UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2025 Matěj Hauschwitz

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Vývoj webové aplikace pro výuku matematiky na základní škole

Bakalářská práce

2025 Matěj Hauschwitz

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Matěj Hauschwitz**
Osobní číslo:	**I21323**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Vývoj webové aplikace pro výuku matematiky na základní škole**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem této bakalářské práce je vytvořit moderní a uživatelsky přívětivou webovou aplikaci, která umožní efektivní výuku matematiky na základní škole. Aplikace bude nabízet širokou škálu matematických příkladů, od základních po pokročilé, a umožní uživatelům nejen spočítat příklady, ale také se seznámit s krok za krokem jejich řešení.

Rozsah pracovní zprávy: **min. 30 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

VRÁNA, Jakub. 1001 tipů a triků pro PHP. Brno: Computer Press, 2010. ISBN 978-80-251-2940-1.
LAURENČÍK, Marek. Tvorba www stránek v HTML a CSS. Praha: Grada Publishing, 2019. Průvodce (Grada). ISBN 978-802-7122-417.

Vedoucí bakalářské práce: **Ing. Jan Panuš, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. prosince 2023**
Termín odevzdání bakalářské práce: **10. května 2024**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2024

Prohlašuji:

Práci s názvem Vývoj webové aplikace pro výuku matematiky na základní škole jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že 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, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 11. 5. 2025

Matěj Hauschwitz

PODĚKOVÁNÍ

Tímto bych rád poděkoval mému vedoucímu bakalářské práce Ing. Janu Panušovi, Ph.D. za odborné konzultace, cenné rady, vedení a trpělivost při vypracování této bakalářské práce. Dále bych chtěl poděkovat své rodině za podporu během studia.

ANOTACE

Cílem této bakalářské práce je vývoj webové aplikace určené pro výuku matematiky na základní škole. Aplikace umožňuje žákům procvičovat matematické příklady z různých tematických oblastí, jako jsou rovnice, sčítání, odčítání, násobení, dělení či nerovnice. Uživatel si volí téma a obtížnost, na základě čehož jsou automaticky generovány příklady. Po vyřešení příkladu je zobrazena zpětná vazba a statistiky, které umožňují sledovat úspěšnost. Součástí aplikace je také administrátorské rozhraní pro správu celé aplikace.

KLÍČOVÁ SLOVA

Webová aplikace, výuka matematiky, základní škola, PHP, Symfony, generování příkladů, procvičování, statistiky

TITLE

Development of a Web Application for Teaching Mathematics at Primary School

ANNOTATION

The aim of this bachelor's thesis is to develop a web application designed for teaching mathematics at primary school level. The application allows pupils to practice math problems from various topics such as equations, addition, subtraction, multiplication, division, and inequalities. Users select a topic and difficulty level, upon which examples are automatically generated. After solving the problem, feedback and statistics are displayed to help track performance. An admin interface is also included for managing the application

KEYWORDS

Web application, math education, primary school, PHP, Symfony, problem generation, practice, statistics

Obsah

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM VÝPISŮ PROGRAMOVÉHO KÓDU

SEZNAM ZKRATEK A ZNAČEK

SCSS	Preprocesor CSS
jQuery	JavaScriptová knihovna
Twig	Šablonovací jazyk
PHP	Skriptovací jazyk pro web
SQL	Dotazovací jazyk pro databáze
CRUD	Operace Vytvořit, Číst, Aktualizovat, Smazat
ORM	Mapování objektů na databázi
DBAL	Abstrakce databázové vrstvy
URL	Adresa zdroje na webu
XSS	Cross-Site Scripting (zranitelnost)
DOM	Document Object Model (model objektů dokumentu)
RBAC	Role-Based Access Control (řízení přístupu na základě rolí)
ID	Identifikátor
CSV	Comma-Sepated Values (formát souboru)
HTML	HyperText Markup Language
AJAX	Asynchonous JavaScript and XML

Úvod

Matematika je jedním ze základních pilířů vzdělávání a hraje klíčovou roli v rozvoji logického a analytického myšlení. Přestože je součástí školních osnov od prvních ročníků, mnoho žáků ji vnímá jako náročný a nezajímavý předmět. Moderní technologie však nabízejí nové možnosti, jak výuku zpřístupnit, zatraktivnit a přizpůsobit individuálním potřebám žáků. Interaktivní webové aplikace představují efektivní nástroj, který může doplnit klasickou výuku a nabídnout žákům prostor pro samostatné procvičování.

Cílem této bakalářské práce je navrhnout a vytvořit webovou aplikaci zaměřenou na výuku a procvičování základních matematických dovedností, určenou primárně pro žáky základních škol. Aplikace umožňuje automatické generování matematických příkladů v několika tématech a úrovních obtížnosti. Žáci mají možnost si zvolit typ úloh a následně je interaktivně řešit. Po vyřešení každého příkladu obdrží okamžitou zpětnou vazbu a aplikace zároveň ukládá jejich výsledky, aby bylo možné sledovat pokrok.

Součástí systému je také administrátorské rozhraní, které umožňuje správu témat, konfiguraci testů a dohled nad statistikami uživatelů. Celý systém je postaven na technologii PHP s využitím frameworku Symfony, databázového systému a dalších webových technologií.

Tato práce se v teoretické části věnuje zejména požadavkům na aplikaci a také použitým technologiím pro implementaci. Praktická část je zaměřena na samotný vývoj aplikace, návrh uživatelského rozhraní a implementaci jednotlivých funkcionalit.

1 Požadavky aplikace

Tato kapitola definuje požadavky na aplikaci. Obsahuje funkční požadavky, které popisují hlavní funkce systému, a nefunkční požadavky zaměřené na kvalitu a provoz aplikace.

1.1 Funkční požadavky

Generování příkladů

Aplikace umožní generovat matematické příklady z různých témat. V každém tématu bude možné generovat příklady podle obtížností, například lehká, střední, těžká.

Řešení příkladů s vysvětlením kroků

Pro každý vygenerovaný příklad aplikace zajistí správný výsledek a popis kroků, které je potřeba provést k dosažení výsledku. Zároveň porovná odpověď uživatele se správnou odpovědí a zobrazí uživateli, jestli je odpověď správná, nebo špatná.

Testování

Aplikace bude umožňovat administrátorům vytvářet testy pro uživatele. Tyto testy se budou zaznamenávat do statistik, aby mohl jak uživatel, tak i administrátor sledovat zlepšení, či zhoršení výkonů daného uživatele.

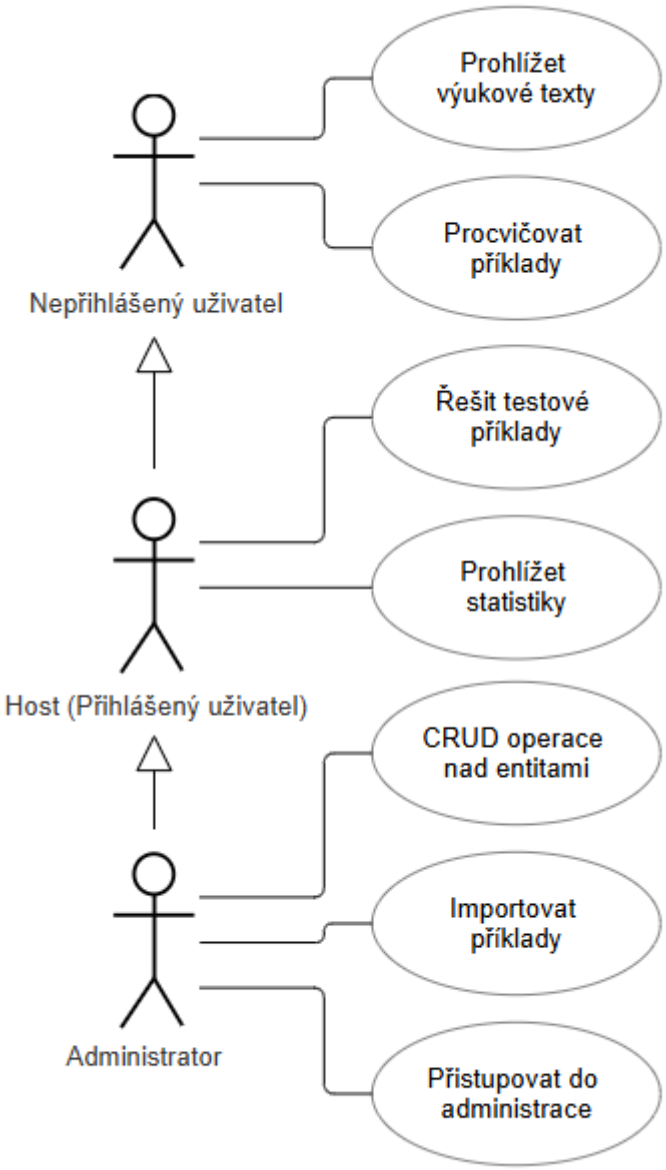
Administrace

Aplikace umožní administrátorům spravovat veškeré entity, tak aby ani administrátor nemohl změnit, přidat, nebo odebrat nějakou entity, která by později v aplikaci mohla vyvolat výjimku.

Uživatelské role

Aplikace bude podporovat tři hlavní role.

- **Nepřihlášený uživatel**: Má přístup k procvičování generovaných příkladů, výukovým textům. Nemá přístup k testům, a tedy se neukládají ani jeho postupy v testech. Nebude mít přístup ke statistikám a ani do administrace.
- **Host**: Má přístup k veškerému výukovému obsahu, může generovat a řešit příklady a jeho postupy i statistiky se ukládají. Tato data mu umožňují sledovat pokrok a zlepšení v čase. Nebude mít přístup do administrace.
- **Administrátor**: Má veškerá práva. Může dělat vše co může dělat host, navíc má práva pro využívání administrace, kde může mazat, upravovat a přidávat všechny entity.



Obrázek 1: Diagram případů užití [zdroj vlastní]

Statistiky uživatelů

Aplikace poskytne uživatelům statistiky k jednotlivým tématům. Statistiky se budou počítat na základě informací z absolvovaných testů, nikoliv z procvičování. Statistiky budou zahrnovat informace jako počet správných a špatných odpovědí, počet pokusů, celkový počet pokusů. Zároveň se uživatel dozví zlepšení, které bude počítáno od posledního pokusu.

