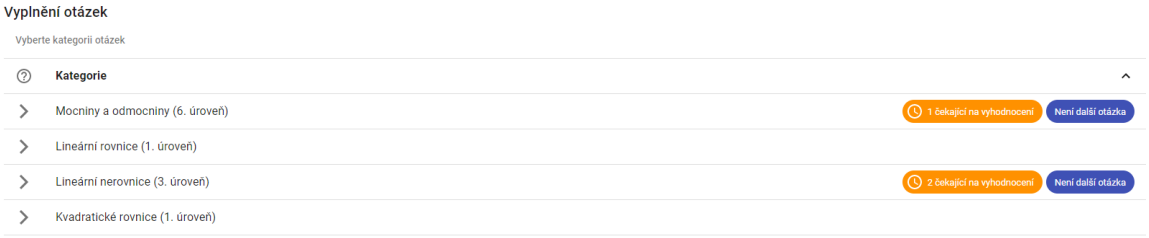
6.8.2 Vyhodnocení odpovědí

Jak již bylo zmíněno v předchozí podkapitole, učitel si při vytváření otázky může nastavit, jestli má být odpověď na danou otázku automaticky vyhodnocována, či ne. Tato možnost je v systému přidána hlavně pro případy, kdy je potřeba vyhodnotit i postup studenta k řešení příkladu, což by automaticky vyhodnotit nešlo.

Automatické vyhodnocení odpovědí lze zapnout či vypnout u typu odpovědi „Text + LaTeX“, naopak u obrázkových odpovědí, je automatické vyhodnocení odpovědi vždy vypnuto. U typu odpovědi „Výběr z odpovědí“, je zase naopak automatické vyhodnocení odpovědi vždy aktivováno.



