DELTA – Střední škola informatiky a ekonomie, s.r.o.

Ke Kamenci 151, Pardubice

Systém pro procvičování gramatiky českého jazyka

Příjmení, jméno: Svatoň Jindřich

Třída: 4.A

Studijní obor: Informační technologie - denní studium

Školní rok: 2020 - 2024

Zadání maturitního projektu z informatických předmětů

Jméno a příjmení:	Jindřich Svatoň
Školní rok:	2023/2024
Třída:	4.A
Obor:	Informační technologie 18-20-M/01

Téma práce:	Systém na procvičování gramatiky
Vedoucí práce:	RNDr. Jan Koupil, Ph.D.

Způsob zpracování, cíle práce, pokyny k obsahu a rozsahu práce:

Cílem práce je vytvořit jednoduchý informační systém s anonymním i neanonymním přístupem sloužící k procvičování gramatiky českého jazyka.

V systému bude implementováno několik základních druhů cvičení (doplňovačka, diktát, hledání chyb), bude ale navržen tak, aby přidávání dalších druhů cvičení bylo velmi jednoduché.

Cvičení, která učitel vytvoří, budou označena tagy(štítky) a budou také nabízet konfiguraci podmínek, za kterých se odkryje řešení. Pro potřeby znovupoužití je bude také možné exportovat a importovat, resp. sdílet mezi jednotlivými uživateli.

Součástí projektu bude systém uživatelských oprávnění na úrovních Host (nepřihlášený uživatel) / Uživatel / Učitel / Administrátor

Vyhodnocování cvičení bude probíhat automaticky, přičemž testovaný uvidí svoji procentuální úspěšnost a chyby, kterých se při vyplňování dopustil. Hodnocení cvičení bude pro každého přihlášeného uživatele po určitou dobu archivováno.

Stručný časový harmonogram (s daty a konkretizovanými úkoly):

- **Září:** Analýza problému, návrhu funkcionalit systému a výběr technologií
- **Říjen:** Návrh architektury a databázového modelu
- **Listopad-leden:** Tvorba aplikace
- **Únor:** Testování aplikace a tvorba dokumentace
- **Březen:** Dokončení dokumentace

Prohlašuji, že jsem maturitní projekt vypracoval samostatně, výhradně s použitím uvedené literatury.

V Pardubicích 30. 3. 2024

Rád bych poděkoval RNDr. Janu Koupilovi, Ph.D. za vedení projektu, udržování projektu v rozumných mezích, připomínání toho, co je důležité, nápady a trpělivost. Dále bych rád poděkoval své rodině za trpělivost a podporu.

Anotace

Tato práce se zabývá vývojem jednoduchého informačního systému s anonymním i neanonymním přístupem pro procvičování gramatiky českého jazyka pomocí samovyhodnocovacích pracovních listů.

Klíčová slova

PHP, Laravel, React, ČJ, samovyhodnocovací cvičení, samovyhodnocovací pracovní listy, XML, IS, web

Anotation

This thesis deals with the development of a simple information system with anonymous and non-anonymous access for practicing Czech grammar using self-assessment worksheets.

Keywords

PHP, Laravel, React, Czech, self-assessment exercises, self-assessment worksheets, XML, IS, web

Obsah

Úvod **8**

2 Rešerše **9**

3 Architektura **11**
3.1 Frontend 11
3.2 Backend 12
3.2.1 Validace 12

4 Autentifikace **13**

5 Úloha **15**
5.1 Atributy úlohy 15
5.2 Zdrojový kód 15
5.2.1 Části 15
5.3 Cvičení 16
5.3.1 Typy cvičení 16
5.4 Syntaxe 18
5.4.1 Úloha 18
5.5 Životní cyklus úlohy 19

6 Databáze **21**
6.1 Tabulka tasks 21
6.1.1 Atributy 22
6.2 Tabulka task_reviews 22
6.2.1 Atributy 22
6.3 Tabulka task_review_templates 23
6.3.1 Atributy 23
6.4 Tabulka task_infos 23
6.5 Tabulka task_sources 23
6.6 Tabulka groups 24
6.6.1 Atributy 24
6.7 Tabulka exercises 24
6.7.1 Atributy 24

7 Výsledky **25**

8 Závěr **26**

Seznam obrázků **29**

9 Seznam příloh **30**

Use case diagram 1 **I**

Dokumentace API 2 **II**

1.1 Reference II

1.2 Základní struktura požadavků a úspěšných odpovědí II

1.3 Základní struktura chyb II

1.4 Chybové kódy III

1.5 Koncové body III

1.5.1 Učitel III

1.5.2 Přihlášený uživatel IV

1.5.3 Nepřihlášený uživatel V

Úvod

Každá dovednost se musí neustále procvičovat, to platí i pro gramatiku českého jazyka.

K procvičování gramatiky se běžně využívají pracovní listy, ty se však musí vytisknout a i v případě elektronických listů se musí zkontrolovat.

Domnívám se, že řešením jsou elektronické samovyhodnocovací pracovní listy a cvičení, jež tím, že se vyhodnocují samy, zjednodušují a zrychlují celý proces procvičování, což jej činí přístupnějším i pro lidi, jež by nutnost vlastní kontroly odradila.

Moje webová stránka se zaměřuje na poskytnutí jednoduchého rozhraní pro správu a vyplňování samovyhodnocovacích pracovních listů zaměřených na gramatiku českého jazyka.

2 Rešerše

Existuje spousta webů nabízejících procvičování gramatiky českého jazyka formou samovyhodnocovacích cvičení a pracovních listů. Některé se na již zmíněnou oblast specializují, u jiných jde pouze o jedno z široké škály možných použití.

V této kapitole představím tři z nich a provedu jejich srovnání s mojí aplikací.

1. Pravopisně

   Prvním řešením, o němž zde budu hovořit, je web Pravopisně, jež se mimo jiné specializuje i na vysvětlování pravidel českého jazyka a nejčastějších pravopisných chyb. Nabízí spoustu interaktivních cvičení pro procvičování češtiny.

   Pomocí nástroje Pravopisně+ umožňuje rychlé a intuitivní vytváření pravopisných cvičení, v nichž se se na vynechaná místa v textu vybírá správný znak z nabízené sady možností. Většina těchto cvičení se vyhodnocuje v průběhu jejich vyplňování.

   Vytváření těchto pravopisných cvičení je na opravdu vysoké úrovni, lze si vybrat různé pravopisné jevy a jejich obtížnost. Program sám vygeneruje cvičení spolu s řešením, vysvětlujícím vybrané pravopisné jevy.

   Tento generátor nabízí 14 denní bezplatnou zkušební verzi.

2. Umíme česky

   Druhým webem poskytujícím již zmíněnou formu procvičování je Umíme česky, jež je součástí série webů zaměřených na procvičování školních předmětů od společnosti Umíme to, s.r.o.