Uživatelské účty a přihlášení

Aplikace umožní uživatelům registrovat se a přihlašovat, přičemž bude používat bezpečné metody uchovávání hesel a ochrany osobních údajů. K uživateli se budou pak pojit statistiky a poslední pokusy.

1.2 Nefunkční požadavky

Detekce uživatelských chyb

Aplikace bude upozorňovat uživatele na nesprávně vložené hodnoty do formulářů. Také bude upozorňovat uživatele, pokud nebude mít přístup do nějaké části aplikace.

Výkon a odezva aplikace

Aplikace bude schopna generovat příklady a poskytovat výsledky v reálném čase, bez výrazných prodlev. Optimalizace pro rychlé načítání stránek a efektivní práci s databází.

Bezpečnost uživatelských dat

Aplikace bude chránit osobní údaje uživatelů pomocí šifrování a dalších bezpečnostních praktik. Ochrana proti běžným bezpečnostním hrozbám, jako jsou SQL injection, CSRF a XSS.

Přístupnost a uživatelská přívětivost

Uživatelské rozhraní bude intuitivní a snadno ovladatelné, s přehlednou navigací. Zároveň se uživatelské rozhraní bude přizpůsobovat podle zařízení, bude tedy responzivní. Primárně bude ale aplikace určená pro desktopové zobrazení.

Škálovatelnost a flexibilita

Architektura aplikace bude implementována tak aby bylo jednoduché přidávat nová témata a příklady. Zároveň bude implementována tak aby bylo jednoduché přidávat další nové funkcionality.

2 Technická dokumentace

2.1 Frontend

Frontendové technologie se starají o část aplikace, kterou je uživatel schopný vidět. To zahrnuje jak samotný obsah, tak i stylování, nebo také chování aplikace na základě uživatelské interakce s aplikací.

JavaScript

JavaScript je interpretovaný programovací jazyk, který vznikl v 90. letech díky Brendanovi Eichovi, jenž jej vyvinul pro prohlížeč Netscape Navigator. Na rozdíl od jiných jazyků, které se musí před spuštěním kompilovat, je JavaScript interpretován přímo v prohlížeči, což zajistí rychlejší odezvu při interakcích uživatele s aplikací a aktualizaci obsahu bez nutnosti znovunačtení stránky. Nejčastěji se používá na straně klienta pro tvorbu dynamických webových stránek, ale dá se použít i na backendu, například pomocí Node.js. Ve frontendu se JavaScript využívá k přidávání animací, interaktivních prvků anebo pro validaci formulářů. V rámci této aplikace se JavaScript používá jen pro frontendové účely. Na rozdíl od jiných jazyků se JavaScript zpracovává až ve chvíli potřeby. (1)

jQuery

jQuery je populární knihovna jazyka JavaScript, která poskytuje mnoho užitečných funkcí a byla vyvinuta k usnadnění psaní JavaScriptového kódu. Tato knihovna umožňuje jednodušší procházení a manipulaci s HTML dokumenty, zpracování událostí, vytváření animací a komunikaci se serverem prostřednictvím AJAX, což je technika pro asynchronní načítání dat mezi klientem a serverem. jQuery značně usnadňuje práci tím, že komplexní JavaScriptové funkce redukuje do jednoduchých příkazů a nabízí jednotné rozhraní, které zajišťuje kompatibilitu napříč různými prohlížeči. V této aplikaci je jQuery zvláště užitečný pro snadnější práci s AJAX, který zde slouží k odesílání uživatelských odpovědí při procvičování příkladů. AJAX umožňuje dynamickou komunikaci se serverem, aniž by se musela znovu načítat celá stránka. S využitím jQuery je také mnohem snadnější psaní kódu, který se přizpůsobí rozdílům mezi různými prohlížeči. (2)

SCSS

SCSS je rozšíření tradičního jazyka CSS, které přináší pokročilé možnosti pro organizaci kódu a usnadňuje práci při tvorbě složitějších webových projektů. Nabízí funkce, jako jsou proměnné, vnořené styly, mixiny a dědičnost, což umožňuje vytvořit čistší a strukturovanější kód, který je zároveň snáze udržovatelný. Použití proměnných zajišťuje konzistentní barvy, rozměry a další hodnoty v celém projektu, zatímco vnořené styly umožňují logickou strukturu, která odráží HTML dokument. SCSS podporuje mixiny a funkce, které mohou obsahovat často používané styly nebo výpočty a následně se používat opakovaně. Pro tuto aplikaci byl SCSS použit při návrhu rozvržení, s důrazem na estetiku a konzistenci. Vytvořené styly jsou snadno rozšiřitelné, což zjednodušuje úpravy při rozšiřování aplikace nebo při vizuálních změnách. (3)

Twig

Twig je šablonovací systém pro programovací jazyk PHP. Svou syntaxi odvozuje od systémů Jinja2 a Django. Twig šablony se kompilují do PHP kódu a je poháněn lexerem a parserem, díky tomu je možné přímo v šabloně vytvářet proměnné. Umožňuje snadnější vytváření dynamických šablon HTML. V tomto projektu je použit v rámci Symfony pro přehledný a efektivní vývoj aplikace, protože se její obsah mění v závislosti na interakci uživatele. Umožňuje lepší správu návrhu stránek pomocí šablon, což usnadňuje oddělení logiky od prezentace. (4)

2.2 Backend

Backendové technologie se starají o samotnou logiku aplikace. Dá se říct, že backend je část aplikace, kterou nevidí uživatel. Mezi činnosti, které jsou prováděny backendovými technologiemi mohou spadat různé výpočty, přesměrování, nebo třeba práce s databází.

PHP

PHP je skriptovací jazyk na straně serveru, který se používá pro tvorbu dynamických webových aplikací a hraje zásadní roli v backendu této aplikace. PHP umožňuje efektivní zpracování uživatelských požadavků, správu dat, komunikaci s databázemi a manipulaci se soubory. V této aplikaci slouží PHP především k vytváření dynamického obsahu, zpracování formulářů a spravování uživatelských dat. Jednou z klíčových výhod PHP je jeho široká podpora na různých serverových platformách a snadné použití v rámci běžných hostovacích služeb. PHP je také známý svou rychlostí, což umožňuje rychlé načítání stránek a krátké doby odezvy při uživatelských akcích. V kombinaci s frameworkem Symfony je PHP ideální pro vývoj rozsáhlých webových aplikací, kde poskytuje stabilní základ pro škálovatelné řešení. (5)

Symfony

Symfony je robustní a flexibilní framework pro vývoj v PHP, který přináší strukturu a efektivitu do vývoje webových aplikací. Tento framework byl vybrán pro svou modularitu a podporu komplexních backendových funkcí, jako je autentizace, správa uživatelských účtů, směrování požadavků a bezpečnostní opatření. Díky pokročilým modulům, jako je například Doctrine ORM, usnadňuje Symfony práci s databázemi, umožňuje snadné mapování databázových entit a poskytuje nástroje pro tvorbu bezpečné a škálovatelné aplikace. Symfony také podporuje snadné testování kódu a rychlý vývoj, což umožňuje průběžné rozšiřování aplikace a její údržbu. Tento framework usnadňuje vývoj nejen pro práci s databází, ale také při správě formulářů a validací, čímž zajišťuje konzistentní strukturu a stabilní výkon aplikace. (6)

SQLite

SQLite je lehká a jednoduchá vestavěná databáze, která ukládá všechny informace do jednoho souboru a nevyžaduje instalaci samostatného databázového serveru. V aplikaci se SQLite používá proto, že se snadno používá a nevyžaduje žádnou složitou konfiguraci. Součástí aplikace je databáze SQLite, do které lze snadno ukládat veškerá data týkající se uživatelů, pokusů, statistik a dalších důležitých údajů, což usnadňuje vývoj, testování a sdílení aplikace.

SQLite pomáhá vyhnout se složité konfiguraci nastavení databáze, což je užitečné hlavně pro menší nebo samostatné aplikace. Navíc SQLite v Symfony plně podporuje Doctrine ORM, proto je snadné spravovat data a používat model ORM bez nutnosti nastavovat celý systém správy databází. (7)

Docker

Docker je platforma pro kontejnerizaci aplikací, která umožňuje spouštět aplikace v izolovaných prostředích zvaných kontejnery. V kontextu této bakalářské práce je Docker využit k vytvoření konzistentního a předvídatelného prostředí pro běh aplikace, nezávisle na systému, na kterém je nasazena. Docker kontejner obsahuje všechny potřebné závislosti, včetně verzí PHP, MySQL, Symfony a dalších nástrojů, což usnadňuje vývoj i testování.

Díky Dockeru je možné rychle vytvořit repliku produkčního prostředí, což minimalizuje problémy s kompatibilitou mezi různými počítači nebo servery. To nejen zjednodušuje týmovou spolupráci a nasazování, ale také umožňuje snadné škálování aplikace nebo její migraci na jiný server bez složitých úprav. Docker tedy přispívá k efektivitě vývoje a zajišťuje stabilní běh aplikace v různých prostředích. (8)

GitHub

GitHub je software, který slouží zejména k ukládání verzí projektu. Každá změna v kódu aplikace je spravována pomocí git a to umožňuje sledovat historii změn v projektu. Další využití je automatizace procesu nasazení, kdy se při každém pushi spustí pipeline, což je proces, který zajišťuje integraci a nasazení (CI/CD). Tento proces zahrnuje jak testování, build aplikace, tak i nasazení na server. To zajistí, že změny v kódu jsou ověřeny a aplikace je stabilní. GitHub tedy nejen že slouží jako úložiště pro projekt, ale také jako nástroj pro zajištění kvality a stability kódu díky automatizaci procesů. (9) (10)

3 Architektura aplikace

3.1 MVC

Model-View-Controller je architektonický vzor, který odděluje aplikaci na 3 hlavní části. Toto uspořádání pak definuje úkol jednotlivých částí.

