Masarykova univerzita
Fakulta informatiky

Portál pre písanie a vyhodnocovanie diktátov

Diplomová práca

Bc. Jakub Rumanovský

Brno, Jeseň 2015

Prehlásenie

Prehlasujem, že táto diplomová práca je mojím pôvodným autorským dielom, ktoré som vypracoval samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používal alebo z nich čerpal, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

Bc. Jakub Rumanovský

Vedúci práce: Mgr. Marek Grác, Ph.D.

Poďakovanie

Ďakujem vedúcemu práce Mgr. Marekovi Grácovi, Ph.D. za vedenie, trpezlivosť a rady pri písaní diplomovej práce a mojej rodine, kamarátom a priateľke za podporu.

Zhrnutie

Úlohou diplomovej práce je vytvorenie diktátového webového systému určeného predovšetkým žiakom základných a stredných škôl a ich učiteľom. Žiak bude môcť samostatne trénovať zvolený diktát, po jeho napísaní si ho nechať opraviť a okamžite listovať výsledkami. Taktiež sa bude môcť k výsledku kedykoľvek vrátiť a prezerať si svoje štatistiky. Učiteľom systém umožní nahrávať diktáty a spravovať administrátorom pridelenú skupinu žiakov – tzv. školskú triedu. Portál bude tiež poskytovať verejne dostupné rozhranie slúžiace na opravu diktátov, ktoré bude zároveň použiteľné ako služba.

Abstract

The aim of this master thesis is to create web-based system for dictate training. The users will mostly be teachers and students of elementary schools and high schools. A student will be able to train chosen dictate by himself, let the system correct the dictate automatically and immediately see the results. He will also be able to see the results later and list his or her own statistics of already trained dictates. Teacher will be able to upload a dictate and administrate a group of students called school class. This portal will also provide public endpoint for dictate correction, which can be used as a service.

Kľúčové slová

dictate, trainer, correction module, grammar correction, Java EE

Obsah

1 Úvod

Informačné technológie v súčasnosti prenikajú do stále väčšieho množstva odvetví ľudskej činnosti. Sú medzi nimi aj také oblasti, v akých by sme ich pred pár rokmi nevedeli predstaviť. V mnohých z nich majú nepopierateľne veľký potenciál budúceho rozvoja. Jedným z príkladov takéhoto odvetvia je vzdelávanie prostredníctvom informačných technológií[1]. V predchádzajúcich rokoch došlo v tejto oblasti k rapídnemu rozmachu, hlavne od začiatku 21. storočia[1]. Prispeli k tomu nemalou mierou aj niektoré renomované univerzity [2], ktoré poskytujú svoje kurzy zdarma alebo za rozumný poplatok. Súčasná evolúcia vzdelávania poskytuje mnoho možností rozvoja pre každého, kto má záujem. Je dokonca možné absolvovať vopred vybrané kurzy a získať certifikát, ktorý potom možno uviesť ako doplnkové vzdelanie v životopise a preukázať tak svoju kompetenciu v určitom odvetví alebo činnosti[3]. Z horeuvedených príkladov je vidieť, že e-learning v takmer akejkoľvek podobe má zmysel i potenciál a prináša nesporné výhody pre žiakov, študentov aj vyučujúcich.

K dôležitým schopnostiam každého človeka, či už v profesionálnom alebo súkromnom živote, patrí písomný prejav. Nepochybne jeden z jeho dôležitých aspektov je správna gramatika a pravopis. Tie sa žiaci učia na školách od najútlejšieho veku. Osvojovanie gramatiky na školách prebieha už niekoľko desaťročí takmer nezmeneným spôsobom. K dôležitým nástrojom na trénovanie správneho pravopisu patrí písanie diktátov. Do tejto oblasti moderné technológie ešte celkom neprenikli. Diktát sa dnes stále píše zväčša na papier a následne je opravovaný a hodnotený učiteľom v škole.

Pretože v súčasnej dobe neexistuje aplikácia, ktorá by používateľovi plnohodnotne umožňovala trénovať diktáty za pomoci počítača, je v nasledujúcom texte navrhnutý webový portál, ktorý slúži ako doplnok, poprípade alternatíva k testovaniu gramatiky v škole. Žiak sa môže kedykoľvek, či už na popud učiteľa alebo z vlastnej vôle otestovať. Aplikácia si kladie za cieľ rozšíriť písanie a trénovanie diktátov, ktoré sa momentálne píše žiakmi a opravuje učiteľmi v škole o výhody, ktoré ponúkajú informačné technológie. Konkrétne

1. anglicky e-learning

medzi ne patria rýchlejšia – a hlavne automatická – oprava chýb, možnosť ukladania a následného analyzovania chýb v širšom kontexte (či už v prípade konkrétneho študenta alebo prípadne celej triedy) a v neposlednom rade tiež zobrazenie týchto chýb v štatistike.