   Umíme česky nabízí ještě více interaktivních cvičení zaměřených na český jazyk než Pravopisně, avšak chybí zde možnost vytvářet vlastní cvičení.

   Stejně jako u Pravopisně, i zde se cvičení vyhodnocují v průběhu jejich vyplňování.

3. Live Worksheets

   Tento web nabízený ve třech různých jazycích, v nichž čeština nefiguruje, umožňuje vytváření interaktivních nestrukturovaných pracovních listů pomocí základních interaktivních a neinteraktivních elementů, jako například z výběru správné možnosti z rozevíracího seznamu, textového pole, spojování možností, textu, videa a dalších.

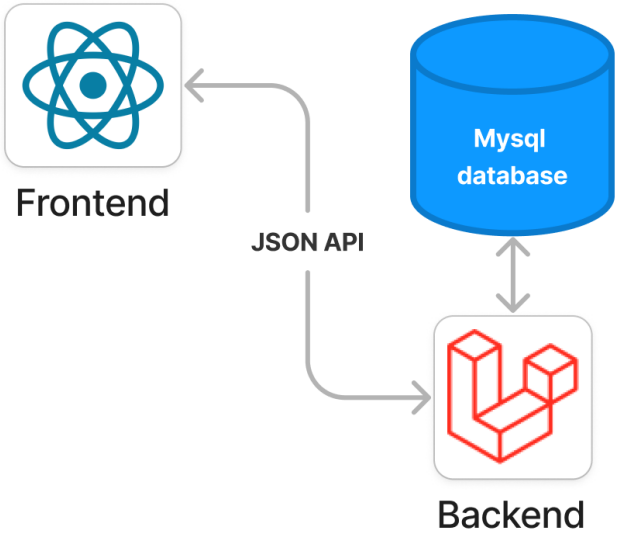
   Pro vytvoření pracovního listu je potřeba nahrát obrázek, jež je interpretován jako jeho pozadí. Na toto pozadí se poté umisťují výše zmíněné elementy.

Všechny z výše zmíněných webů umožňují organizaci uživatelů do struktur podobných třídám.

Moje aplikace na rozdíl od těchto webů umožňuje seskupení různých typů cvičení do jednoho samovyhodnocovacího listu, jenž je vyhodnocen až po jeho odeslání. Tím se více přibližuje standardním školním testům a úkolům.

Výše zmíněné pracovní listy a cvičení z nichž se skládají, jsou úzce zaměřeny na procvičování gramatiky českého jazyka.

3 Architektura



Obrázek 3.1: Architektura aplikace.
[1] [2]

Jak je vidět z obrázku 3.1, aplikace je tvořena ze tří částí, z webu (frontend), serveru (backend) a databáze.

Architektura na serveru i webu je vystavěna pro snadné přidání nového typu cvičení.

Frontend a backend spolu komunikují pomocí json api, definované pomocí json schémat (draft 7) [3] .

3.1 Frontend

Frontend byl napsán v Typescriptu [4] , používá knihovnu React [5] , umožňující rozdělení kódu do komponent a snadnější interakci s DOM modelem. Transformaci webu do podoby hostovatelné na jakémkoli serveru zajistila knihovna Vite.js [6] . Spoustu hotových komponent a snadnější vytváření nových zajistila knihovna Mantine [7] . Pro zobrazení tabulek, jež umožňují zobrazení úloh a jejich vyhodnocení, byla použita knihovna Mantine DataTable [8] . Neméně důležitou částí tohoto projektu byl editor pro vytváření úloh, Monaco Editor [9] .

3.2 Backend

Backend je založen na php frameworku Laravel [10] . Pro autentizaci byla použita knihovna Laravel Sanctum 4 spolu s Laravel Fortify [11] , jež poskytla koncové body pro správu uživatelů.

3.2.1 Validace

Jedna z forem validace příchozí komunikace na serveru je založena na již zmíněných json schématech, avšak není to jediná kontrola provedená na příchozím požadavku.

K validaci pomocí json schémat byly použity knihovny swaggest/php-code-builder [12] , pro generaci kontrolních php tříd, umožňujících import a export s kontrolou správnosti dle daného schématu, a swaggest/php-json-schema [13] pro samotnou validaci pomocí těchto tříd.

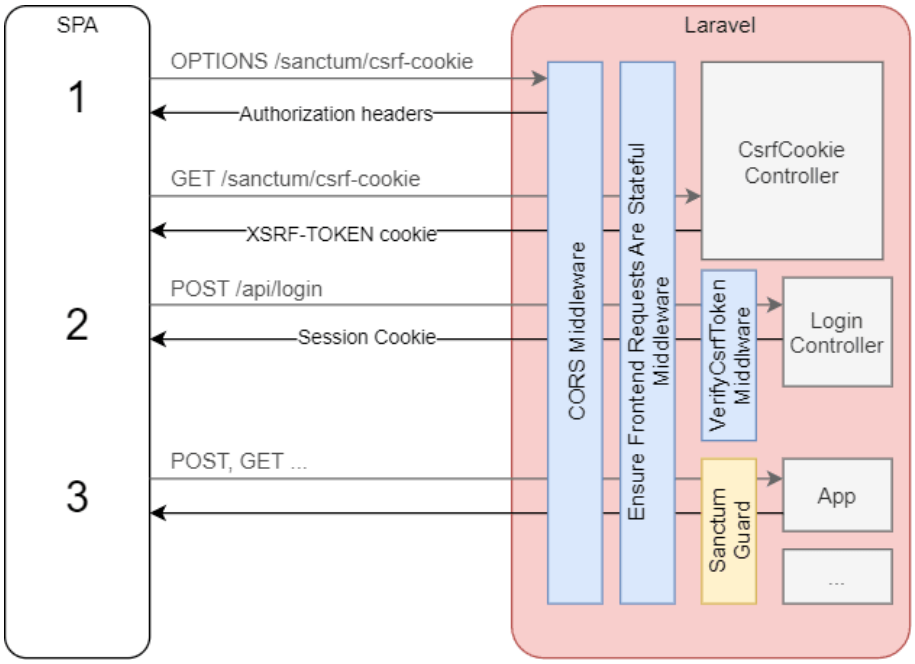
Jakákoli data, jež nejsou součástí url před query částí, tudíž musí být v json formátu. Jelikož ale http metoda GET [14] nepodporuje tělo, nebo je alespoň spousta serverů ignoruje, tak dochází k překladu json požadavku na query část url. Překlad je učiněn pomocí knihovny qs [15] .

Na serveru se potom takto zformulovaný query řetězec přeloží do php asociativního pole, jež je poté předáno validační třídě odpovídající požadovanému koncovému bodu. Je tu ovšem jeden problém, qs knihovna nerozlišuje mezi čísly a textovými řetězci.