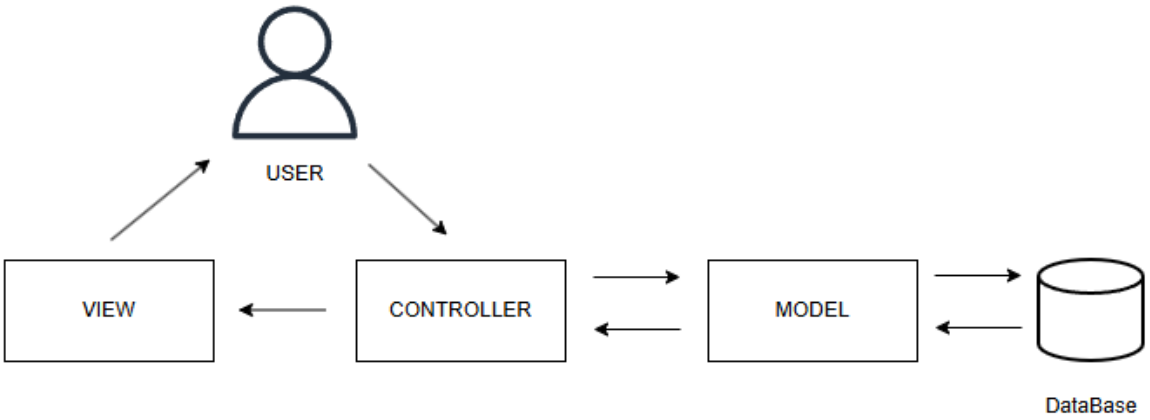
Model má na starost logiku aplikace a práci s daty. Představuje strukturu dat a obsahuje všechny nutné metody pro získávání, ukládání a manipulaci s daty. V této aplikaci model zahrnuje přístup k databázi (pomocí ORM, kterým je Doctrine) a dělá úkony, jako je načítání uživatelských dat, záznam výsledků pokusů. Modelové třídy tedy zajišťují, že data jsou správně uspořádaná a připravená pro další použití v aplikaci. (11)

View je zodpovědné za prezentaci dat uživateli. V tomto případě je toto řešení realizováno pomocí šablonovacího systému Twig, který usnadňuje vytváření dynamických HTML šablon. Twig umožňuje vložení dat do HTML metodou speciální syntaxe, která například umožňuje uživatelům vidět na stránce aktuální počet správných a chybných odpovědí, případně poskytuje seznam řešených úkolů. Jsou zodpovědné za obsah a formát informací, zajišťují uživatelsky přívětivé prostředí. (11)

Controller funguje jako spojovací článek mezi modelem a view. Tato vrstva zpracovává HTTP požadavky od uživatele, volá potřebné metody v modelové vrstvě a získává data, která jsou následně předána do view. Controller obsahuje aplikační logiku, která rozhoduje o tom, jakým způsobem budou uživatelovy požadavky zpracovány a jaká data budou zobrazena. (11)

Průběh zpracování požadavku v aplikaci

- **Příjem požadavku:** Uživatel zadá požadavek (např. klikne na tlačítko pro nový matematický příklad). Požadavek se zachytí a předá příslušnému controlleru.
- **Zpracování v Controlleru:** Controller vyhodnotí typ požadavku a rozhodne, kterou část modelu volat. Například pro generování příkladu volá příslušnou modelovou třídu. Operace v Modelu: Modelová třída provede akci, načte data nebo generuje nový příklad podle parametrů. Zpracovaná data vrátí controlleru.
- **Předání dat do View:** Controller předá data do Twig šablony (view), která je přizpůsobí pro přehledné zobrazení uživateli. Zobrazení uživateli: Uživatel vidí výsledek, například nový příklad nebo statistiku úspěšnosti. View aktualizuje zobrazení na základě změn dat.



Obrázek 2: Architektura aplikace MVC [zdroj vlastní]

3.2 Nasazení aplikace pomocí Docker a CI/CD

Tato kapitola se věnuje využití softwaru Docker pro vytvoření přenositelného a konzistentního prostředí. Dále je v kapitole popsáno využití GitHub Actions (CI/CD pipeline), což zajišťuje automatizované nasazení aplikace.

Docker zabaluje aplikaci do kontejneru. Tu je pak možné spouštět bez ohledu na prostředí, protože aplikaci zabalí včetně její závislostí do virtuálního kontejneru, který je možné spustit na jakémkoliv operačním systému, který docker podporuje. Aplikace je rozdělena do dvou fází v rámci tzv. multi-stage Docker buildu.

Build fáze

Pro první fázi je využíván image php:8.3-cli, kde dochází k doinstalování potřebných nástrojů. Pro sestavení frontendu se jedná o zip, unzip, nodejs a npm. U backendu se jedná o composer, který zajistí správu PHP závislostí.

Po doinstalování nástrojů se zkopíruje zdrojový kód a vytvoří se konfigurační soubor .env, který obsahuje proměnné prostředí potřebné pro sestavení. Následně se pomocí composeru nainstalují PHP závislosti, bez balíčků, které jsou určeny pro vývoj (--no-dev).

V závěru první fáze jsou zkompilovány frontendové soubory (CSS a JavaScript) pomocí npm run build. Posledním krokem je odstranění nepotřebných souborů jako je složka node_modules, tím se sníží velikost výsledného Docker image.

Deploy fáze

Pro druhou fázi je použit image unit:1.33.0-php8.3, který obsahuje webový server NGINX Unit s podporou PHP 8.3. Do image se nainstalují PHP rozšíření opcache, pdo, pdo_mysql, simplexml a dom.

V závěru se zkopíruje hotová aplikace z první fáze a nastaví se oprávnění pro adresář var, který obsahuje logy a cache. Výsledkem je image, který obsahuje pouze soubory, které jsou potřebné pro běh aplikace v produkci.

Konfigurace aplikačního prostředí

Soubor php.ini obsahuje konfiguraci pro běh aplikace. OPcache zajišťuje přednačtení a optimalizaci PHP kódu. Realpath cache zlepšuje práci s cestami k souborům.

```
opcache.memory_consumption = 256
opcache.max_accelerated_files = 20000
opcache.preload ="/opt/app/config/preload.php"
opcache.preload_user = "unit"
opcache.validate_timestamps = 0
realpath_cache_size = 4096K
realpath_cache_ttl = 600
```

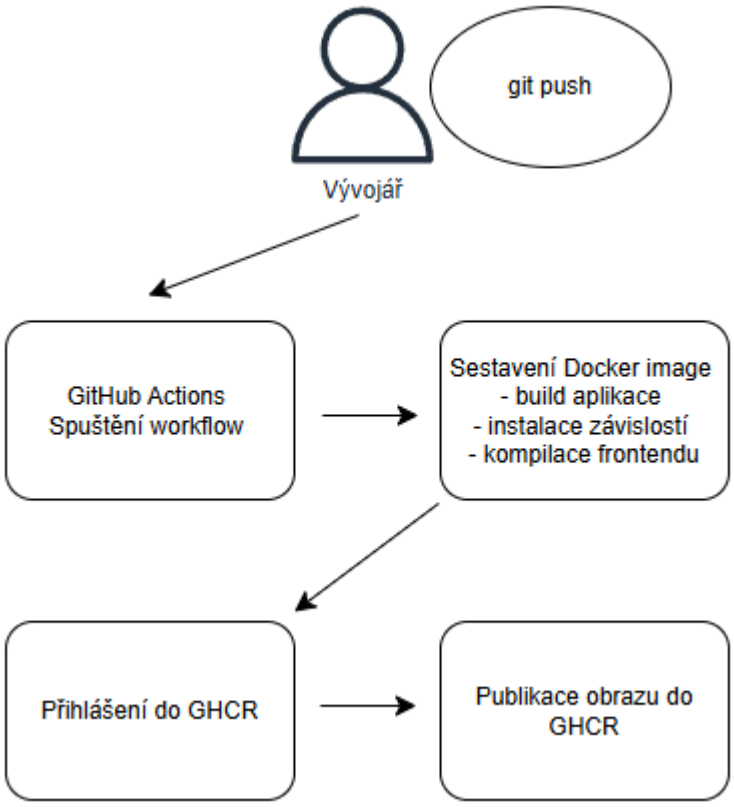
Výpis 1: Konfigurace pro běh aplikace [zdroj vlastní]

Automatické nasazení (CI/CD)

Automatické sestavení a publikování Docker image do git repositáře je zajištěno pomocí GitHub Actions, což je nástroj pro tzv. Continious Integration and Continues Delivery (CI/CD).

Při použití příkazu git push, tedy nahrání lokálního repositáře do vzdáleného repositáře se spustí workflow publish-docker-image.yaml, který provede následující kroky. Stáhne zdrojový kód a přihlásí se do GitHub Container Registry. Poté se staví Docker image a výsledný image nahraje do registru pod tagem latest.

Tento postup zajišťuje aktualizaci produkční verze aplikace při každé aktualizace vzdálené repositáře.



Obrázek 3: Automatické nasazení (CI/CD) [zdroj vlastní]

3.3 Databáze a práce s daty

Mapování entit

V aplikaci jsou entity reprezentovány PHP třídami s atributy odpovídajícími sloupcům v databázových tabulkách. Knihovna Doctrine se o entitách dozví díky mapovacím metadatům. Díky těmto metadatům je možné vytvořit jakousi konfiguraci, díky které Doctrine zjistí, jak mají být jednotlivé entity uloženy v databázi. (12)

Operace s databází

Doctrine ORM poskytuje funkcionalitu pro práci s databází prostřednictvím tzv. Entity Managera, což je hlavní nástroj pro provádění CRUD operací. To znamená vytvoření, čtení, aktualizaci a mazání dat. Entity Manager řídí životní cyklus entit. Doctrine nikdy nevyužívá metody v třídách entit jako jsou gettery, settery a ani konstruktory. Místo toho využívá reflexi, takže manipuluje přímo s atributy. Doctrine používá strategii “transactional write-behind“, což znamená, že zdrží většinu SQL příkazů, dokud se nezavolá metoda flush. (13)

Například zde metoda persist upozorní UnitOfWork na to, že při další zavolání metody flush se má provést operace na vložení userAttempts do databáze. Pokud by se na konci metoda flush

zavolala, tak se do databáze nic neuloží. UnitOfWork se implicitně spouští při vytvoření EntityManager, je možné je i manuálně zavřít zavoláním metody close, v takovém případě se žádné změny v databázi neprovedou. (13)

```
$newAttempt = new UserAttempts();
$newAttempt->setUser($user);
$newAttempt->setTheme($theme);
$newAttempt->setCorrectAnswers($correctCount);
$newAttempt->setIncorrectAnswers($incorrectCount);
$entityManager->persist($newAttempt);
$entityManager->flush();
```

Výpis 2: Ukázka operace s databází [zdroj vlastní]

Migrace

Aplikace používá knihovnu Doctrine Migrations pro správu změn v databázové struktuře. Tato knihovna umožňuje vytvářet migrace, tedy skripty se změnami ve struktuře databáze jako přidání tabulek, změny sloupců či úpravy vztahů mezi tabulkami. Každá migrace je ve třídě, která má dvě metody:

- **UP**: Určuje kroky pro zavedení nové změny (například přidání sloupce).
- **DOWN**: Umožňuje vrátit změny (například odstranění sloupce).