Portál má už v dobe písania tejto práce záujemcov z radov učiteľov, ktorí sa neskôr spolu so svojimi žiakmi budú podieľať na funkčnom testovaní a hlásení chýb. Jednak sú to vyučujúci vybraných základných škôl a jednak sa systém bude využívať na Pedagogickej fakulte Masarykovej univerzity pri výuke budúcich pedagógov.

1.1 Ciele práce

Úlohou tejto diplomovej práce je vytvorenie webového systému na komplexnú prácu s diktátmi, pričom je vopred určená vývojová platforma Java Enterprise Edition. Tento systém bude pozostávať z:

1. Administračnej časti pre vyučujúcich, ktorá bude obsahovať
   - Možnosť nahrania vlastného diktátu a doplnenie anotácií k diktátu priamo v prostredí
   - Základné štatistiky o využívaní jednotlivých diktátov
   - Prácu so skupinami používateľov
2. Používateľskej časti pre študentov, v ktorej bude možné
   - Prihlásiť sa pomocou mailu alebo sociálnych sietí
   - Písať diktáty a nechať si ich po napísaní opraviť
3. Samostatného modulu integrovaného s webovou aplikáciou, ktorý bude dostupný cez definované rozhranie a bude implementovať pravidlá nájdené počítačovými lingvistami. Tieto pravidlá odhaľujú a zaraďujú jednotlivé chyby do kategórií (Viac o jednotlivých typoch chýb v kapitole C) a pridávajú zdôvodnenia pre vybrané jazykové javy v češtine.

1.2 Štruktúra práce

Kapitola 1 - Úvod uvádza čitateľa do problematiky výučby za pomoci výpočtovej techniky a popisuje motiváciu písania práce.
Kapitola 2 - Analýza systému zahŕňa analýzu súčasného stavu a možností trénovania diktátov za pomoci počítača. Druhá časť kapitoly sa zaoberá popisom navrhovaného systému, rozoberá jeho výhody oproti súčasnému stavu a špecifikuje požiadavky na aplikáciu.
Kapitola 3 - Návrh a implementácia systému obsahuje popis návrhu diktátového systému, poprípade zmieňuje niektoré implementačné detaily, pokiaľ nie sú implicitne zrejmé. Na začiatku sa kapitola venuje problémom a slepým vetvám vývoja popisovaného systému spolu s odôvodnením, prečo sa konkrétna požiadavka nakoniec riešila iným spôsobom. Kapitola tiež zahŕňa popis štruktúry aplikácie a postupne rozoberá každú jej vrstvu zvlášť.
Kapitola 4 - Modul na opravu diktátov obsahuje popis samostatného modulu na opravu diktátov, ktorý je integrovaný spoločne s aplikáciou. Kvôli väčšej prehľadnosti práce bol presunutý do samostatnej kapitoly. Tá zahŕňa analýzu a návrh modulu ako aj popis jeho fungovania a v prípade potreby aj implementačné detaily (podobne ako v kapitole 2 a 3).
Kapitola 5 - Testovanie systému popisuje spôsob testovania portálu pre písanie a vyhodnocovanie diktátov.
Kapitola 6 - Nasadenie systému sa venuje detailom nasadenia systému a použitým technológiám.
Kapitola 7 - Záver zhŕňa dosiahnuté výsledky a popisuje plány a rozšírenia do budúcnosti.

2 Analýza systému

2.1 Existujúce riešenia

Ako bolo zmienené v úvode práce, hrajú pri výuke a trénovaní gramatiky ešte stále významnú úlohu zbierky diktátov, z ktorých je vybraný diktát diktovaný učiteľom v triede počas vyučovania. Zbierka diktátov slúži ako doplnok k učebniciam českého jazyka, pretože učebnice (hlavne kvôli obmedzenému rozsahu), nemôžu prinášať dostatočné množstvo materiálov k precvičovaniu pravopisu [4].

Ako ďalší zdroj cvičenia gramatiky je možné uviesť veľkú skupinu cvičení dostupných zadarmo na internete. Podľa Prášilovej[4] je výhodou takejto formy precvičovania okamžité vyhodnotenie, ktoré však nemusí byť príliš dobrý motivačný prvok.