Rozlišování mezi čísly a textovými řetězci bylo vyřešeno pomocí přidání jednoduché uvozovky ' před všechny textové řetězce před odesláním na server. A na serveru dochází v případě GET metody ke kontrole zda daný textový řetězec začíná s jednoduchou uvozovkou a pokud ano, je odebrána, pokud ne, je řetězec převeden na číslo.

4 Autentifikace

Pro autentizaci založenou na relacích byla použita knihovna Laravel Sanctum. Tento způsob autentikace vyžaduje aby frontend a backend sdíleli stejnou doménu. Toho je dosaženo umístěním spa (frontendu) a backendu na vlastní subdomény a na serveru jsou povoleny požadavky ze všech subdomén.



Obrázek 4.1: Diagram spa autentikace pomocí Laravel Sanctum [16]

Jak je vidět z obrázku 4.1:

1. Jako první spa (frontend) pošle požadavek na */sanctum/csrf-cookie*, jelikož se ale tento web nenachází na stejné doméně jako server (nacházejí se na jiných subdoménách, tudíž jde o požadavek napříč doménami [17]), prohlížeč požadavek přeruší a pošle speciální typ požadavku *OPTIONS*.

   OPTIONS požadavek je použit pro získání webů, jež mohou posílat požadavky na tento server. Tento typ požadavků se zopakuje pro všechny metody kromě *GET* a *HEAD*, u nichž dojde k zopakování až po vypršení času uložení v cache úložišti.

   Původní požadavek se odešle a *Ensure Frontend Requests Are Stateful Middleware* se pokusí získat uživatele asociovaného s aktuální relací, jelikož zatím žádná relace neexistuje, nic se nestane. Následně server vrátí session id a csrf token jako sušenky (cookies).

2. Poté se v případě přihlášení odešle požadavek, obsahující email a heslo, na */api/login*.

Prohlížeč automaticky pošle i sušenky a axios, jež frontend používá k posílání požadavků, automaticky přidá *X-XSRF-TOKEN* hlavičku.

Jelikož jde o POST metodu, prohlížeč znovu tento požadavek přeruší a pošle *OPTIONS* požadavek 1, poté *VerifyCsrfToken Middleware* porovná *X-XSRF-TOKEN* hlavičku s tokenem uloženým na serveru a v případě shody se požadavek dostane až ke koncovému bodu.

Ten se pokusí vyhledat podle poslaných údajů uživatele a v případě úspěchu vrátí jméno a email přihlášeného uživatele. Pokud takový uživatele neexistuje, vrátí chybu s kódem 1 (viz. api dokumentace).

3. V případě požadavku na koncový bod chráněný pomocí *Santum Guard* je postupováno stejně jako u přihlášení, akorát dojde ke kontrole existence uživatele asociovaného s touto relací, pokud selže, dojde k vrácení chyby Unauthenticated s kódem -10 (viz. api dokumentace).

[18] [19]

5 Úloha

Úloha je hlavním konceptem, kolem kterého se točí celá aplikace.
Skládá se ze dvou částí:

- Atributů pro hledání úlohy a orientace zobrazení úlohy (na šířku / na výšku)
- Zdrojového kódu úlohy, který definuje obsah samotného pracovního listu.

5.1 Atributy úlohy

- Jméno úlohy, jež slouží jako unikátní uživatelsky přívětivý identifikátor úlohy, podle něhož lze danou úlohu najít.
- Orientace úlohy, tento atribut jako jediný neslouží k hledání, ale k určení zobrazení úlohy na šířku či na výšku.
- Štítky slouží pro hledání úloh dle tématu, jež testují.
- Obtížnost určuje jak lehká je daná úloha s ohledem na ostatní atributy, zejména rozsah tříd.
- Rozsah tříd určuje rozsah tříd pro něž je úloha určena, tato možnost by měla reflektovat úplnost a komplexitu probírané látky spíše než obtížnost. Podporované jsou hodnoty od první třídy základní školy až po vše nad rámec střední školy.

5.2 Zdrojový kód

Samotný obsah úlohy se píše v značkovacím jazyce XML [20] s trochou vlastní syntaxe pro jednotlivé typy cvičení.

5.2.1 Části

Obsah se skládá z

- Popisu úlohy, jenž je možné využít pro instrukce a rady k vyplnění celé úlohy, jelikož tento popis je viditelný při vyplňování jakéhokoli cvičení dané úlohy.
- Ze cvičeních a jejich skupin

- Cvičení má
  - Váhu - od 1 do 100, jež představuje váhu použitou při výpočtu váženého průměru, jenž představuje výsledné hodnocení úlohy.
  - Typ, textový řetězec značící typ cvičení viz. typy cvičení
  - Instrukce, jež představují pokyny k vyplnění.
  - Obsah samotného cvičení.
- Skupina se skládá ze
  - Zdrojů, jež představují textové bloky nutné pro vyplnění všech cvičeních ve skupině. Tyto zdroje budou při vyplňování závislého cvičení vždy dostupné.
  - Členů skupiny
    - Cvičení (ref to Cvičení item)
    - Skupina (ref Skupina se skládá ...)

Cvičení by na sobě měly být nezávislé a jejich pořadí je implementačním detailem. K tvorbě textu na němž je závislých více cvičení by se měly použít skupiny a jejich zdroje.

5.3 Cvičení

$$\text{Skutečný počet bodů} = [(\text{počet bodů}) \times \text{váha}] \,/\, (\text{max počet bodů})$$

Při vyhodnocování cvičení se vezme počet a maximální počet bodů za cvičení, jež si každý typ cvičení počítá jinak. Počet bodů se vynásobí vahou cvičení a to celé se vydělí maximálním počtem bodů, výsledek je interpretován jako skutečný počet bodů získaných za cvičení.

$$\text{Reálný maximální počet bodů} = \text{váha}$$

A váha cvičení se stane reálným maximálním počtem získaných bodů.

5.3.1 Typy cvičení

Podporované jsou následující typy cvičení

Doplňovačka

Představuje cvičení, v němž se na vynechaná místa doplní správná písmena či slova.

Doplňovačka se skládá z textu proloženého vyplnitelnými komponentami.

- Textový vstup,
  očekává, že uživatel zadá stejný text jaký byl použit při jeho vytváření.
- Combobox,
  umožňuje výběr z možností specifikovaných při vytváření.
  Pořadí těchto položek je podle abecedy, takže nemá žádnou souvislost s umístěním správné možnosti. Kontroluje, zda je vybrána položka při vytváření označená za správnou.

$$\text{Počet bodů} = \text{Počet správně vyplněných komponent}$$

Při vyhodnocování se počet získaných bodů vypočítá jako počet správně vyplněných comboboxů a textových vstupů.

$$\text{Maximální počet bodů} = \text{Počet komponent}$$

Jako maximální počet bodů se interpretuje počet všech vyplnitelných komponent.