Tímto jsou změny sledovány a zachovány v historii migrací. To usnadňuje nasazení nových verzí a správu databázové struktury v různých prostředích. Doctrine Migrations zajišťuje konzistentní správu struktury a snižuje riziko chyb při manuálních úpravách databáze. (14)

Relační model

Databáze se skládá z šesti tabulek. Každá tabulka má specifickou funkci a obsahuje informace, které jsou vzájemně propojeny pomocí cizích klíčů. Následující části popisují jednotlivé tabulky a jejich význam v rámci celkového modelu.

Obrázek 4: Relační model [zdroj vlastní]

Popis tabulek

user

Uchovává veškeré údaje o uživatelích. Primární klíč tabulky je id. Vztah s tabulkami grade, user_statistics, user_attempts je realizován pomocí cizího klíče user_id.

theme

Tabulka, která představuje téma (např. “Lineární rovnice”). Obsahuje cizí klíč test_settings_id, který ukazuje na nastavení testu (test_settings). Tvoří vztahy s tabulkami example, block, grade, user_statistics, user_attempts pomocí cizího klíče theme_id.

test_settings

Nastavení pro testy ke každému tématu. Primární klíč je id. Tvoří vztah pouze s tabulkou theme, kde je pro referenci použito test_settings_id.

example

Jednotlivé příklady pro dané téma. Obsahuje vždy otázku a odpověď. Primární klíč je id. Vztah tvoří s tabulkou theme pomocí theme_id.

block

Obsahuje další bloky obsahu k danému tématu. Obsahuje pouze studijní text. Primární klíč je id a tvoří vztah s tabulkou theme pomocí theme_id.

grade

Známky uživatelů z testů. Jeden záznam obsahuje vždy známku a datum vytvoření. Primární klíč je id. Tvoří vztahy s user a theme pomocí user_id a theme_id.

user_statistics

Dlouhodobé statistiky uživatelů u témat. Obsahuje správné, špatné a všechny pokusy, level uživatele a poslední pokus. Primární klíč je id. Tvoří vztahy s tabulkami user a theme pomocí user_id a theme_id.

user_attempts

Záznam jednotlivých pokusů uživatele. Obsahuje správné a nesprávné odpovědi. Primární klíč je id. Tvoří vztahy s user a theme pomocí user_id a theme_id.

quotes

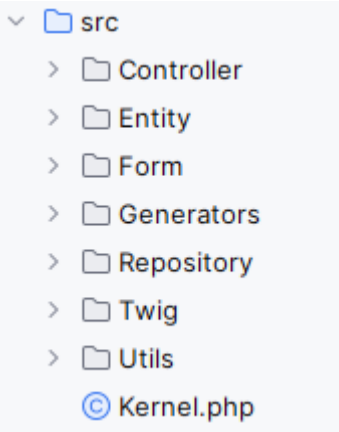
Citáty a motivační texty, které se zobrazují na hlavní stránce. Obsahuje citát a pozici, aby šlo citáty řadit. Tabulka netvoří žádné vztahy.

Shrnutí vztahů

- user má mnoho grades, user_statistics, user_attempts.
- theme má mnoho examples, blocks, grades, user_statistics, user_attempts.
- Každé theme má nastavení testu pomocí test_settings.

3.4 Backend

Backend aplikace je zorganizována ve složce src, která obsahuje komponenty pro správu a obsluhu požadavků uživatelů. Následující části podrobně popisují jednotlivé složky a jejich význam v rámci aplikace.



Obrázek 5: Backend struktura [zdroj vlastní]

Controller

Složka Controller obsahuje kontroléry, které obsahují logiku aplikace a zpracovávají HTTP požadavků. Controller třídy jsou prostředníkem mezi frontendem a backendem. Každý kontrolér má své specifické zaměření na konkrétní část aplikace a je zodpovědný za zpracování HTTP požadavků, komunikaci s databází pomocí repositářů, ověření uživatelů a přípravu dat k zobrazení v šablonách. (15)

Všechny kontroléry jsou děděny od třídy AbstractController, což umožňuje využívat funkce jako vykreslování šablon, nebo přesměrování. Mezi vlastnosti kontroléru také patří Dependency Injection, kde jsou potřebné služby, nebo repositáře automaticky poskytnuty kontroléru. Díky tomu Symfony dokáže rozpoznat co má kontroléru poskytnout, to umožňuje modularitu kódu a minimalizuje nutnost vytvářet nové instance uvnitř kontroléru.

Entity

Složka Entity obsahuje třídy entit, které zároveň reprezentují tabulky v databázi. Každá entita odpovídá konkrétní tabulce v databázi a obsahuje atributy, které odpovídají sloupcům této tabulky. Atributy Entit jsou anotovány pro mapování dat a definici vztahů mezi jednotlivými entitami. Tímto způsobem se zajišťuje, že data z databáze mohou být jednoduše mapována na objekty v aplikaci. (12)

```
#[ORM\Entity(repositoryClass: QuotesRepository::class)]
class Quotes
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    #[ORM\Column]
    private ?int $position = null;

    #[ORM\Column(length: 255)]
    private ?string $quote = null;
```

Výpis 3: Ukázka entity [zdroj vlastní]

Anotace definuje třídu Quotes jako Entitu pro Doctrine ORM, což zajistí, že bude uložena v databázi jako tabulka. Dále určuje, že pro práci s touto entitou bude používaná třída QuotesRepository, která zajišťuje metody pro manipulaci s daty v databázi. Na příkladu jsou použity tři typy anotací: anotace pro definici primárního klíče, pro sekvenční automatické generování hodnot a pro označení vlastností jako sloupců databázové tabulky. Třída dále obsahuje standardní metody typu getter a setter pro přístup a úpravu jednotlivých vlastností.

Form

Složka Form obsahuje třídy FormTypes, které definují formuláře používané v aplikaci. Tyto třídy určují strukturu a validaci formulářových polí, jako jsou vstupy pro uživatelská jména, hesla nebo jiná data. Pomocí těchto formulářových typů může aplikace snadno generovat HTML formuláře a spravovat uživatelské vstupy s potřebnou validací. (16)

```
public function buildForm(FormBuilderInterface $builder, array
$options): void
{
    $builder
        ->add('position', IntegerType::class, [
            'constraints' => [
                new NotBlank([
                    'message' => 'Pozice musí být vyplněna.',
                ]),
                new PositiveOrZero([
                    'message' => 'Pozice musí být nezáporné
číslo.',
                ]),
            ],
        ])
        ->add('quote', TextType::class, [
            'constraints' => [
                new NotBlank([
```

```
                    'message' => 'Citát musí být vyplněn.',
                ]),
                new Length([
                    'max' => 255,
                    'maxMessage' => 'Citát nesmí být delší než
255 znaků.',
                ]),
            ],
        ]);
}
```

Výpis 4: Ukázka metody build form [zdroj vlastní]

Metoda pro definování jednotlivých položek formuláře. Zde jsou definované dvě pole. Pole position s integer typem, která má dvě restrikce a to, že nesmí být prázdná a že musí být větší nebo rovno nule. Druhá položka je quote, která má typ text a nesmí být prázdná a má maximální možnou délku 255 znaků. U všech restrikcí jsou také zprávy, které uživatel uvidí, pokud je nesplní.

Generators

Složka Generators obsahuje pomocné třídy pro generování příkladů, které se používají v aplikaci pro vytváření matematických příkladů. Tyto třídy implementují logiku pro generování různých typů příkladů na základě definovaných parametrů, jako je obtížnost nebo téma. Tímto způsobem se uživatelům prezentují příklady, které mohou procvičovat. Každý generátor dědí, z abstraktní třídy AbstractGenerator, funkce, které jsou potřeba pro správnou funkčnost.

```
abstract class AbstractGenerator
{
    // Metoda pro generování příkladu
    abstract public function generate(int $minValue, int
$maxValue, int $numberOfExamples, string $difficulty): array;

    // Metoda pro ověření výsledku
    abstract public function verify($input,$correctResult):
bool;
}
```

Výpis 5: Ukázka AbstractGenerator [zdroj vlastní]

Repository

Složka Repository obsahuje třídy pro práci s entitami. Repositáře se používají k oddělení logiky pro přístup k datům od samotné logiky aplikace. Každý repositář obsahuje metody pro vykonávání operací nad entitami, jako je načítání, ukládání, aktualizace, odstraňování dat z databáze nebo také vlastní složitější funkce. (12)

Twig

Složka twig obsahuje třídu AppExtension, která rozšiřuje funkce Twig šablon. V rámci této aplikace byla implementována jedna funkce pro zajištění správného skloňování. Jedná se o filtr inflection, který na základě počtu vrací správný tvar slova. Filtr je využit na banneru hlavní stránky aplikace, kde se uživateli zobrazuje informace kolik má vytvořených statistik.

```
public function inflection(int $count, string $singular,
string $few, string $many): string
{
    if ($count === 1) {
        return $singular;
    } elseif ($count >= 2 && $count <= 4) {
        return $few;
    } else {
        return $many;
    }
}
```

Výpis 6: Ukázka rozšiřující twig funkce [zdroj vlastní]

```
{{ statsCount | inflection('statistiku', 'statistiky',
'statistik') }}
```

Výpis 7: Ukázka použití rozšiřující twig funkce [zdroj vlastní]

3.5 Frontend

Templates

Složka templates obsahuje Twig šablony, díky kterým jsou data zobrazována uživateli. Základní šablona base.html.twig obsahuje layout pro stránky aplikace. V této šabloně jsou definovány bloky, které jednotlivé stránky dědí a plní svým vlastním obsahem.

```
<div class="header">
    <div class="container">
        {% block header %}
            {% include 'Components/NavBar.html.twig' %}
        {% endblock %}
    </div>
</div>
```

Výpis 8: Ukázka definování bloku [zdroj vlastní]

Takto například vypadá hlavička v rodičovské šabloně. V adminu je například jiný navigační menu, a díky tomuto zápisu je možné menu změnit, ale zachovat stejné styly hlavičky.