Ďalšou skupinou existujúcich riešení sú *audiodiktáty*, ktoré vychádzajú z rovnakých zdrojových textov ako zbierky diktátov. Nevýhodou tohto riešenia sú technické problémy, ku ktorým môže dochádzať počas diktovania (napr. šum a preskakovanie. Tieto však už dnes vďaka technologickému pokroku nie sú príliš kritické). Ku komplikáciám môže podľa Prášilovej tiež dochádzať vo chvíli keď žiak zle rozumie, chce sa spýtať, tým sa dostane do časového stresu a stráca pozornosť. Hlavnú nevýhodu však vidí v nemožnosti prispôsobiť tempo danej skupine. Výhodou naproti tomu je zvýšenie pozornosti, ktorú môže učiteľ žiakom venovať namiesto čítania textu.

V súčasnosti sa na internete objavujú aj celé portály zamerané na precvičovanie gramatických javov. Jedným zo zástupcov tejto kategórie je portál *pravopisne.cz*, ktorý ponúka širokú paletu rôznych gramatických cvičení, zahŕňajúcu doplňovačky, gramatickú teóriu, hry na precvičovanie pravopisu a gramatických javov ako aj tzv. zvukové diktáty[1]. Tie však fungujú tak, že si žiak zvukový súbor cez prehliadač spustí a píše na papier a potom ho porovná so správnym riešením, dostupným na stránke bezprostredne po skončení cvičenia. Ďalším portálom patriacim do tejto skupiny je web *umimcesky.cz*, kde si žiak precvičuje gramatické javy pomocou testových otázok (väčšinou typu *vyber správnu možnosť*) a po označení odpovede portál

1. Viac na `http://pravopisne.cz/category/zvukove-diktaty/`

zobrazí, či bola správna a ak áno, prečo boli ostatné nesprávne.

Najnovším projektom spadajúcim do kategórie precvičovania gramatiky je projekt webového portálu na trénovanie diktátov vznikajúci ako súčasť Informačného systému Masarykovej Univerzity v Brne[2]. Tento portál v budúcnosti uvažuje o využití modulu navrhovaného v tejto práci ako externej služby k oprave diktátov.

2.1.1 Problémy a nedostatky existujúcich riešení

V nasledujúcej podkapitole v krátkosti zhrniem hlavné nedostatky a naopak výhody existujúcich riešení zmienených v predchádzajúcej kapitole. Toto zhrnutie je súčasťou motivácie pre vytvorenie popisovaného portálu. Hlavnou nevýhodou všetkých uvedených riešení je nemožnosť interakcie učiteľa so systémom (či už v podobe nahrávania vlastných diktátov alebo vo forme správy žiakov svojej triedy). V druhom rade chýba možnosť akéhokoľvek triedenia, či ukladania výsledkov a ich následnej agregácie do rôznych štatistík (či už jednotlivcov alebo celých školských tried). Posledným vážnym nedostatkom je, že žiak po dokončení trénovania a následnej opravy diktátu vie kde urobil chybu, ale nevie, prečo sa ten ktorý jav píše práve tak ako je uvedené, tzn. chýba popis a vysvetlenie chýb[3]. Všetky uvedené nedostatky sú obsiahnuté v špecifikácii požiadavkov na popisovaný webový portál.

2. Viac na `https://is.muni.cz/auth/el/1433/test/s_zakazky/rozprac15/040_zizkova/diktaty/diktaty.html`, je potrebné mať právo prístupu – zabezpečený zdroj
3. S výnimkou `umimcesky.cz`, ktoré ponúka inú formu trénovania

2.2 Navrhovaný systém

2.2.1 Špecifikácia požiadaviek - definícia

Požiadavkou nazývame neformálnu špecifikáciu toho, čo má byť implementované. Požiadavky rozdeľujeme na funkčné a nefunkčné. Funkčné požiadavky (skrátene FP) určujú, aké chovanie bude navrhovaný systém ponúkať. Nefunkčné požiadavky (NFP) špecifikujú vlastnosti a obmedzujúce podmienky navrhovaného systému. Špecifikácia požiadaviek je základným dokumentom akéhokoľvek požadovaného systému. Požiadavky sú v podstate vyjadrením toho, čo by mal systém robiť, nie však toho, ako by to mal robiť[5].

2.2.2 Vlastná špecifikácia požiadaviek

NFP1 Na vývoj systému je použitá platforma Java EE
Je definovaná konkrétna platforma, s použitím ktorej je portál vyvíjaný, pričom použité technológie v rámci platformy nie sú vopred určené a ich výber je ponechaný na vývojárovi aplikácie a špecifikovaný neskôr v texte.