Obrázek 6.2: Ukázka automatického vyhodnocení zodpovězené otázky studenta

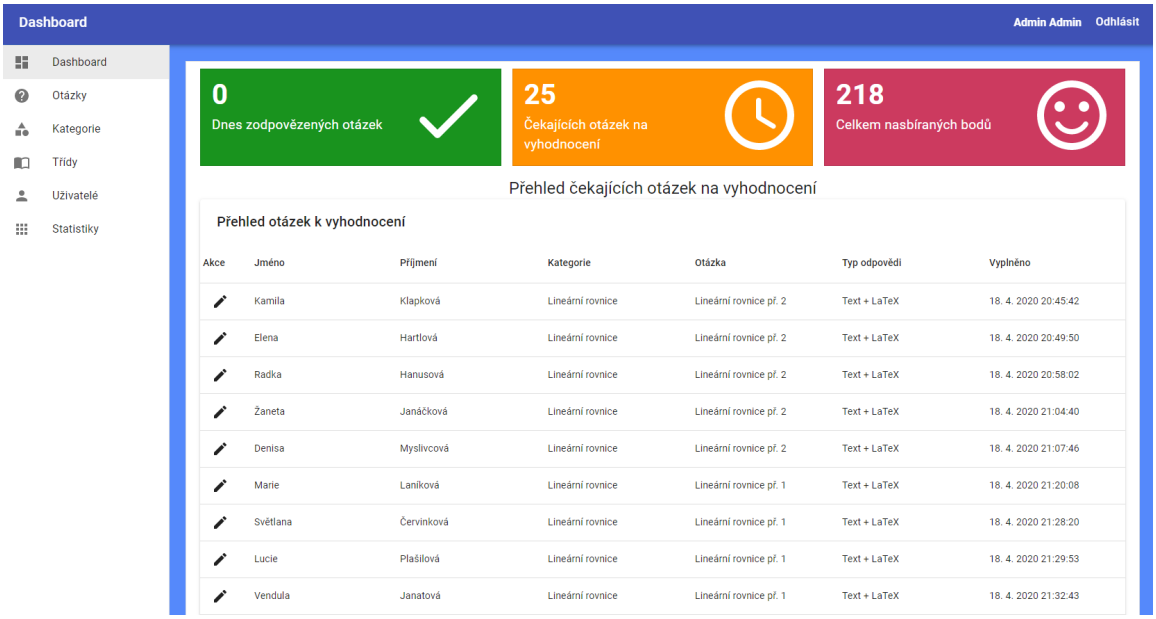


Obrázek 6.3: Ukázka seznamu kategorií s otázkami

6.9 Uživatelský dashboard

Dashboard je část, která je první, co uživatel vidí, jakmile se do aplikace přihlásí. Tato část zobrazuje rozdílné informace pro studenty a učitele. U studentů se na této obrazovce zobrazí zajímavé statistiky, jako je počet otázek, které vyplnil k dnešnímu datu, celkový počet jeho odpovědí, které čekají na ruční vyhodnocení učitelem a celkový počet bodů, které získal. Dále se pod těmito statistikami zobrazí tabulka s přehledem jednotlivých kategorií, do kterých má student přístup a k nim příslušné informace, jako aktuální úroveň otázky, počet správně a špatně zodpovězených otázek a počet získaných bodů v kategorii.

U učitele či administrátora se zobrazí trochu jiné informace. Opět se jim zobrazí statistiky o počtu zodpovězených otázek k dnešnímu datu, počet čekajících otázek na vyhodnocení a celkový počet nasbíraných bodů. Ovšem tentokrát se tyto statistiky zobrazují pro třídy, které má učitel přiřazeny, nebo pro všechny uživatele, v případě, že přihlášený uživatel má roli administrátora. Ukázku konečné formy dashboardu u přihlášeného administrátora můžeme vidět na obrázku 6.4.



Obrázek 6.4: Ukázka dashboardu administrátora

6.10 Adresářová struktura front-end části

Níže uvádím konečnou adresářovou strukturu front-endové části aplikace s jednoduchým popisem obsahu každé složky.

- `/`
  - `node_modules` ... Složka se závislostmi - spravováno NPM
  - `public` ... Veřejně přístupné soubory
  - `src` ... Zdrojové soubory Reactu
    - `components` ... React komponenty
      - `form` ... Formuláře
      - `formComponent` ... Formulářové prvky
      - `navigation` ... Navigační prvky
      - `utilities` ... Ostatní komponenty (TinyMCE editor, načítací overlay)
    - `containers` ... Jednotlivé stránky
      - `student` ... Stránky pro studentskou část
      - `teacher` ... Stránky pro učitelskou část
    - `router` ... Routery
    - `store` ... Soubory pro Redux
      - `reducers` ... Reducery pro Redux
    - `utilities` ... Pomocné soubory (CSS komponenty, nastavení knihoven)

Kapitola 7

Testování

Testování je velmi důležitou součástí vývoje aplikací. Díky testování se dokáže odhalit velké množství chyb ještě předtím, než se aplikace uvede do ostrého provozu. V této kapitole se budu zabývat uživatelským testováním vytvořené webové aplikace.

7.1 Uživatelské testování

Po implementaci aplikace probíhalo uživatelské testování. Do aplikace jsem přidal testovací data v podobě 60 studentských účtů a 2 učitelských účtů. Následně jsem přidal 5 kategorií učiva a v nich dohromady přes 20 otázek.

Celkem se testování účastnilo 5 osob - 2 osoby pro otestování prostředí pro učitele byly studentky posledního ročníku pedagogické fakulty se zaměřením na matematiku a zbylí 3 byli studenti střední školy. Přihlašovací údaje k účtům byly předány jednotlivým osobám účastnících se testu. Účty pro jednotlivé osoby byly již částečně naplněny daty. Úkoly pro uživatelské testování studentské části byly následující:

1. Přihlaste se do aplikace.
2. Proveďte změnu hesla u vašeho účtu.
3. Zjistěte celkový počet vašich bodů.
4. Zjistěte počet získaných bodů v kategorii „Algebraické výrazy“.
5. Odpovězte na otázku z kategorie „Lineární rovnice“.
6. Zobrazte si statistiky vaší třídy a zjistěte uživatele s nejvyšším počtem získaných bodů.
7. Seřaďte si studenty vaší třídy podle příjmení.
8. Odhlaste se z aplikace.