Opravování chyb

Je cvičení, jež pro své vyhodnocení používá Myers diff algoritmus.
Toto cvičení se vytváří pomocí dvou textů.

- Gramaticky správně napsaného textu.
- Textu, jež dostane uživatel k opravení na výše zmíněný správný text.

Při vytváření se vypočítá vzdálenost (počet vložených a smazaných znaků) mezi gramaticky správným textem a textem, který dostane uživatel. Tato vzdálenost se poté spolu s těmito dvěma texty uloží do databáze.

$$\text{Počet bodů} = (\text{uložená vzdálenost}) - (\text{míra odlišnosti})$$
$$\text{If (Počet bodů)} < 0 \text{ ? Počet bodů} = 0$$

Při vyhodnocení se vypočítá míra odlišnosti opraveného textu od správného textu. Následně se tato míra odečte od uložené vzdálenosti a pokud je menší než 0, nastaví se na 0. Poté se tato míra použije jako počet získaných bodů.

$$\text{Maximální počet bodů} = \text{uložená vzdálenost}$$

Jako maximální počet bodů je použita uložená vzdálenost.

5.4 Syntaxe

Zde je ukázka syntaxe jednotlivých částí úlohy zmíněných v předešlých sekcích.

5.4.1 Úloha

```
<dokument>
    <popis>
        <!-- Popis úlohy -->
    </popis>
    <polozky>
       <!-- Cvičení a skupiny -->
    </polozky>
</dokument>
```

Cvičení

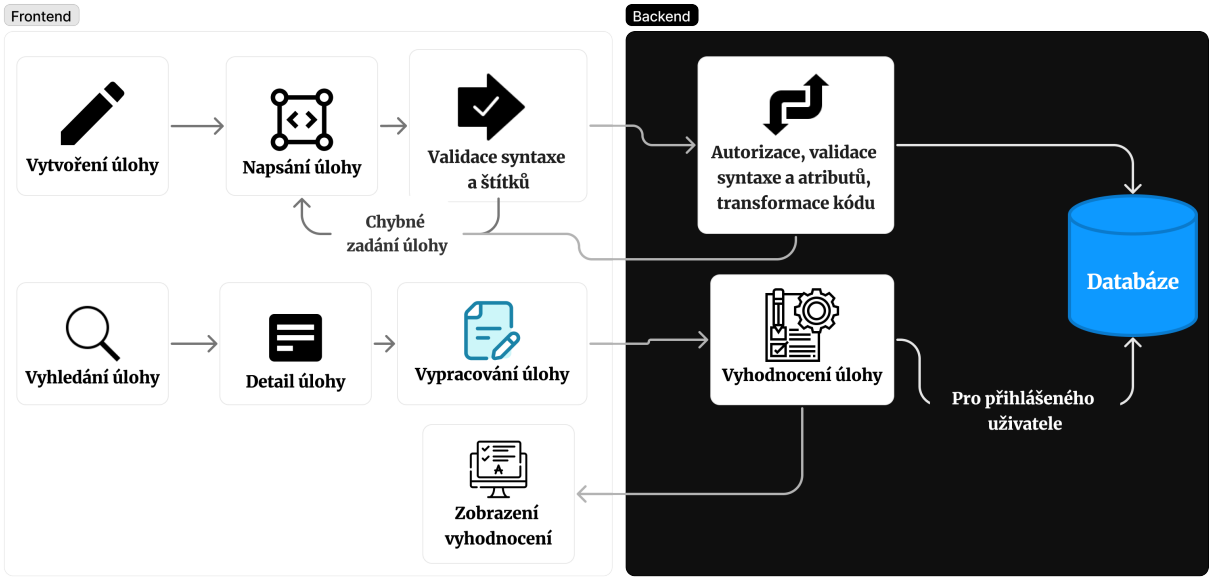
```
<!--
    type= Doplnovacka (Doplňovačka)  / OpravovaniChyb (Opravování chyb)
    vaha= váha, celé číslo od 1 do 100
 -->
<cviceni typ="" vaha="">
    <instrukce>
        <!-- Instrukce pro vyplnění cvičení -->
    </instrukce>
    <obsah>
        <!-- Obsah cvičení specifický pro každý typ cvičení -->
```

```
    </obsah>
</cviceni>
```

Skupina

```
<skupina>
    <zdroje>
        <zdroj>
            <!-- Textový zdroj -->
        </zdroj>
        <!-- Další zdroje -->
    </zdroje>
    <clenove>
        <!-- Cvičení a skupiny -->
    </clenove>
</skupina>
```

5.5 Životní cyklus úlohy



Obrázek 5.1: Životní cyklus úlohy.

Jak je vidět z obrázku 5.1, k vytvoření úlohy musí uživatel s rolí učitele vyplnit atributy úlohy 5.1 a její zdrojový kód 5.2. Po jejich kontrole je úloha odeslána na server, kde jsou atributy a role tvůrce znovu zkontrolovány.

Zdrojový kód je na serveru parsován pomocí php xml sax parseru [21] Tento parser je zabalen do mého vlastního parseru zajišťujícího správnost vrácené pozice v případě chyby. Jelikož tento xml parser kontroluje pouze správnost xml syntaxe, musel jsem vytvořit vlastní řešení pro kontrolu mé vlastní struktury 5.2 a její transformaci na podobu uložitelnou v databázi.

Výše zmíněný problém jsem vyřešil pomocí definování mé vlastní struktury tříd podobné DOM uzlům. Tyto třídy zkontrolují názvy elementů, jejich atributy a děti, jež jsou zkontrolovány pomocí tříd za ně odpovědných. Při kontrole jsou zároveň získaná data transformována pro uložení v databázi.

Po již zmíněném vytvoření přichází na řadu vyhledání úlohy a její vyplnění. Při zobrazení se podle orientace úlohy vybere komponenta pro zobrazení, tyto komponenty využívají stejné komponenty pro zobrazení zdrojů a stejnou komponentu pro zobrazení jednotlivých cvičení.