NFP2 Jednoduchá rozšíriteľnosť
Webový systém na komplexnú prácu s diktátmi by mal byť ľahko rozšíriteľný o novú funkcionalitu, predovšetkým nižšie zmienený samostatný modul (FP8) určený na opravovanie textu napísaného používateľom.

FP1 Používateľské role
V systéme budú vytvorené nasledujúce používateľské role s rôznou úrovňou oprávnení:

- ADMINISTRATOR - používateľská rola, ktorá slúži na správu systému. Jeho hlavnou úlohou je pridávať používateľov role učiteľ (definovaná nižšie v špecifikácii) a priradiť im konkrétnu skupinu používateľov do ich správy.
- TEACHER - rola reprezentujúca učiteľa. Má možnosť pridávať nový diktát, spravovať priradenú skupinu používateľov a zobraziť štatistiky pre pridelenú skupinu.

- STUDENT - má možnosť listovať dostupnými diktátmi, trénovať diktát, nechať si ho opraviť, vidieť vlastné štatistiky a upravovať vlastný profil.

FP2 Správa skupín používateľov
Učiteľovi je pridelená práve jedna skupina používateľov (konkrétna trieda v rámci školy), ktorú môže spravovať a vidieť štatistiky chýb v diktátoch celej skupiny aj jednotlivcov.

FP3 Nahrávanie diktátov do systému
V systéme je možné nahrávať zvukový súbor s diktátom a priradiť k nemu popisné informácie priamo v prostredí portálu.

FP4 Používateľské štatistiky
Každý žiak môže vidieť svoje vlastné štatistiky, učiteľ vidí štatistiky celej skupiny používateľov (triedy v škole, ktorú spravuje) aj každého jednotlivca v skupine.

FP5 Prihlasovanie používateľov pomocou mailu alebo sociálnej siete
Každý používateľ má možnosť sa prihlásiť klasicky pomocou mailovej adresy. Žiak má možnosť sa prihlásiť prostredníctvom sociálnej siete.

FP6 Registrácia používateľov
Používateľ má možnosť sa pri prvom vstupe na portál zaregistrovať vyplnením registračného formulára a následne sa prihlásiť pomocou zvoleného emailu a hesla.

FP7 Písanie diktátov a ich oprava
Žiak si môže napísať diktát a po jeho napísaní si ho nechať systémom automaticky opraviť.

FP8 Samostatný modul integrovaný s webovou aplikáciou určený na opravu diktátov
Webový portál obsahuje samostatný modul slúžiaci na opravu diktátov, ktorý je dostupný cez verejné rozhranie ako služba. Definícia rozhrania aj zdrojový kód budú riadne zdokumentované a pripravené na rozširovanie.

FP9 Implementácia pravidiel nájdených počítačovými lingvistami

V rámci portálu budú implementované pravidlá nájdené počítačovými lingvistami, ktoré budú triediť jednotlivé chyby do kategórií a pridávať zdôvodnenia pre vybrané jazykové javy v češtine.

3 Návrh a implementácia systému

Webový portál je rozdelený do troch častí. Najnižšou vrstvou systému je časť starajúca sa o prístup k dátam, prácu s dátami a ich poskytovanie vyššej vrstve. V angličtine sa nazýva výstižným názvom *back-end*[1]. S vyššou vrstvou *back-end* komunikuje pomocou tzv. REST[2] rozhraní. Nad vrstvou pristupujúcou k dátam stojí tzv. prezentačná vrstva (anglicky front-end), teda vrstva s ktorou priamo komunikuje používateľ systému. Prezentačná vrstva je navrhnutá ako tzv. jednostránková aplikácia (skrátene SPA – Single Page Application[3]).

Poslednou časťou je samostatný modul nasadený spoločne s aplikáciou, slúžiaci na opravu chýb. Takisto ako k *back-endu*, je k nemu možné pristupovať skrz REST rozhranie. Jeho úlohou je vo vstupnom texte od užívateľa nájsť chyby, označiť ich, zozbierať o každej chybe čo najviac informácií a na ich základe potom zaradiť chyby do kategórií (definovaných v prílohe C) a priradiť k ním zodpovedajúci popis. Rozhranie je tiež možné použiť aplikáciami tretích strán ako službu. Štruktúra modulu je do hĺbky rozobratá v kapitole 4 a popis rozhrania vrátane správneho vstupu a výstupu je popísaný v prílohe A.1.