Hlavní stránka administrace dědí z rodičovské šablony, přepisuje navigační menu, nadpis a má prázdné tělo a patičku.

```
{% extends 'base.html.twig' %}

{% block header %}
    {% include 'Components/AdminNavBar.html.twig' %}
{% endblock %}

{% block title %}Admin{% endblock %}

{% block body %}
    <div class="admin-main">
        {% include 'Components/AdminEntityCards.html.twig' %}
        {% include 'Components/AdminImportExamples.html.twig'
%}
    </div>
{% endblock %}

{% block footer %}{% endblock %}
```

Výpis 9: Ukázka předefinování bloku [zdroj vlastní]

Assets

Složka assets obsahuje další složky, styles a scripts.

Složka styles je určena pro stylování aplikace. Každá komponenta, případně stránka má vlastní soubor se styly, kde je tato část stylována. Jsou zde i tři globální soubory. Soubor variables definuje proměnné, které jsou pak používány v rámci jednotlivých komponent, převážně zde jsou definovány barvy, aby byla zajištěna jednotnost aplikace. Stejný princip má soubor btns, který definuje styly tlačítek, aby všechna tlačítka byla stejná a zajistila se jednotnost i zde. Poslední globální soubor je app, kde je definovaný základní layout, tedy styly pro hlavičku patičku, body, nebo kontejnery.

Složka scripts obsahuje JavaScript, nebo případně jQuery. Zde jsou skripty pro zpracovávání uživatelských interakcí různých částí aplikace.

Testování

Jednotkové testování je metoda pro kontrolu správnosti funkcí a metod aplikace, kdy se testují jednotlivé části kódu nezávisle na ostatních částech aplikace. Cílem těchto testů je ověřit, že funkce a metody jednotlivých tříd pracují přesně podle specifikací a že vracejí očekávané výsledky. V této části jsou popsány jednotkové testy pro třídu WordProblemGenerator, která v aplikaci slouží ke generování a řešení slovních úloh na různých úrovních obtížnosti (lehká, střední a obtížná). Třída také ověřuje, zda je zadaný uživatelský vstup správný.

Třída WordProblemGenerator obsahuje metody generate, solve a verify. Jednotkové testy byly vytvořeny s cílem ověřit správné fungování každé z těchto metod. Testovací skripty byly napsány pomocí knihovny PHPUnit.

První skupina testů ověřuje metodu generate. Test testGenerateEasyDifficulty například zajišťuje, že generátor správně vytvoří zadaný počet slovních úloh lehkou obtížnost. Každá úloha musí obsahovat atributy equation (slovní zadání úlohy) a format. Test kontroluje nejen, že generátor vrátí správný počet úloh, ale i to, že každá úloha obsahuje správné atributy.

Druhá skupina testů ověřuje metodu solve, která řeší slovní úlohy. Každý test této skupiny kontroluje konkrétní typ úlohy, například test testSolveAdditionProblem ověřuje, že metoda solve správně vyřeší úlohu na sčítání, zatímco testSolveDivisionProblem kontroluje úlohu na dělení. Test solve vrací nejen řešení úlohy, ale i postup, jak k němu dospět. Každý test také kontroluje, že kroky vedoucí k řešení jsou správně zapsány, například pro úlohu „Na farmě bylo 5 jablek. Poté farmář přidal 3 jablka. Kolik jablek má nyní?“ test kontroluje, že řešení je 8 a že krok obsahuje text „Vypočítáme: 5 + 3 = 8“.

```
public function testSolveAdditionProblem()
{
    $generator = new WordProblemGenerator();
    $equation = "Na farmě bylo 5 jablek. Poté farmář přidal 3 
jablek. Kolik jablek má nyní?";
    $result = $generator->solve($equation);

    $this->assertEquals(8, $result['solution']);
    $this->assertStringContainsString("Vypočítáme: 5 + 3 = 8", 
$result['steps'][0]);
}
```

Výpis 10: Ukázka testovací metody [zdroj vlastní]

Poslední část testů je zaměřena na metodu verify, která ověřuje uživatelský vstup. Tato metoda porovnává správný výsledek s odpovědí uživatele a při kontrole umožňuje malou toleranci kvůli možným zaokrouhlovacím chybám. Test testVerifyCorrectAnswer kontroluje, že při odpovědi s malou odchylkou od správného výsledku (například o 0.01) metoda výsledek uzná, zatímco testVerifyIncorrectAnswer testuje, že pokud je odpověď mimo toleranci, bude výsledek vyhodnocen jako nesprávný.

4 Zabezpečení

V této části se čtenář dozví mechanismy, které byly implementovány k ochraně aplikace proti bezpečnostním hrozbám.

Autentizace

Autentizace je proces, který určuje, zda uživatel, který používá aplikace je tím, za koho se vydává. Tento proces je zajištěn přihlašovacím a registračním formulářem. Každý uživatel musí být registrovaný se jménem, příjmením, jedinečným prvkem, který je v této aplikaci e-mail, protože dva uživatelé nemohou mít stejný e-mail, posledním prvkem je heslo, které se do databáze vkládá v zašifrované podobě. Aplikace pro implementaci autentizace využívá symfony balíček "symfony/security-bundle". (17)

Autorizace

Autorizace je implementována pomocí RBAC. Každý uživatel má přiřazenou roli, kde každá role má určitá práva. Například role ADMIN může využívat veškeré funkce aplikace, ale USER má omezená práva a nemůže využívat administraci. Na základě rolí může být odepřen i přístup k určitým stránkám, a to je řešený buďto pomocí anotací v kontrolérech anebo v konfiguračním souboru security.yaml. (17)

```
access_control:
     - { path: ^/admin, roles: ROLE_ADMIN }
```

Výpis 11: Omezení přístupnosti podle role [zdroj vlastní]

CSRF ochrana

CSRF je útok, který přiměje oběť k odeslání škodlivého požadavku. Zdědí identitu a oprávnění oběti a provede jejím jménem nežádoucí funkci. Tyto útoky se zaměřují na funkce, které způsobí změnu stavu na serveru, například změnu e-mailové adresy, hesla oběti, nebo dokonce nákup. Toto je příklad, který může nastat u přihlášeného uživatele, protože zde útočník nedokáže získat osobní data oběti. Pokud je ale uživatel nepřihlášen, může ho útočník donutit, aby se přihlásil k účtu, který útočník ovládá. Oběť může k účtu přidat osobní údaje a útočník se pak k účtu znovu přihlásit a zobrazit tyto údaje.

Ochrana proti CSRF je bezpečnostním prvkem, který zajišťuje, že formuláře jsou zasílány pouze z důvěryhodných zdrojů. Symfony umožňuje generovat CSRF tokeny pro všechny formuláře. Tyto tokeny jsou pak ověřovaný při odeslání formuláře na server, což zajišťuje, že žádný podvodný skript nemůže odeslat podvodný požadavek. Pro implementace CSRF ochrany je v aplikaci použít CsrfTokenManager, který spravuje vytváření a ověřování tokenů. Tokeny jsou generovány při vykreslení formuláře a ověřovány na straně serveru. (17)

Ochrana proti XSS

Útoky XSS jsou typem injektáže, při níž jsou do jinak neškodných a důvěryhodných webových stránek injektovány škodlivé skripty. K útokům XSS dochází, když útočník použije webovou aplikaci k odeslání škodlivého kódu, obvykle ve formě skriptu na straně prohlížeče, jinému koncovému uživateli. Útočník může pomocí XSS odeslat nic netušícímu uživateli škodlivý skript. Prohlížeč koncového uživatele nemá možnost poznat, že skript nemá být důvěryhodný, a skript spustí. Protože si myslí, že skript pochází z důvěryhodného zdroje, může škodlivý skript získat přístup ke všem souborům cookie, tokenům relace nebo jiným citlivým informacím uchovávaným prohlížečem a používaným s daným webem. Tyto skripty mohou dokonce přepsat obsah stránky HTML. (18)

Ochrana proti XSS útokům je zajištěna automatickým ošetřením vstupních dat. Tato ochrana je implementována v rámci Twig šablon, které automaticky zabezpečují všechny proměnné proti nechtěnému spouštění kódu, čímž zabraňují injektování škodlivých skriptů. K tomu dochází při vykreslování proměnných v šablonách. (18)

Hashování hesel

Hesla uživatelů jsou uložena v databázi v zašifrované podobě, což zajišťuje jejich bezpečnost. Hashovaní hesel je implementováno pomocí třídy UserPasswordHasherInterface, která je součástí PHP Symfony. Funkce pro hashovaní hesel využívá algoritmus bcrypt, který vytváří hesla dlouhá 60 znaků. Hesla také obsahují kryptografickou sůl, která se vygeneruje automaticky pro každé nové heslo. (17)

SQL injection

Ochranu pro SQL injection útokům zajišťuje Doctrine ORM a Query Buidler, které automaticky parametrizují SQL dotazy. Použitím Doctrine ORM pro manipulaci s entitami se v aplikace vyhýbá přímému psaní SQL dotazů, což snižuje riziko injekce. Doctrine ORM zajišťuje, že všechny dotazy jsou správně parametrizovány a že uživatelské vstupy jsou neutralizovány pro použití v SQL, čímž se minimalizuje riziko útoků. Pro složitější dotazy je využíván Doctrine Query Builder, který taktéž zabezpečuje parametrizaci a zajišťuje, že uživatelský vstup je bezpečně zpracován a nelze ho zneužít k provedení neautorizovaných SQL dotazů. (19)