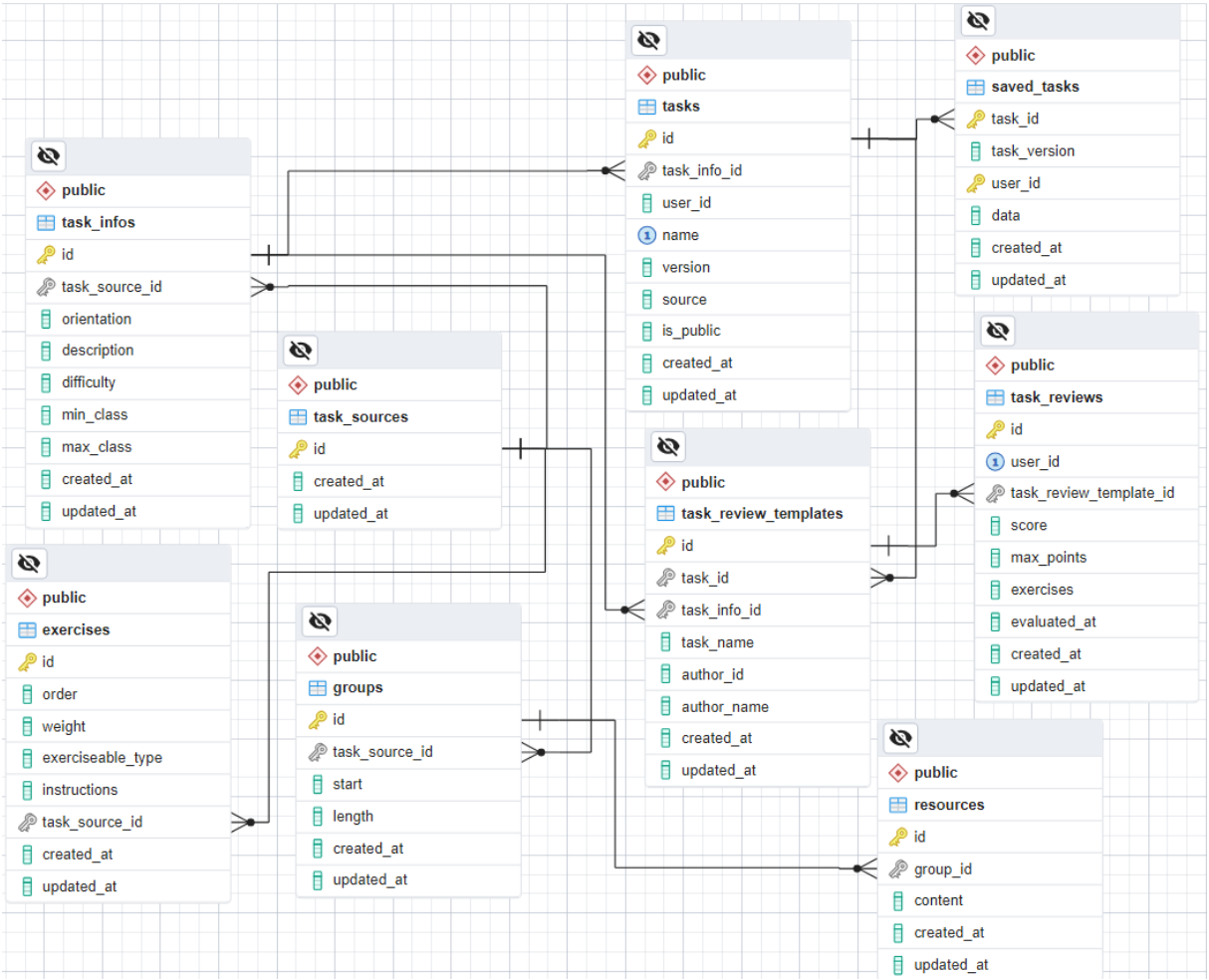
Komponenta pro cvičení zobrazuje instrukce a poté samotný obsah, závislý na typu cvičení. K získání komponenty pro zobrazení obsahu je použit slovník přebírající typ a vracející danou komponentu.

Každý typ cvičení musí definovat komponentu pro vyplnění a komponentu pro zobrazení vyhodnocení.

6 Databáze

Při vývoji projektu byla použita relační databáze PostgreSQL [22] . Ale při produkci byla kvůli hostujícímu prostředí (školní web) změněna na MySQL [23] databázi. Tato změna byla díky Laravelu (odkaz na technologie) velice jednoduchá.

V této kapitole bude zmíněno uložení hlavních zdrojů v databázi, jež jsou vidět na obrázku 6.1. Jmenovitě úloh a jejich vyhodnocení, jež se ukládají pro všechny přihlášené uživatele.



Obrázek 6.1: ERD databáze

6.1 Tabulka tasks

Představuje jednotlivé úlohy uložené v databázi, logické operace provedené nad úlohou se projevují těmi samými operacemi nad touto tabulkou a přidruženými tabulkami.

6.1.1 Atributy

- Jméno (Name)
- Id autora (user_id), cizí klíč na tabulku users
- Verzi (version)
- Zdrojový kód 5.2 (source), jenž je uložen ve stejné podobě v níž byl napsán.
- Zda je úloha veřejně přístupná (is_public) - veřejně přístupné úlohy jsou přístupné i pro ostatní uživatele, neveřejné úlohy vidí pouze jejich autor

Tato tabulka se odkazuje na task_infos 6.4, kde se ukládají další atributy úlohy a jež slouží i jako spojení s rozparsovaným zdrojovým kódem, nutným pro samotné vyplnění a vyhodnocení úlohy.

Aby byly důvody uložení úlohy do takovéto struktury jasnější, je nutné započítat uložení vyhodnocení těchto úloh.

6.2 Tabulka task_reviews

Ukládá vyhodnocení úloh pro jednotlivé uživatele.

6.2.1 Atributy

- Id uživatele (user_id), jemuž vyhodnocení patří.
- Uživatelovo skóre (score).

$$\text{Skóre} = (\text{počet bodů}) \,/\, (\text{maximální počet bodů})$$

  Uložené jako podíl dosaženého počtu bodů a maximálního počtu bodů.

- Maximální dosažitelný počet bodů (max_points)
- Cvičení (exercises), obsahující vyhodnocená cvičení. Toto pole je uloženo jako json, protože je s ním nakládáno jako s jedním celkem.
- Časové razítko vyhodnocení (evaluation_timestamp), datum a čas zkontrolování úlohy.

Odkazuje se na tabulku task_review_templates 6.3, v níž jsou uloženy atributy společné pro všechna vyhodnocení dané verze úlohy.

6.3 Tabulka task_review_templates

Obsahuje atributy společné pro všechny vyhodnocení dané verze úlohy.

6.3.1 Atributy

- Jméno (task_name), jež úloha nesla při jejím vyhodnocením.
- Jméno (author_name), patřící autorovi úlohy v době prvního vyhodnocení úlohy.
- Id autora úlohy (author_id), pokud ještě existuje.
- Id úlohy (task_id) 6.1, jež byla vyhodnocena, pokud ještě existuje. Toto pole se odkazuje na danou úlohu i po jejím upravením.

Tato tabulka se stejně jako tasks 6.1 odkazuje na task_infos 6.4.

6.4 Tabulka task_infos

Tato tabulka pro další informace o úloze slouží jako propojení mezi úlohami a jejich vyhodnoceními, jež ji mimo jiné používají jako verzi úlohy. Ukládá všechny zbývající atributy úlohy 5.1, jež nejsou uloženy v tasks 6.1 tabulce. Štítky úlohy jsou ukládány pomocí dalších dvou tabulek:

- tags, jež obsahuje jméno a id štítků
- tag_task_infos, sloužící jako pivot tabulka skládající se z id štítku (tag_id) a id informací o úloze (task_info_id)

Dále se tato tabulka odkazuje na task_sources 6.5.