Pro učitelskou část byl seznam úkolů následující:

1. Přihlaste se do aplikace.
2. Proveďte změnu hesla u vašeho účtu.

3. Zjistěte počet čekajících otázek na vaše vyhodnocení.
4. Vyhodnoťte jednu ze studentských odpovědí čekající na vyhodnocení.
5. Přidejte otázku.
6. Přiřaďte otázku do příslušné kategorie, případně ji vytvořte.
7. Vytvořte novou třídu.
8. Přiřaďte danou kategorii do nově vytvořené třídy.
9. V přehledu uživatelů vyhledejte, do kterých tříd patří uživatel s emailem „ladislav-zednicek@itiomail.com“.
10. Zobrazte si statistiky studentů třídy 2.A.
11. Seřaďte si studenty dle počtu získaných bodů a zjistěte studenta z třídy 2.A, který má nejvyšší počet bodů.
12. Odhlaste se.

V tabulkách 7.1 a 7.2 můžeme vidět výsledky úkolů u jednotlivých studentů a učitelů. Znak „fajfky“ označuje, že uživatel u úkolu neváhal a ihned věděl co má dělat. V případě udávaného časového intervalu je udávána doba „přemýšlení“ uživatele, jak se má k řešení úkolu dostat. Při plnění úkolů nebylo testovaným uživatelům nijak napomáháno.

Úkol	Student 1	Student 2	Student 3
1	✓	✓	✓
2	1-2 minuty	do 1 minuty	2 a více minut
3	✓	✓	do 1 minuty
4	✓	✓	✓
5	✓	1-2 minuty	2 a více minut
6	✓	✓	do 1 minuty
7	✓	✓	✓
8	✓	✓	✓
Komentář	+ aplikace je přehledná - žádné	+ líbí se mi moderní vzhled, většina údajů na pár kliknutí - složité zadávání LaTeX rovnic	+ líbí se mi dashboard aplikace - některé funkcionality se hůře hledají, zdlouhavé zadávání LaTeX odpovědí

Tabulka 7.1: Výsledky jednotlivých testů studentů

7.2 Zpracování výsledků a návrhy na zlepšení

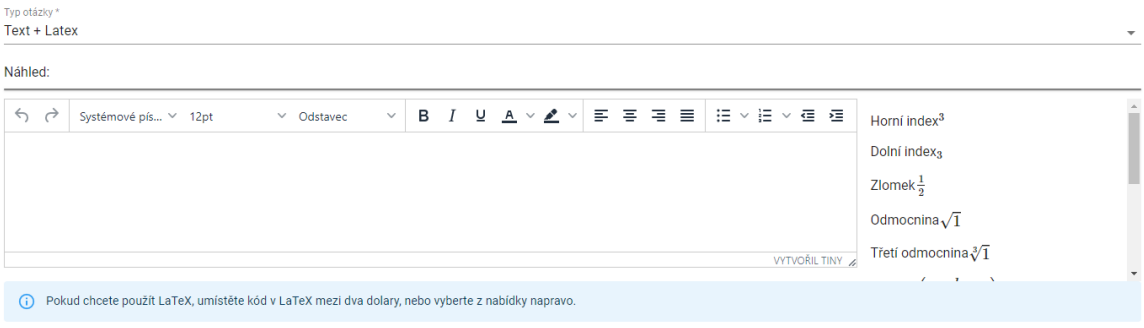
Jak můžeme z výsledků vidět, tak u studentů i učitelů dělali největší problémy úkoly 2 a 5. Úkol 2 se týká u obou změny hesla a úkol 5 byl odpovídání na otázku u studentů, a vytvoření nové otázky u učitelů.