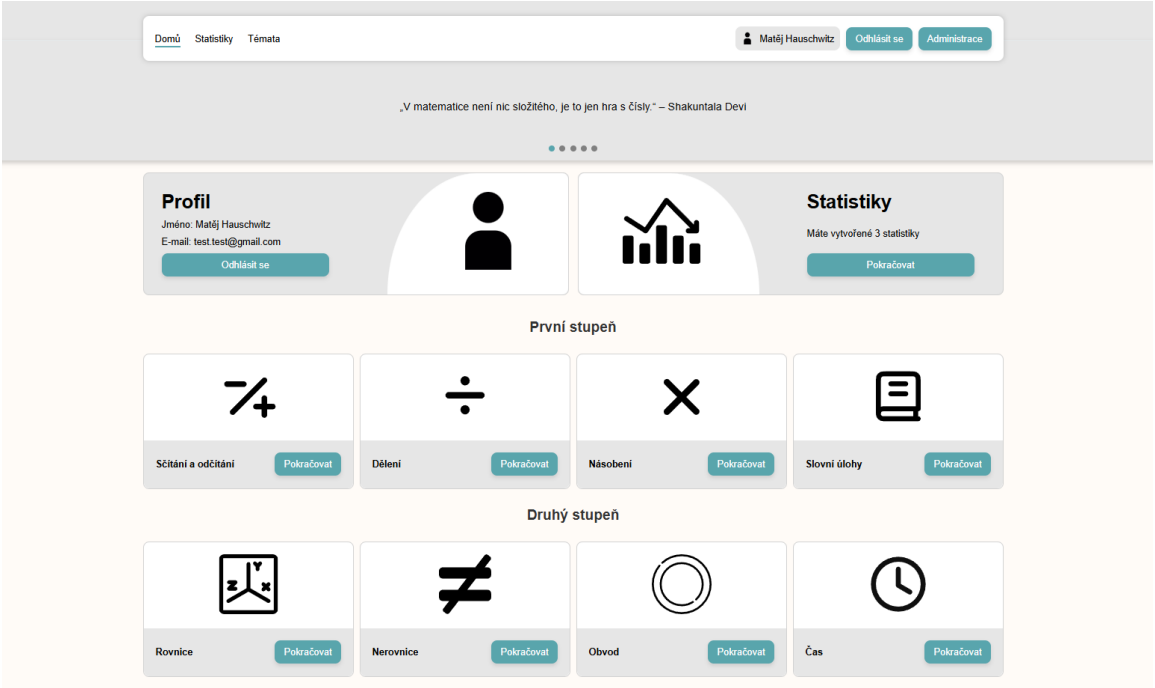
```
return $this->createQueryBuilder('s')
    ->select('u.firstName', 'u.lastName',
        's.correctAnswers', 's.totalAttempts',
's.incorrectAnswers')
    ->join('s.user', 'u')
    ->where('s.theme = :themeId')
    ->setParameter('themeId', $themeId)
    ->orderBy('(s.correctAnswers * 1.0) /
NULLIF(s.totalAttempts, 0)', 'DESC')
    ->setMaxResults(10)
    ->getQuery()
    ->getResult();
```

Výpis 12: Ukázka Query Builder [zdroj vlastní]

5 Aplikace

Hlavní stránka

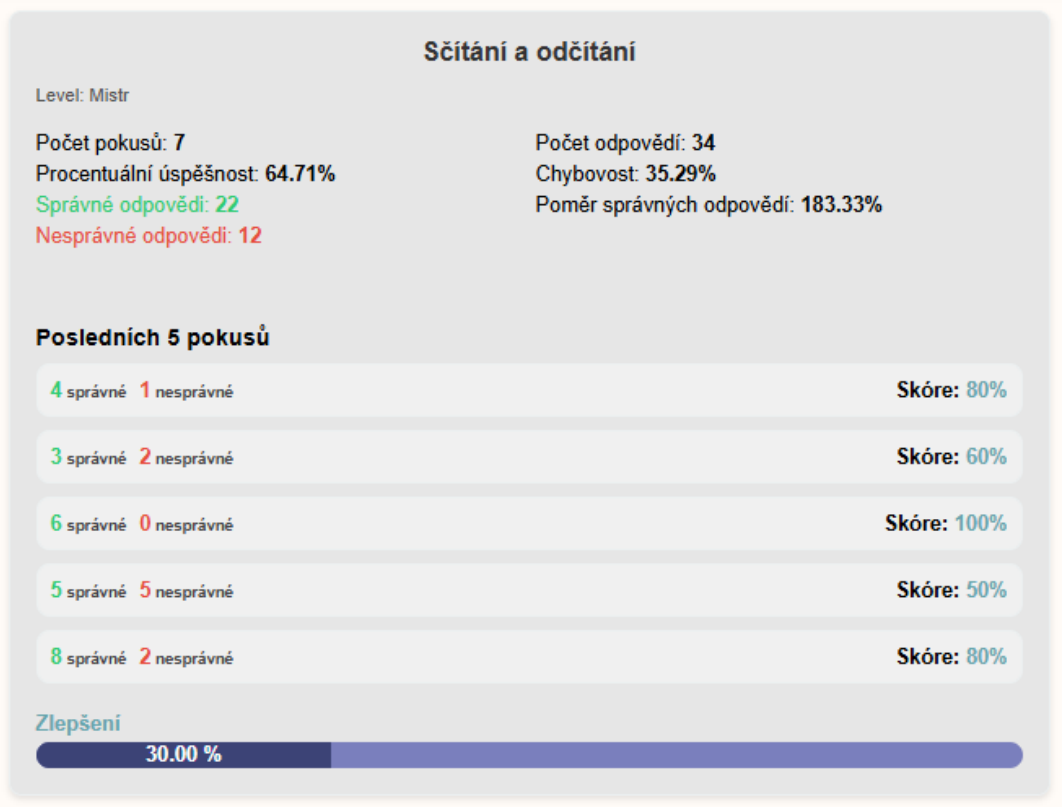
Hlavní stránka funguje jako takový rozcestník do dalších částí aplikace. Stránka obsahuje dva bannery. Banner profil ukazuje informace o uživateli a tlačítko, které uživatele odhlásí. Pokud stránku navštívil nepřihlášený uživatel, tak se na banneru zobrazí pouze tlačítko pro přihlášení. Druhý banner zobrazuje informaci o statistikách, konkrétně kolik má uživatel dostupných statistik. Pokud uživatel není přihlášen, tak se na banneru opět zobrazí pouze tlačítko pro přihlášení. V další části stránky se zobrazují dlaždice všech témat, ze kterých se uživatel může dostat na detail tématu, zde nezáleží na tom, jestli je uživatel přihlášený, nebo není. Zároveň se na hlavní stránce zobrazují citáty a navigační menu.



Obrázek 6: Hlavní stránka aplikace [zdroj vlastní]

Statistiky

Tato funkcionalita je dostupná pouze pro registrované uživatele, pokud by se chtěl neregistrovaný uživatel dostat na stránku statistik, tak bude přesměrován do přihlašovacího formuláře. Statistiky se počítají pouze z testů, progres v rámci procvičování není zaznamenáván. Ve statistikách se zobrazují informace z tabulek user_attempts a user_statistics. Úroveň uživatele je určena na základě procentuální úspěšnosti. Aplikace obsahuje 8 úrovní nejnižší úroveň je do 10 % (Nováček) a nejvyšší úroveň je nad 80 % (grandmister). Z tabulky user_statistics se načítají správné odpovědi, nesprávné odpovědi a počet odpovědí, ostatní statistiky jsou počítány v kontroléru. Poslední pokusy jsou uloženy v tabulce user_attempts, kde se uživateli ukládá ke každému tématu maximálně 5 pokusů, pokud je limit překročen, tak se nejstarší pokus přepíše. Spodní ukazatel zobrazuje zlepšení vůči poslednímu pokusu.



Obrázek 7: Statistika uživatele [zdroj vlastní]

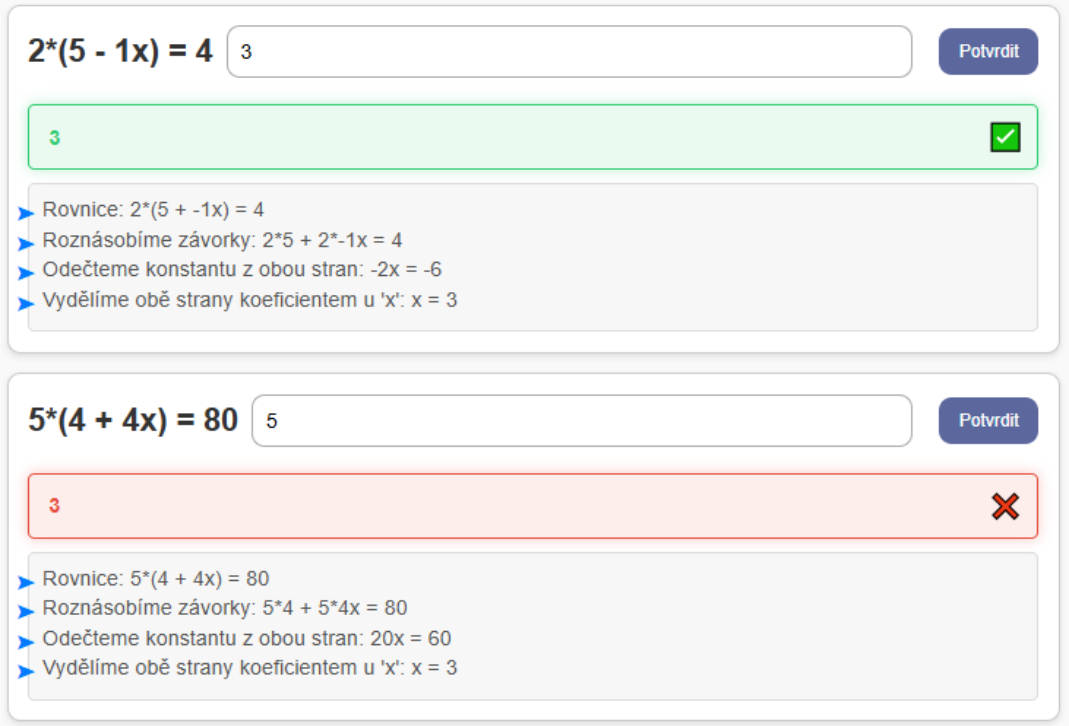
Detail Tématu

Detail tématu funguje jako takový mezikrok pro zapnutí procvičování, nebo testu. Uživatel se v této části seznámí s problematikou určitého tématu, kde je v rámci studijního textu popsané, o jaké téma se jedná a jakým způsobem se počítá. Na této stránce se objevují čtyři tlačítka, zpět, procvičování, test, žebříček. Tlačítko zpět vrátí uživatele na předchozí stránku, tedy na hlavní stránku aplikace. Tlačítko procvičování přesměruje uživatele na vstupní formulář, kde zadá parametry pro vygenerování příkladů. Toto tlačítko je přístupné uživateli pouze v případě, že je implementován potřebný generátor, v opačném případě tlačítko není možné použít a uživateli po pokusu o kliknutí vyskočí hláška, že procvičování není dostupné. Tlačítko test je možné použít pouze v případě, že je uživatel registrovaný a pokud dané téma obsahuje alespoň 5 testovacích příkladů. V opačném případě bude opět znemožněno tlačítko použít a uživateli po pokusu o kliknutí vyskočí upozornění.