6.5 Tabulka task_sources

Tato tabulka se skládá pouze s jednoho sloupce a tím je id, slouží jako spojení mezi úlohami 6.1, šablonami vyhodnocení úloh 6.3 a rozparsovanými zdroji úloh.

6.6 Tabulka groups

Představuje skupiny cvičení v úloze. Jelikož skupina může obsahovat další skupiny, byla kvůli jednoduchosti zvolena struktura, jež nemá žádný cizí klíč a i přesto se odkazuje na jednotlivá cvičení.

6.6.1 Atributy

- Index prvního cvičení ve skupině (start).
- Počet cvičení ve skupině (length).
- Id příslušného zdroje úlohy (task_source_id) 6.5.

Tato struktura naprosto závisí na tabulce exercises 6.7 přes indexy cvičení.

6.7 Tabulka exercises

Obsahuje prvky společné pro všechny typy cvičení.

6.7.1 Atributy

- Id cvičení (exerciseable_id).
- Index cvičení (order), určující pořadí, ve kterém bylo cvičení definováno ve zdrojovém kódu úlohy 5.2.
- Váha cvičení (weight) 5.2.1.
- Typ cvičení (exerciseable_type) 5.3.1.
- Instrukce (instructions) 5.2.1.
- Id příslušného zdroje úlohy (task_source_id) 6.5.

Na této tabulce závisí tabulky pro jednotlivé typy cvičení 5.3.1 přes id cvičení, jež v jednotlivých tabulkách plní roli primárního i cizího klíče.

7 Výsledky

Byl vytvořen jednoduchý informační systém s anonymním i neanonymním přístupem, jež umožňuje vytváření a správu symovyhodnocovacích pracovních listů pro procvičování gramatiky českého jazyka. Pracovní listy se skládají z různých samovyhodnocovacích cvičení s instrukcemi a váhou použitou při váženém průměru, pomocí nějž se pracovní list po odeslání vyhodnotí. Toto vyhodnocení je poté zobrazeno uživateli a v případě přihlášeného uživatele i uloženo do databáze. Archivovaná vyhodnocení je možné zobrazit i smazat.

Tvorba výše zmíněných pracovních listů je umožněna pouze uživateli s rolí „učitel“ a to skrze formulář pro zadání atributů 5.1 a zdroje úlohy ve formátu XML, obsahující samotná cvičení 5.2.

V aplikaci chybí následující funkcionality ze zadání projektu:

- Cvičení typu diktát.
- Konfigurace podmínek, za nichž se odkryje řešení úlohy. V nynější verzi se řešení odkryje všem ihned po odeslání vyplněné úlohy.
- Role administrátora, jež je pro aplikaci v nynější fázi zbytečná.

8 Závěr

V nynější podobě je informační systém použitelný, avšak pro dlouhodobější používání by bylo potřeba, kromě opravy případných chyb, přidat a upravit spoustu stávajících funkcionalit, jmenovitě:

- Uložení rozpracované úlohy.

  Tato možnost je již na serveru podporována, i když neotestována. Bylo by tedy nutné přidat možnost uložení na webu a opravit případné chyby na serveru.

  V případě změny uložené rozpracované úlohy jejím autorem předvyplnit všechny nezměněné části. U částí, jež byly změněny, ukázat, co zde bylo vyplněno před autorovou úpravou úlohy. Tato část souvisí s další úpravou stávajících funkcionalit 8.

- Třídy pro uživatele s rolí „učitel“.

  Výše zmíněná funkcionalita je v různých formách implementována i ve všech aplikacích zmíněných v rešerši 2.

  Vyžadovala by velké rozšíření na straně serveru i na straně aplikace v prohlížeči. Umožňovala by správu tříd pro uživatele s rolí „učitel“, což zahrnuje i zvaní ostatních uživatelů (žáků) do těchto tříd a možnost zadávat žákům úkoly s konfigurovatelnými podmínkami, za nichž se odkryje řešení. Tato vyhodnocení by poté byla odeslána učiteli k zobrazení.

- Více typů cvičení.

  Tato aplikace byla vyvinuta s ohledem na přidávání nových typů cvičení. Díky tomu by se obtížnost tohoto úkolu odvíjela od komplexity přidávaných cvičení.

- Změna atributů úlohy 5.1 bez předpokládání změny zdrojového kódu 5.2.

  V nynější verzi aplikace se předpokládá, že pokud došlo ke změně jakéhokoli atributu úlohy, došlo také ke změně zdrojového kódu. Jinými slovy nerozlišuje se mezi změnami atributů a zdrojového kódu.

- Úprava XML struktury cvičení ve zdrojovém kódu úlohy.

  V nynější podobě se obsah cvičení nedá zkontrolovat pomocí XML schémat, protože se typ cvičení určuje pomocí atributu. Aby se dal celý zdroj zkontrolovat pomocí XML schémat, musel by se typ cvičení určovat pomocí elementu různého pro každý typ cvičení, jež by se umístil místo elementu „content“ 5.2.

Literatura

[1] PLATFORMS, Meta, 8 November 2023n. l. React Logomark SVG. In: *Wikimedia Commons* [online]. [cit. 2024-03-30]. Dostupné z: https://commons.wikimedia.org/wiki/File:React_Logo_SVG.svg

[2] Laravel, 5 January 2019n. l. Laravel's logo. In: *Wikimedia Commons* [online]. [cit. 2024-03-30]. Dostupné z: https://commons.wikimedia.org/wiki/File:Laravel.svg

[3] Creating your first schema. *JSON Schema* [online]. [cit. 2024-03-29]. Dostupné z: https://json-schema.org/learn/getting-started-step-by-step

[4] The TypeScript Handbook, ©2012-2024. MICROSOFT. *TypeScript: JavaScript With Syntax For Types.* [online]. [cit. 2024-03-31]. Dostupné z: https://www.typescriptlang.org/docs/handbook/intro.html

[5] Quick Start, ©2024. META. *React* [online]. [cit. 2024-03-31]. Dostupné z: https://react.dev/learn

[6] Getting Started, ©2019. *Vite* [online]. [cit. 2024-03-29]. Dostupné z: https://vitejs.dev/guide

[7] Getting started, March 26, 2024. *Mantine* [online]. [cit. 2024-03-29]. Dostupné z: https://mantine.dev/getting-started

[8] FLORESCU, Ionut-Cristian, 2024. Mantine DataTable. GITHUB, INC. *GitHub* [online]. 2024 [cit. 2024-03-31]. Dostupné z: https://icflorescu.github.io/mantine-datatable