Úkol	Učitel 1	Učitel 2
1	✓	✓
2	1-2 minuty	1-2 minuty
3	✓	✓
4	✓	do 1 minuty
5	do 1 minuty	1-2 minuty
6	✓	do 1 minuty
7	✓	✓
8	do 1 minuty	✓
9	✓	do 1 minuty
10	✓	✓
11	✓	do 1 minuty
12	✓	✓
Komentář	+ přehlednost, možnosti nastavení otázek - není možnost nastavení bodů u kategorií, možno pouze přímo u otázek	+ podrobné statistiky, možnosti nastavení - složité zadávání LaTeXpříkladů

Tabulka 7.2: Výsledky jednotlivých testů učitelů

Pro řešení prvního problémů jsem využil zobrazení nabídky pro změnu hesla, po najetí na jméno uživatele v horní liště. Předtím zde žádné upozornění zobrazeno nebylo, pouze se změnil kurzor myši a uživatelé si tak této možnosti nevšimli, nebo si jí všimli až po delší době hledání.

Pro řešení druhého problému s odpovídáním a vytvářením nových otázek mi uživatelé sdělili, že problém, který je hodně zdržoval při odpovídání na otázky a vytváření nových, je zápis a zapamatování si LaTeX příkazů pro vytváření příkladů. Na základě této zpětné vazby jsem se rozhodl přidat seznam s nejpoužívanějšími příkazy LaTeX, při zadávání otázek a odpovědí. Ukázku této přidané části můžete vidět na obrázku 7.1 v pravé části. Po kliknutí na daný prvek, který chce uživatel přidat, se mu přidá na konec textu.



Obrázek 7.1: Ukázka rozhraní pro přidání příkladu

Kapitola 8

Závěr

Cílem této práce bylo vytvořit webovou aplikaci pro on-line výuku matematiky. Aplikace je směřována hlavně pro výuku na středních školách, ale nic nebrání použití i na základních či vysokých školách. Nejprve byly prozkoumány současné technologie, které se využívají pro vývoj webových aplikací. Pro účel co nejlepšího návrhu aplikace byla poté analyzována již existující řešení, a z nich následně stanoveny body, ze kterých byl stanoven samotný návrh aplikace. Následně proběhl návrh aplikace pomocí diagramu případu užití, ER diagramu databáze a návrh uživatelského prostředí byl realizován pomocí drátěného modelu.

Následně byla samotná aplikace implementována pomocí Symfony frameworku jazyka PHP a frameworku React jazyka Javascript. Po implementaci aplikace následovalo uživatelské testování, které otestovalo hlavně přehlednost uživatelského prostředí. Pro účely testování byla vytvořena sada vhodných úloh. Následné chyby získané uživatelským testováním byly odstraněny. Hlavní body práce tímto byly splněny.

V současné době aplikace běží na `http://bakalarska-prace.dohnalweb.cz`. Do budoucna bych se chtěl pokusit aplikaci rozšířit do škol, na čem již aktivně také pracuji. Další případná vylepšení aplikace budou tak záviset na zpětné odezvě studentů a učitelů na školách.

Literatura

[1] Baar, O. *Historie SQL | PC World.cz* [online]. Internet Info DG, duben 2008 [cit. 2020-04-04]. Dostupné z: https://pcworld.cz/archiv/historie-sql-17391.

[2] Bos, B. *A brief history of CSS until 2016* [online]. W3C, prosinec 2016 [cit. 2020-04-03]. Dostupné z: https://www.w3.org/Style/CSS20/history.html.

[3] DB Engines.com. *Historical trend of the popularity ranking of database management systems* [online]. solid IT gmbh, duben 2020 [cit. 2020-04-04]. Dostupné z: https://db-engines.com/en/ranking_trend.

[4] Garcia, D. J. *The History of ASP.NET – Part I | DotNetCurry* [online]. DotNetCurry.com, duben 2019 [cit. 2020-04-04]. Dostupné z: https://www.dotnetcurry.com/aspnet/1492/aspnet-history-part-1.