Obrázek 8: Stránka tématu [zdroj vlastní]

Procvičování

Procvičování je část, kde se generují náhodné příklady na základě vstupního formuláře. Vstupní formulář obsahuje minimální a maximální generované hodnoty, počet příkladů a obtížnost. Poté co se vygenerují příklady může uživatel zadávat výsledky do vstupních polí, které jsou u jednotlivých příkladů. Po kliknutí na tlačítko „Potvrdit“ se uživateli zobrazí správný výsledek a zároveň postup, který k tomuto výsledku vedl. Pokud uživatel zadá výsledek špatný, tak se správný postup i výsledek zobrazí také, ale zobrazí se červeně zabarven. Pokud již uživatel jednou zadal výsledek, může ho zadat znovu, není tedy omezený počet pokusů na spočítání příkladu. Uživatel zde také může sledovat kolikrát správně, či špatně odpověděl. Tato statistika se ukládá v lokálním uložišti prohlížeče a pokud dojde k znovu načtení příkladů, nebo přesměrování na jinou stránku, dojde k promazání těchto informací. Pokud uživatel spočítá všechny příklady, tak si může vygenerovat znovu příklady se stejnými parametry, nebo se může vrátit na vstupní formuláře a parametry si změnit.

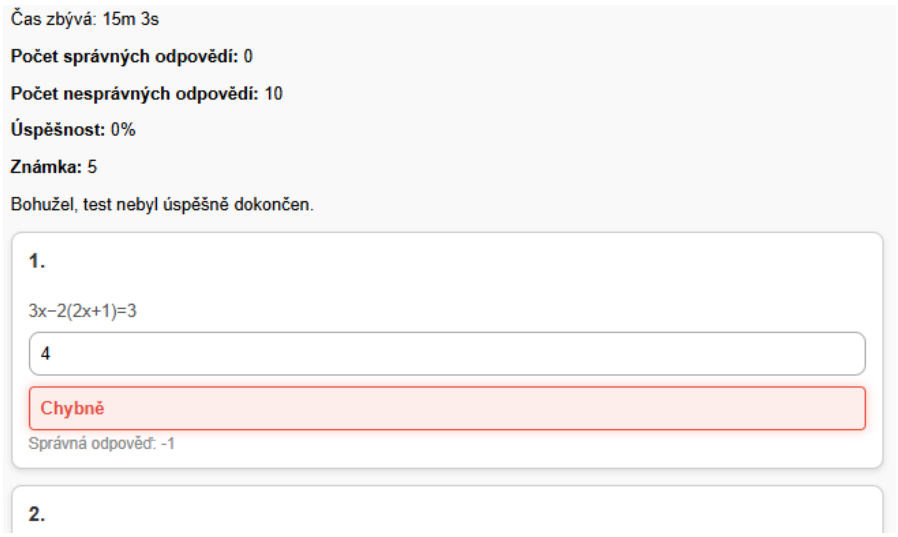


Obrázek 9: Procvičování [zdroj vlastní]

Test

Testy zobrazují příklady uložené v databázi. V administraci je možné u každého tématu najít i nastavení testu. Kde je možné jeho chování upravovat. V nastavení je možné určit dobu trvání testu, počet otázek, náhodné řazení otázek, zobrazení správných odpovědí po dokončení testu, jestli se z testu mají vytvořit známky, heslo a procentuální úspěšnosti k jednotlivým známkám.

Při nastavení známkovaného testu se po dokončení vytvoří známka. Veškeré známky je možné najít v administraci u uživatele. Známky je pak možné přepisovat, nebo mazat.



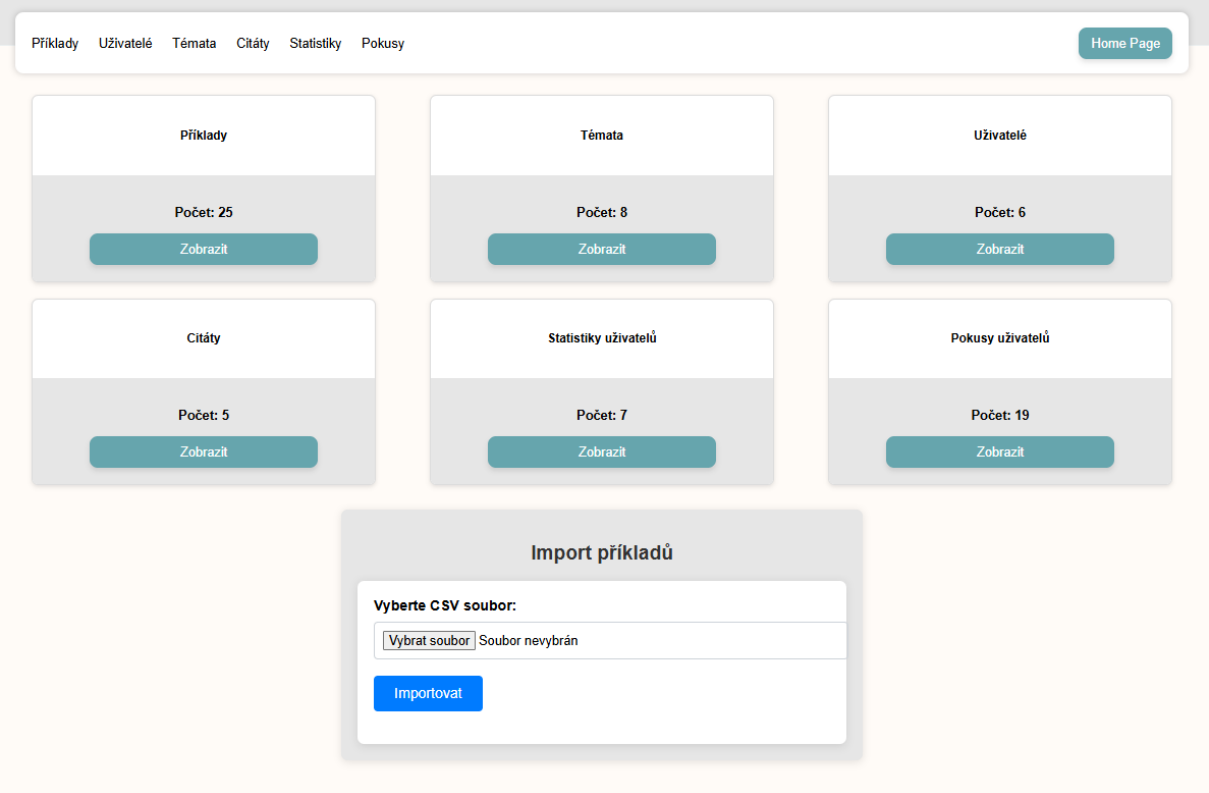
Obrázek 10: Test [zdroj vlastní]

Žebříček

Každé téma má žebříček, který ukazuje, jaký uživatel si v daném tématu vede nejlépe. V žebříčku se ukazuje 10 nejlepších uživatelů, kde skóre se počítá z tabulky UserStatistics vydělením správných odpovědí počtem celkových odpovědí.

Obrázek 11: Žebříček tématu [zdroj vlastní]

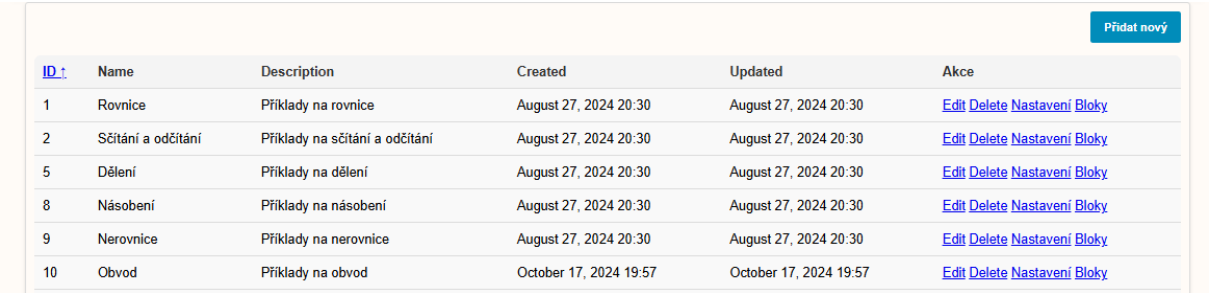
Administrace



Obrázek 12: Administrace hlavní stránka [zdroj vlastní]