Pri implementácii systému je kladený dôraz na použitie moderných technológií a štruktúra aplikácie je navrhnutá tak, aby sa ktorákoľvek súčasť dala v prípade potreby rýchlo nahradiť inou bez nutnosti zásahu do iných vrstiev systému. V následujúcich sekciách sú postupne uvedené použité technológie rozdelené podľa príslušnosti k vrstve systému.

---

1. Anglický výkladový slovník *Cambridge* definuje výraz back-end ako *časť počítačového programu alebo systému ktorý sa stará o dáta a ku ktorému nemá používateľ priamy prístup*
2. REST - Štýl návrhu softwaru a prístup k riešeniu komunikácie medzi vrstvami systému
3. SPA sú webové aplikácie, ktoré načítajú jediný HTML stránku a potom ju dynamicky obnovujú podľa toho, ako s ňou používateľ komunikuje[6]

3.1 Entity - charakteristika

Entita v kontexte analýzy a návrhu webovej aplikácie reprezentuje jednu konkrétnu tabuľku v databáze, ktorá je na ňu *mapovaná* (Viac o entitách v kapitole 3.3). Navrhovaný webový systém pozostáva z niekoľkých navzájom prepojených entít. Ich zoznam je jedným z výstupov analýzy aplikácie. Sú to `Category`, `Dictate`, `User`, `Trial`, `Error`, `SchoolClass` a `School`. Nižšie je uvedený krátky popis.

- `Category` reprezentuje kategóriu diktátu, teda väčšinou gramatický jav, ktorý má diktát v prvom rade precvičiť. Napr. vybrané slová, veľké a malé písmená, spodobovanie atď. Kategórie definuje správca systému. Poprípade môže kategória určovať, pre koho je diktát určený.

- `Dictate` reprezentuje samotný diktát. Obsahuje základné informácie o diktáte: meno, krátky popis diktátu, názov súboru, pod ktorým je uložený na serveri a referenciu na kategóriu, do ktorej diktát spadá. Keďže jedna z požiadaviek na systém bola možnosť nahrávať diktáty, obsahuje tiež odkaz na učiteľa, ktorý diktát pridal. Entita `Dictate` tiež ukladá dáta popisujúce samotný zvukový súbor s diktátom, menovite časové značky koncov viet, počet opakovaní diktátu a opakovaní jednotlivých viet, pauzy medzi vetami a textový prepis diktátu, podľa ktorého potom bude prebiehať oprava.

- Entita `User` nesie základné informácie o používateľovi. Pod tým sa rozumie časová známka registrácie používateľa, meno, adresa elektronickej pošty (ktorá je unikátna, aby bolo možné sa pomocou nej prihlásiť), heslo v zašifrovanom tvare, typ používateľa a pridelená rola v systéme. Pretože má byť aplikácia rozdelená na používateľskú časť a časť pre učiteľov, delí sa entita na dva podtypy: študent a učiteľ, ktoré sa líšia tým, akého je používateľ typu (STUDENT resp. TEACHER) a aké role môže zastávať (rola ADMINISTRATOR a TEACHER pre učiteľa resp. STUDENT pre žiaka). Každý administrátor systému má zároveň aj rolu učiteľa, opačne to, samozrejme, neplatí.

- Entita `Trial` obsahuje informácie o každom dokončenom diktáte trénovanom žiakom. Pozostáva zo žiakom napísaného neopraveného textu, časovej známky trénovania, odkazu na trénovaný diktát a rovnako odkazu na trénujúceho študenta. Tieto informácie sú použité predovšetkým pri generovaní štatistických dát.

- Entita `Error` reprezentuje jednu chybu používateľa v diktáte. Chyba nesie všetky popisné informácie (ktoré sú totožné objektom `Mistake` z modulu na opravu diktátov, pretože ho k svojmu vytvoreniu využíva). Zároveň obsahuje odkaz na entitu `Trial`, odkaz na diktát, v ktorom sa chyba vyskytla a odkaz na používateľa, ktorý ju urobil. Rovnako ako u predchádzajúcej entity, aj tieto dáta sú predovšetkým určené na účely štatistiky.