[5] Group, T. P. *PHP: History of PHP - Manual* [online]. The PHP Group, březen 2020 [cit. 2020-04-04]. Dostupné z: https://www.php.net/manual/en/history.php.php.

[6] Holzschlang, M. E. *HTML a CSS: jdi do toho*. 1. vyd. Grada, 2006. ISBN 80-247-1454-X.

[7] Hujer, M. *Jaké novinky přinese PHP 7.4 - Zdroják* [online]. Devel.cz Lab, říjen 2019 [cit. 2020-04-04]. Dostupné z: https://www.zdrojak.cz/clanky/jake-novinky-prinese-php-7-4/.

[8] Husar, P. *Informace o projektu – e-Matematika.cz* [online]. Březen 2020 [cit. 2020-04-05]. Dostupné z: https://www.e-matematika.cz/info.php.

[9] Kučera, F. *Java na serveru: úvod - Zdroják* [online]. Devel.cz Lab, leden 2010 [cit. 2020-04-04]. Dostupné z: https://www.zdrojak.cz/clanky/java-na-serveru-uvod/.

[10] ManagementMania. *Třívrstvá architektura (Three-tier architecture)* [online]. Wilmington (DE), prosinec 2015 [cit. 2020-04-03]. Dostupné z: https://managementmania.com/cs/trivrstva-architektura-three-tier-architecture.

[11] ManagementMania. *Architektura klient-server (Client–server model)* [online]. Wilmington (DE), listopad 2016 [cit. 2020-04-03]. Dostupné z: https://managementmania.com/cs/architektura-klient-server.

[12] ManagementMania. *Relační databáze* [online]. Wilmington (DE), březen 2016 [cit. 2020-04-04]. Dostupné z: https://managementmania.com/cs/relacni-databaze.

[13] Marianne Hauser, C. W. *HTML a CSS: Velká kniha řešení*. 1. vyd. Computer Press, 2006. ISBN 80-251-1117-2.

[14] Martin, S. *List of Top JavaScript Frameworks 2020 For Front-End Developers* [online]. Free Code Camp, listopad 2019 [cit. 2020-04-09]. Dostupné z: https://www.freecodecamp.org/news/complete-guide-for-front-end-developers-javascript-frameworks-2019/.

[15] Matthew Macdonald, A. F. a. M. S. *ASP.NET 4 a C# 2010: tvorba dynamických stránek profesionálně*. Zoner Press, 2011. ISBN 978-80-7413-131-8.

[16] Monus, A. *PHP 7: 10 Things You Need to Know - Hongkiat* [online]. Hongkiat, únor 2016 [cit. 2020-04-04]. Dostupné z: https://www.hongkiat.com/blog/php7/.

[17] Moodle. *MoodleDocs* [online]. 2020 [cit. 2020-04-05]. Dostupné z: https://docs.moodle.org/.

[18] Q Success. *Usage Statistics and Market Share of Server-side Programming Languages for Websites, April 2020* [online]. W3Techs.com, duben 2020 [cit. 2020-04-04]. Dostupné z: https://w3techs.com/technologies/overview/programming_language.

[19] Summerfield, M. *Python 3: výukový kurz*. 1. vyd. Computer Press, 2010. ISBN 978-80-251-2737-7.

[20] W3C. *World Wide Web Consortium (W3C)* [online]. 2020 [cit. 2020-04-03]. Dostupné z: https://www.w3.org/.

[21] Škultéty, R. *JavaSript: programujeme internetové aplikace*. 2. vyd. Computer Press, 2004. ISBN 80-251-0144-4.