[9] ATOYAN, Suren, 2023. Monaco-react. GITHUB, INC. *GitHub* [online]. [cit. 2024-03-31]. Dostupné z: https://github.com/suren-atoyan/monaco-react#readme

[10] Laravel Documentation, ©2011-2024. *Laravel - The PHP Framework For Web Artisans* [online]. [cit. 2024-03-31]. Dostupné z: https://laravel.com/docs/10.x/readme

[11] Laravel Fortify, ©2011-2024. LARAVEL HOLDINGS INC. *Laravel* [online]. [cit. 2024-03-31]. Dostupné z: https://laravel.com/docs/10.x/fortify

[12] SWAGGEST, ©2024. Swaggest JSON-schema enabled PHP code builder. GITHUB, INC. *GitHub* [online]. [cit. 2024-04-01]. Dostupné z: https://github.com/swaggest/php-code-builder

[13] SWAGGEST, 2023. Swaggest JSON-schema implementation for PHP. GITHUB, INC. *GitHub* [online]. [cit. 2024-04-01]. Dostupné z: https://github.com/swaggest/php-json-schema

[14] GET, ©1998-2024. *MDN Web Docs* [online]. [cit. 2024-03-31]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/GET

[15] HARBAND, Jordan, 2024. Qs. *GitHub* [online]. [cit. 2024-03-31]. Dostupné z: https://github.com/ljharb/qs

[16] [Laravel Sanctum SPA Authentication diagram], 2020. In: BAILLY, Nicolas. *Laravel Sanctum Explained : SPA Authentication - DEV Community* [online]. 24. 5. 2020 [cit. 2024-03-30]. Dostupné z: https://dev.to/nicolus/laravel-sanctum-explained-spa-authentication-45g1

[17] Cross-Origin Resource Sharing (CORS), ©1998-2024. *MDN Web Docs* [online]. [cit. 2024-03-31]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

[18] BAILLY, Nicolas, 2020. *Laravel Sanctum Explained : SPA Authentication* [online]. 24. 5. 2020 [cit. 2024-03-29]. Dostupné z: https://dev.to/nicolus/laravel-sanctum-explained-spa-authentication-45g1

[19] 22 Jan 2024n. l. Laravel Sanctum SPA Auth Overview. In: *YouTube* [online]. [cit. 2024-03-29]. Dostupné z: https://www.youtube.com/watch?v=ujDnuzi1t1s

[20] XML introduction, ©1998-2024. *MDN Web Docs* [online]. [cit. 2024-03-31]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/XML/XML_introduction

[21] BAKKEN, Thies C. ARNTZEN a Sterling HUGHES. XML Parser. *PHP: Hypertext Preprocessor* [online]. [cit. 2024-03-25]. Dostupné z: https://www.php.net/manual/en/book.xml.php

[22] PostgreSQL 16.2 Documentation, ©1996-2024. THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. *PostgreSQL: The World's Most Advanced Open Source Relational Database* [online]. [cit. 2024-03-31]. Dostupné z: https://www.postgresql.org/docs/current

[23] MySQL :: MySQL Documentation, ©2024. ORACLE. *MySQL :: Developer Zone* [online]. [cit. 2024-03-31]. Dostupné z: https://dev.mysql.com/doc

Seznam obrázků

3.1 Architektura aplikace. 11

4.1 Diagram spa autentikace pomocí Laravel Sanctum 13

5.1 Životní cyklus úlohy. 19

6.1 ERD databáze 21

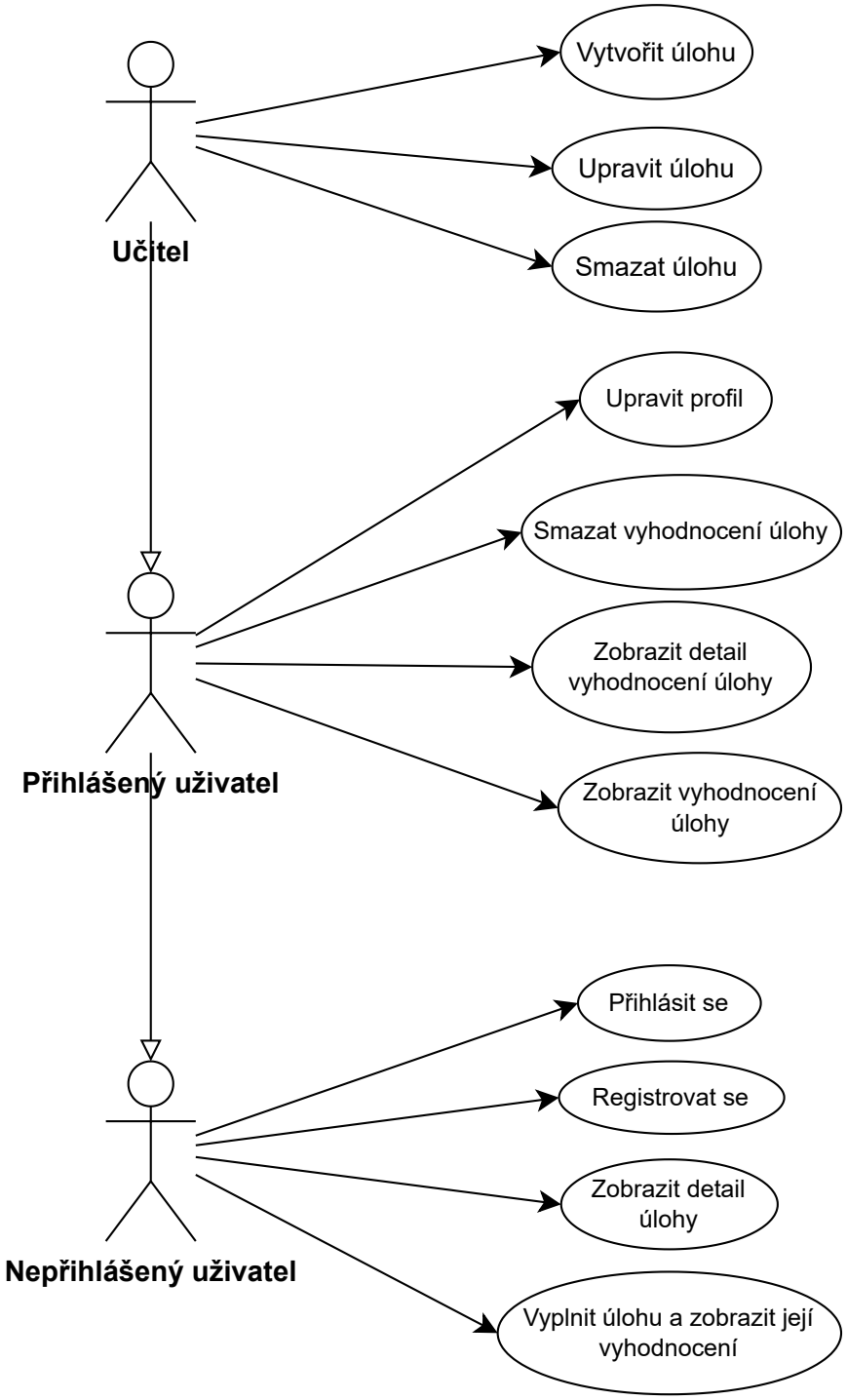
10.1 Use case diagram. I

9 Seznam příloh

I ... Use case diagram

II ... Dokumentace API

Use case diagram 1



Obrázek 10.1: Use case diagram.