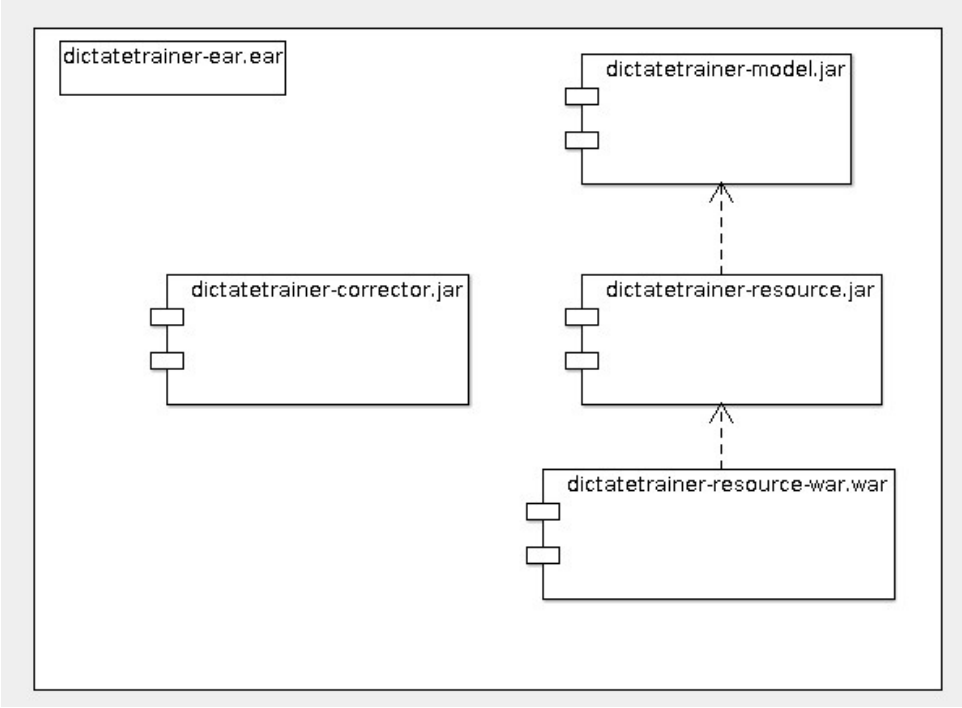
- Entita `School` slúži k ukladaniu jednotlivých škôl v systéme. Je v prvom rade určená na reprezentáciu skupín používateľov, prípadne pre štatistické účely.

- Entita `SchoolClass` reprezentuje konkrétnu školskú triedu v konkrétnej škole. Spolu s entitou `School` tvoria v systéme jedinečné dvojice, do jednej z ktorých musí patriť každý registrovaný žiak a ktorá má práve jedného správcu – učiteľa.

  Dátový diagram reprezentujúci tabuľky v databáze pre lepšiu predstavu interakcie entít je možno nájsť v prílohe D.1.

3.2 Rozdelenie aplikácie na moduly

Aplikácia je rozdelená na 5 modulov. Prvý s názvom `dictatetrainer-model` obsahuje triedy zabezpečujúce prístup k databáze, biznis vrstvu aplikácie a k nim prislúchajúce testy. Modul `dictatetrainer-resource` obsahuje verejne prístupné rozhrania spolu s jednotkovými testami a `dictatetrainer- resource-war` zoskupuje súbory prezentačnej vrstvy aplikácie.`Dictatetrainer-corrector` je zvláštny modul zabezpečujúci opravu diktátu a poskytuje verejne prístupné rozhranie použiteľné ako služba. Posledný modul s názvom `dictatetrainer-ear` zabezpečuje správne zabalenie celej aplikácie, oddeľuje použité knižnice do zvláštneho adresára a *back-end* systému do zvláštneho archívu tak, aby bola štruktúra čo najprehľadnejšia. Pre lepšiu predstavu viď obrázok nižšie.



Obr. 3.1: Štruktúra modulov aplikácie

3.3 Štruktúra aplikácie – vrstva prístupu a správy dát (*Back-end*)

Nasledujúca sekcia sa zaoberá popisom štruktúry aplikácie. Postupne budú podrobnejšie rozobrané jednotlivé vrstvy.

3.3.1 Vrstva prístupu k databáze

Aby bolo možné abstrahovať definíciu tabuliek, relácii medzi nimi a dotazovanie na ne od konkrétnej implementácie databázového systému (ako je napr. PostgreSQL, MySQL, Oracle a pod.), je nutné použiť rámec[4] umožňujúci objektovo-relačné mapovanie (skrátene ORM) tabuliek v databáze na Java objekty, tzv. entity. Entita je trieda typicky reprezentujúca jednu tabuľku v databáze. Obsahuje atribúty, ktoré sú ekvivalentné stĺpcom v tabuľke, bezparametrický konštruktor a, ak chceme definovať aj vzájomné vzťahy medzi entitami, musí obsahovať aj preťažené metódy `equals` a `hashCode`.