Příloha A

Obsah přiloženého paměťového média

Přiložené CD má následující adresářovou strukturu:

- `frontend/` - Zdrojové kódy front-endové části aplikace.
- `backend/` - Zdrojové kódy back-endové části aplikace.
- `zprava/` - Zdrojové kódy technické zprávy.
- `databaze.sql` - Inicializační SQL skript pro vytvoření databáze a naplnění základními daty.
- `bp.pdf` - Technická zpráva ve formátu PDF.
- `README` - Instrukce pro instalaci aplikace.

Příloha B

Manuál

V této příloze je popsán postup instalace aplikace. Aplikace pro svůj běh vyžaduje podporu PHP minimálně ve verzi 7.3.6 a využívá MySQL databázi, která musí být minimálně verze 8. Dále je nutno mít přístup ke konzoli, nejlépe na cílovém stroji kvůli využití Composeru a NPM, které musí být nainstalovány.

Databáze, počáteční uživatelské účty

Na začátku instalace je nutné založit MySQL databázi a spustit v ní inicializační SQL skript. Tento skript se nachází v kořenové složce instalačního média a má název `databaze.sql`. Inicializační skript nám zajistí vytvoření potřebných tabulek a naplnění tabulek základními daty.

V aplikaci je od začátku vytvořen pouze jeden účet a ten má roli administrátora. Tento účet má přihlašovací jméno `admin@example.com` a heslo k tomuto účtu je `matikarAdmin`. Pomocí tohoto účtu lze následně vytvořit další účty pro učitele, vytvořit třídy a podobně.

Nastavení konfigurace front-endové části

Pro konfiguraci front-endové části aplikace stačí pouze nastavit proměnnou `REACT_APP_AJAX_SERVER` v souboru `.env`, v kořenové složce aplikace na adresu serveru, kde běží back-endová část.

Nahrání front-endové části na server

Pro kompilaci front-endové části aplikace spustíme ve složce `frontend/` příkaz `npm install` pro instalaci všech potřebných závislostí a následně `npm run-script build` pro kompilaci front-endové části. Zkompilovaná aplikace se následně nachází ve složce `build/`, ze které překopírujeme všechny soubory na cílový server. Dále je nutno překopírovat na server soubor `.htaccess`, který zajišťuje správné směrování parametrů požadavků.

Nastavení konfigurace back-endové části

Kompletní konfigurace back-endové části aplikace se nachází v souboru `.env`, v kořenové složce aplikace. Před spuštěním aplikace je tak nutno nastavit několik proměnných pro nastavení adresy, kde běží front-endová část aplikace, databáze a SMTP serveru.

APP_ENV Nastavení prostředí aplikace. Může nabývat hodnot *prod* pro produkční běh aplikace nebo *dev* pro vývojové prostředí. Rozdíl spočívá hlavně v detailnějším zobrazení důvodu chyb, v případě nastavení vývojového prostředí.

APP_FRONTEND Nastavení URL, kde běží front-endová část aplikace. Nutno nastavit pro správné nastavení odkazů na tlačítko v příchozích mailech pro aktivaci účtu či obnovení zapomenutého hesla.

DATABASE_URL Nastavení adresy, kde běží databáze. Tato proměnná se nastavuje ve stylu `typ-databaze://jmeno:heslo@adresa:port/nazev-databaze`.

MAILER_DNS Nastavení adresy SMTP serveru. Proměnná se nastavuje ve stylu `smtp://adresa-serveru:port`.

CORS_ALLOW_ORIGIN Nastavení adresy, která má oprávnění posílat dotazy na back-endovou část serveru. Zapíšeme sem tedy adresu, na které nám běží front-endová část aplikace. Adresu je nutno zapsat ve stylu `^http://adresa-frontendu$`.

Nahrání back-endové části na server

Pokud nemáme přístup ke konzoli na cílovém serveru, spustíme instalaci závislostí aplikace před nahráním na server. Instalace závislostí se provádí příkazem `composer install`. Po instalaci všech potřebných závislostí, překopírujeme všechny soubory ze složky `backend/` do cílové složky na serveru. Následně musíme zajistit nastavení práv složek `tmp` a `var` alespoň na 775.