Dokumentace API 2

API je zdokumentováno pomocí json schémat (draft 7) nacházejících se ve složce mat-project-laravel-test /schemas v repozitáři serveru (Backend). Tato složka je kořenovým adresářem pro všechna schémata a všechny cesty zde uvedené z ní vycházejí, pokud není uvedeno jinak.

1.1 Reference

Schémata se na sebe navzájem odkazují pomocí „$ref“ klíče obsahujícího cestu k odkazovanému schématu. Cesty jsou definované ve dvou formátech.

- Cesta relativní od kořenové složky, jež se zapisuje pomocí formátu „@/cesta k souboru“, v níž je zavináč nahrazen cestou ke kořenové složce „schemas“.
- Cesta relativní od daného schématu, jež se zapisuje stejně jako jakákoli jiná relativní cesta „./cesta k souboru“.

Jakýkoli z těchto formátů může odkazovat na více schémat naráz pomocí glob vzoru.

1.2 Základní struktura požadavků a úspěšných odpovědí

Základní struktura požadavků (requests) a úspěšných odpovědí (responses) je definována pomocí request.special.json a success_response.special.json schémat. Tuto strukturu splňují všechny koncové body kromě těch, jež jsou definovány pomocí Laravel Fortify. U těchto koncových bodů se základní struktura požadavků nebere v potaz.

Formáty specifické pro daný koncový bod jsou umístěny ve složce defs/endpoints, kde jsou seskupeny ve stromové struktuře libovolného zanoření. Složka definující jeden koncový bod obsahuje minimálně request.special.json a response.special.json schémata, některé z nich obsahují i error.special.json a složku errors s chybami specifickými pro daný koncový bod (viz. errory).

1.3 Základní struktura chyb

Základní struktura chyb je definována pomocí errors/error_response.special.json schématu.

Detaily specifické pro jednotlivé koncové body jsou uloženy spolu s požadavky a odpověďmi pro daný koncový bod 1.2. Zároveň však může každý koncový bod vrátit chybu s detailem definovaným pomocí schématu „defs /errors/general.special.json“ představujícího všeobecný detail chyby.

Každá chyba obsahuje zprávu a kód detailu chyby. Kódy chyb jsou definované v souboru „/mat-project-laravel-test/errors/errors.json“, jež se nenachází ve složce „schemas“. Tento soubor obsahuje objekty a textové řetězce představující mapování mezi chybovými kódy a schématy, jež tyto kódy používají.

Klíče objektů ve výše zmíněném souboru značí prvky cesty, jež je relativní vzhledem k umístění souboru. Klíče textových řetězců se dále dělí.

- Kódové klíče jsou ve formátu celého čísla a představují kód chyby. Jejich hodnotou je název souboru, k němuž je daný kód přidružen. Cesta k danému poli jež se nachází v souboru je definována pomocí níže zmíněných speciálních klíčů.
- Speciální klíč „$posfix“ představuje řetězec, který je přidán na konec všech jmen souborů, definovaných pomocí výše zmíněných klíčů, nacházejících se v jednom z podřízených objektů.

1.4 Chybové kódy

1.5 Koncové body

Standardní chybové kódy jsou ve tvaru negativních celých čísel.
Chybové kódy specifické pro daný koncový bod jsou ve tvaru přirozených čísel.

Koncové body jsou vypsány ve formátu „{METODA} *cesta*“.

1.5.1 Učitel

POST /api/task/create

Vytvoří novou úlohu.

PUT /api/task/{id}/update

Upraví úlohu se specifikovaným id.

DELETE /api/task/{id}/delete

Smaže úlohu se specifikovaným id.

GET /api/my_task/list

Vrátí seznam úloh vytvořených aktuálním uživatelem, jež splňují požadovaná kritéria.

GET /api/my_task/{id}/detail

Vrátí detailní informace o úloze se specifikovaným id, jež byla vytvořena aktuálním uživatelem.

GET /api/task/{id}/source

Vrátí zdrojový kód úlohy 5.2.

1.5.2 Přihlášený uživatel

GET /api/user/get_profile

Vrátí informace o aktuálním uživateli.

GET /api/review/list

Vrátí seznam vyhodnocení úloh vyplněných uživatelem.

GET /api/review/{id}/detail

Vrátí informace o vyhodnocení úlohy se specifikovaným id, jež byla vyplněna aktuálním uživatelem.

GET /api/review/{id}/get

Vrátí vyhodnocení úlohy se specifikovaným id, jež byla vyplněna aktuálním uživatelem.

DELETE /api/review/{id}/delete

Smaže vyhodnocení úlohy se specifikovaným id, jež byla vyplněna aktuálním uživatelem.

POST /api/logout

Odhlásí aktuálního uživatele.

Tento bod nesplňuje základní strukturu požadavků 1.2.

PUT /api/user/profile-information

Upraví profil aktuálně přihlášeného uživatele pomocí poslaných informací.

Tento bod nesplňuje základní strukturu požadavků 1.2.

PUT /api/user/password

Upraví heslo aktuálně přihlášeného uživatele pomocí poslaných informací.

Tento bod nesplňuje základní strukturu požadavků 1.2.

1.5.3 Nepřihlášený uživatel

GET /api/task/create_info

Vrátí štítky, třídy a další zdroje a informace nutné pro vytvoření úlohy i pro filtrování seznamů úloh.

GET /api/task/{id}/detail

Vrátí veřejně přístupné detailní informace o úloze se specifikovaným id.

GET /api/task/{id}/take

Vrátí vyplnitelnou úlohu se specifikovaným id.

GET /api/task/list

Vrátí seznam úloh splňujích požadovaná kritéria.

POST /api/task/{id}/evaluate

Vyhodnotí úlohu, se specifikovaným id, pomocí poslaných hodnot.

GET /sanctum/csrf-cookie

Vrátí CSRF token, používá se pro autentifikaci 4.

POST /api/login

Pokusí se přihlásit uživatele pomocí poslaných informací.

Tento bod nesplňuje základní strukturu požadavků 1.2.

POST /api/register

Pokusí se registrovat uživatele pomocí poslaných informací.

Tento bod nesplňuje základní strukturu požadavků 1.2.