Prístup k entitám

Po vytvorení entít je nutné vedieť k nim pristupovať a vykonávať s nimi základné operácie. Základnými operáciami rozumieme tzv. CRUD operácie[5]. Implementácia týchto operácií je definovaná v špeciálnej triede vlastnej pre každú entitu. Tvorí akýsi medzičlánok spájajúci entity a databázu. K základným operáciám je pre väčšinu entít naviac pridaná možnosť filtrovania výsledkov na základe určených atribútov (resp. z databázového pohľadu stĺpcov tabuľky).

3.3.2 Biznis vrstva aplikácie

Servisná, alebo aj biznis vrstva aplikácie vytvára ďalší stupeň abstrakcie aplikácie. Spolupracuje s vrstvou prístupu k dátam a poskytuje svoje rozhranie vrstve nad ňou. Zabezpečuje vytváranie závislostí jednotlivých entít (napr. entita `User` obsahuje referenciu na entitu `SchoolClass` a tá na entitu `School`). Servisná vrstva vytvára

4. angl. framework
5. **C**reate = vytvorenie, **R**ead = čítanie, **U**pdate = aktualizácia, **D**elete = zmazanie

väzby medzi celými objektami, zatiaľ čo na úrovni databázy túto väzbu reprezentuje cudzí kľúč - `id`. Taktiež overuje správnosť vstupných dát pomocou tzv. validátora a stará sa o správne hlásenie výnimiek v prípade chyby.

3.3.3 Vrstva rozhraní - `REST`

Táto vrstva obsahuje kolekciu rozhraní, ku ktorým môže pristupovať napr. prezentačná vrstva aplikácie a získavať od nej, resp. jej posielať dáta. To, akú operáciu vykoná s dátami aplikácia je definovateľné pomocou HTTP metód[6], pomocou špeciálne definovanej adresy je možné určiť, s ktorými dátami sa operácia vykoná a návratový kód hovorí, či bola požiadavka dokončená úspešne.

Vrstva rozhraní pridáva možnosť filtrovania výsledkov na základe definovaných atribútov v adrese (podobne ako sú definované dodatočné atribúty v URL). Taktiež je implementované radenie výsledkov podľa všetkých atribútov entít, v ktorých je dostupné filtrovanie. Dôležitou funkciou je tiež možnosť definovať, koľko výsledkov má systém maximálne vrátiť (tzv. stránkovanie odpovede[7]). Implementácia týchto funkcií veľmi uľahčuje získavanie konkrétnych informácií, pomáha pri budovaní štatistík a znižuje množstvo prenášaných dát na potrebné minimum.

V neposlednom rade je na tejto úrovni definované zabezpečenie rozhraní proti neoprávnenému prístupu, povolenie prístupu klientom z iných domén (tzv. CORS, viac v kapitole 3.5.4) a riadenie prístupu na základe používateľských rolí3.5.

Táto vrstva tiež poskytuje aplikačnú logiku a rozhrania pre nahrávanie súborov s diktátmi a prihlasovanie pomocou sociálnej siete.

3.3.4 Prezentačná vrstva

K prezentačnej vrstve pristupuje používateľ priamo prostredníctvom svojho webového prehliadača. Na jednej strane vrstva komunikuje s používateľom a na strane druhej používa už zmienené `REST` rozhrania na komunikáciu s *back-endom* a získavanie dát z resp. ukladanie

6. Dostupné HTTP metódy v portáli sú `GET`, `POST`, `PUT` a `DELETE`
7. angl. pagination

dát do databázy. Vrstva taktiež implementuje grafické používateľské rozhranie (tzv. GUI[8]).

Návrh grafického používateľského rozhrania

Pri návrhu rozhrania sa kládol dôraz na to, aby bolo používateľské rozhranie čo najviac minimalistické, prehľadné a bez rušivých elementov. Základný návrh tak tvorí iba na navigačná lišta, v ktorej sa každý používateľ ľahko zorientuje a miesto určené obsahu stránky nachádzajúce sa pod ňou (Obrázky sú pre predstavu umiestnené v prílohe B).

8. angl. graphic user interface

3.4 Použité technológie

3.4.1 Java Persistence API a Hibernate

Ako už bolo zmienené v popise štruktúry aplikácie (kapitola 3.3), z lepšej flexibility implementácie prístupu k databáze je vhodné použiť takzvané objektovo-relačné mapovanie (ORM). Štandardný rámec pre programovací jazyk Java slúžiaci na tento účel je Java Persistence API (skrátene JPA)[9]. Existuje viacero dostupných implementácií JPA - ako napr. EclipseLink, OpenJPA alebo Hibernate. Aplikácia používa poslednú spomínanú knižnicu – Hibernate.