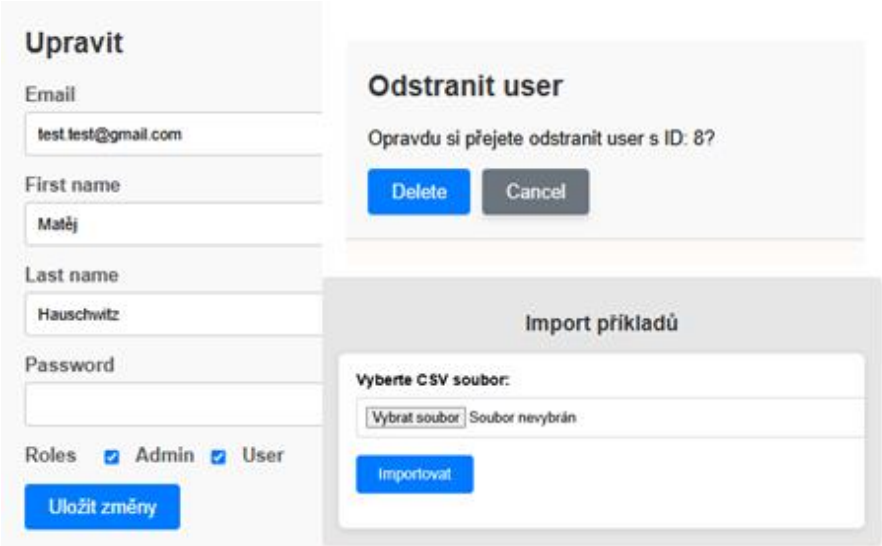
Administrace je část aplikace, která je určena pouze pro administrátory. Pokud uživatel nemá požadovanou roli, tak je přesměrován na přihlašovací formulář. Administrátor zde může

provádět CRUD operace na všech entitách. Při prvním zobrazení listu entity je list seřazen podle ID a pak se dá případně řadit podle cizích klíčů, nebo pozice.



Obrázek 13: Administrace list [zdroj vlastní]

Pro editaci, přidání a mazání aplikace využívá dialogová okna. Tato okna jsou vytvořena jako nová záložka, díky tomu uživatel nemusí interagovat s nimi a mohou se vrátit k hlavnímu oknu. Pro přidání nové entity se zobrazí v dialogovém okně formulář pro vyplnění potřebných informací. Po přidání nové entity se formulář změní na editační a znovu se načte rodičovské okno, aby uživatel mohl zkontrolovat, že se entita přidala. Pro odebrání entity se zobrazí dialogové okno s potvrzením a id entity, kterou se uživatel snaží odstranit. Pokud má entita vazbu na jinou entitu, tak dojde k odstranění i entit, které k ní patří. Tedy například při odstranění uživatele dojde i k odstranění statistik a posledních pokusů, protože by pak tyto entity bez uživatele v aplikaci neměly žádný význam. V rámci administrace je také možné importovat příklady pomocí csv souborů.



Obrázek 14: Dialogová okna administrace [zdroj vlastní]

6 Možnosti dalšího rozšíření aplikace

Aplikace v aktuálním stavu může sloužit jako prostředí pro vzdělávání, procvičování a testování. Je spousta možností, jakým aplikace ještě rozšířit a zvýšit tím atraktivitu jak pro studenty, tak i pro pedagogy základních škol.

Zajímavé rozšíření by mohlo být přidání herních prvků do aplikace. Pomocí jednoduchých vzdělávacích her, sbírání bodů, odemykání úrovní, nebo třeba vizuálních animací by bylo možné zvýšit motivaci žáků a podpořit jejich zájem o vzdělávání v matematice. Hry by mohly být navrženy tak, aby procvičovaly konkrétní dovednosti hravou formou a podporovaly soutěživost i spolupráci mezi žáky.

Pro pedagogy by mohlo být skvělé rozšíření o zadávání domácích úkolů. Tedy vytvořit domácí úkol a umožnit žákům domácí úkol odevzdat. Případně vytvořit úkol, který by říkal, že žák musí splnit určité cvičení, nebo hru na zvolenou úroveň. Po splnění úkolu, by učitel mohl napsat hodnocení.

Momentálně aplikace umožňuje přidávat příklady do testů, ale dalším rozšířením by mohlo být vytvářet více testů pro téma. Umožnilo by to lehčí a přívětivější možnost vytvářet testy podle aktuálních potřeb.

Zajímavou možností je také synchronizace s dalšími nástroji. Například pro správu známek. Díky tomu, že se jedná zejména o desktopovou aplikace, tak by se známky posílali do jiného nástroje a studenti, případně učitelé, by mohli známky rovnou kontrolovat pomocí telefonu.

Z pohledu výuky by mohlo být přínosné také přidání interaktivních výukových materiálů. Například krátká videa s vysvětlením tématu, postupy řešení nebo tipy a triky, jak přistupovat k danému typu příkladů.

Velká nevýhoda aplikace jsou generátory příkladů, které se musejí implementovat manuálně a není možné je nijak nastavit v administraci. To by mohlo vylepšit implementace umělé inteligence, která by tyto generátory zastoupila.

7 ZÁVĚR

Cílem této bakalářské práce bylo vytvořit webovou aplikaci pro výuku matematiky a tento cíl se povedlo úspěšně splnit. Výsledná aplikace umožňuje generování matematických příkladů různých typů a obtížností, jejich vyhodnocení a sledování úspěšnosti uživatele. I když je spoustu dalších možností, jak aplikaci vylepšit, tak nabízí prostředí pro vzdělávání, procvičování a testování žáků základních škol.

Tato práce pro mě měla velký přínos zejména po stránce praktických zkušeností. Měl jsem možnost si vyzkoušet návrh i realizaci webového projektu od začátku až do konce – od analýzy požadavků přes implementaci backendu a frontendového rozhraní až po testování. Díky tomu jsem se výrazně zdokonalil v práci s webovými technologiemi, objektově orientovaným programováním, návrhovými vzory i používáním frameworku Symfony. Získané znalosti a dovednosti jsou pro mě velmi cenné nejen v rámci studia, ale i do budoucna v praxi.

Vývoj aplikace mě bavil a motivoval k hledání nových řešení i zlepšování vlastního kódu. Věřím, že pokud by se aplikace dále rozšířila o navržené funkce, jako jsou herní prvky, nástroje pro učitele nebo implementace umělé inteligence, mohla by najít své uplatnění i v reálné výuce. V aktuálním stavu má aplikace spíše prototypový charakter a pro praktické použití by bylo vhodnější využít některou z dostupných profesionálních platforem. Přesto může sloužit jako důkaz technické proveditelnosti a jako základ pro další vývoj.

POUŽITÁ LITERATURA

1. **ŽÁRA, Ondřej.** *JavaScript: programátorské techniky a webové technologie.* Brno : Computer Press, 2015. ISBN: 978-80-251-4573-9.

2. **Tran, Tania Rascia a Tony.** An Introduction to jQuery. *DigitalOcean Community.* [Online] 1. srpen 2022. [Citace: 25. Duben 2025.] Dostupné z: https://www.digitalocean.com/community/tutorials/an-introduction-to-jquery.

3. **MICHÁLEK, Martin.** *CSS: moderní layout.* Praha : Martin Michálek - Vzhůru dolů, 2022. ISBN: 978-80-88253-07-5..

4. **SYMFONY.** Twig Documentation. *Symfony.* [Online] 2023. [Citace: 4. Duben 2025.] Dostupné z: https://twig.symfony.com/doc/3.x/.

5. **GeeksforGeeks.** PHP Introduction. *GeeksforGeeks.* [Online] 21. Duben 2025. [Citace: 25. Duben 2025.] Dostupné z: https://www.geeksforgeeks.org/php-introduction/.

6. **Symfony.** Introducing. *Symfony.* [Online] [Citace: 25. Duben 2025.] Dostupné z: https://symfony.com/doc/current/create_framework/introduction.html.

7. **Simplilearn.** Simplilearn. *What is SQLite? Everything You Need to Know.* [Online] 7. Červen 2024. [Citace: 25. Duben 2025.] Dostupné z: https://www.simplilearn.com/tutorials/sql-tutorial/what-is-sqlite.

8. **Docker.** Docker. *Docker Overview.* [Online] [Citace: 25. Duben 2025.] Dostupné z: https://docs.docker.com/get-started/docker-overview/.

9. **GitHub.** GitHub. *About GitHub and Git.* [Online] [Citace: 25. Duben 2025.] https://docs.github.com/en/get-started/start-your-journey/about-github-and-git.

10. **Codefresh.** Codefresh. *GitLab CI: Feature Overview, Tutorial and Best Practices.* [Online] [Citace: 25. Duben 2025.] Dostupné z: https://codefresh.io/learn/gitlab-ci/.

11. **TutorialsPoint.** TutorialsPoint. *Symfony Architecture.* [Online] [Citace: 25. Duben 2025.] https://www.tutorialspoint.com/symfony/symfony_architecture.htm.

12. **Symfony.** Doctrine and the Database (Symfony Docs). *Symfony Documentation.* [Online] [Citace: 27. Duben 2025.] Dostupné z: https://symfony.com/doc/current/doctrine.html.

13. **Project, Doctrine.** Doctrine. *Getting Started with Doctrine ORM.* [Online] [Citace: 25. Duben 2025.] Dostupné z: https://www.doctrine-project.org/projects/doctrine-orm/en/3.3/tutorials/getting-started.html.

14. **Pınarbaşı, Enes Selim.** Teknasyon Engineering. *Doctrine Migration: A Powerful Tool for Managing Database Schemas.* [Online] 22. Říjen 2024. [Citace: 25. Duben 2025.] https://engineering.teknasyon.com/doctrine-migration-bd6bd4931417.

15. **Symfony.** Symfony. *Symfony Controller.* [Online] [Citace: 25. Duben 2025.] https://symfony.com/doc/current/controller.html.

16. —. Forms (Symfony Docs). *Symfony Documentation.* [Online] Symfony. [Citace: 27. Duben 2005.] https://symfony.com/doc/current/forms.html.

17. —. Security (Symfony Docs). *Symfony Documentation.* [Online] [Citace: 27. Duben 2025.] https://symfony.com/doc/current/security.html.

18. —. Templates (Symfony Docs). *Symfony Documentation.* [Online] [Citace: 27. Duben 2025.] https://symfony.com/doc/current/templates.html.

19. **Project, Doctrine.** Query Builder — Doctrine DBAL 4.2 documentation. *Doctrine Project.* [Online] [Citace: 27. Duben 2025.]