3.4.2 Enterprise JavaBeans

Enterprise JavaBeans (skrátene EJB) je softvérová komponenta, ktorá zabezpečuje zapúzdrenie biznis logiky aplikácie. Je súčasťou platformy Java Enterprise Edition. EJB obsahuje webový kontajner, ktorý poskytuje webové komponenty ako napr. správu transakcií, zabezpečenie aplikáci a iné. Portál používa v súčasnosti najnovšiu verziu EJB3.

3.4.3 JAX-RS

Klientské API[10] s názvom `JAX-RS` je rozhranie založené na programovacom jazyku Java, ktoré umožňuje implementovať prístup k webovým službám a poskytuje podporu pri ich vytváraní na základe návrhového vzoru REST[7][11]. Prostredníctvom `JAX-RS` je možné jednoducho definovať prístupové body služieb, riadenie prístupu, formát priložených vstupných a výstupných dát i použitú `HTTP` metódu.

3.4.4 Angular JS JavaScript framework

AngularJS je framework založený na programovacom jazyku JavaScript a implementujúci softvérový návrhový vzor *MV**. Skratka *MV**

9. Je to vlastne sada rozhraní definujúcich ako by malo ORM fungovať
10. API – skratka pre aplikačné rozhranie pre programovanie aplikácií
11. REST – skratka pre *Representational State Transfer* - sprístupnenie webových služieb pomocou `HTTP` volaní

označuje začiatočné písmená anglických výrazov *Model* (reprezentácia dát - pomocou premenných jazyka JavaScript), *View* (vrstva používateľského rozhrania - deklarovaná pomocou HTML a špeciálnych atribútov pridaných AngularJS rámcom). Hviezdičkou sa rozumie mechanizmus, ktorý umožňuje komunikáciu medzi týmito dvoma vrstvami. Nazýva sa *Digest Loop* a pridáva sledovanie zmien atribútov deklarovaných v modeli v jazyku JavaScript na strane používateľského rozhrania[8].

Hlavnou výhodou frameworku AngularJS oproti napr. JSF je, že sa logika prezentačnej vrstvy presunula na stranu klienta. [9] AngularJS bol zvolený aj z viacerých iných dôvodov, uvádzam podľa môjho názoru tie najdôležitejšie.

- Jednoduchá použiteľnosť a prehľadnosť
- Priamočiara integrovateľnosť s REST rozhraním
- Veľká komunita poskytujúca množstvo rozširujúcich modulov pridávajúcich potrebnú funkcionalitu
- Základ prezentačnej vrstvy frameworku tvorí HTML
- Určený na tvorbu jednostránkových aplikácií

Tento framework tvorí logiku prezentačnej vrstvy aplikácie, tzn. reprezentuje komunikáciu s *back-endom* a prácu s dátami.

Používané moduly vrámci AngularJS

V rámci implementácie portálu sú použité aj viaceré externé moduly rozširujúce rámec o nové funkcie. Sú to:

- `Satellizer` na prihlasovanie pomocou sociálnych sietí na úrovni prezentačnej vrstvy[12]
- Modul `ng-audio` na jednoduché prehrávanie zvukových súborov a aplikáciu základných operácií s nimi (ako napr. opakovanie prehrávania)[13]

12. Viac na: `https://github.com/sahat/satellizer`
13. Viac na: `https://github.com/danielstern/ngAudio`)

- Modul `ng-file-upload` na implementáciu nahrávania súborov vrámci prezentačnej vrstvy[14]

3.4.5 Aplikačný rámec Bootstrap

Aplikačný rámec Bootstrap je voľne šíriteľný aplikačný rámec založený na kaskádových štýloch (CSS). Je možné ho použiť k návrhu kompletného používateľského rozhrania [15]. Je to kolekcia nástrojov slúžiacich k vytváraniu webových aplikácií [10] obsahujúca tlačidlá, formuláre, tabuľky, navigačné prvky a podobne. Pomocou HTML atribútov (predovšetkým atribútov `class` a `id`) je možné jednoduchým spôsobom definovať vzhľad jednotlivých komponent na webovej stránke. V popisovanom portáli je takýmto spôsobom navrhnuté takmer celé grafické rozhranie.

3.4.6 Verzovací systém GIT

Verzovacie systémy (skrátene VCS) zaznamenávajú zmeny v súboroch v čase. K týmto zmenám má používateľ kedykoľvek prístup. Centralizovaný verzovací systém umožňuje zdieľanie zmien medzi niekoľkými používateľmi a spoluprácu na ich vytváraní. Jedným z hlavných dôvodov jeho použitia je záloha dát[**?**].

Hlavnou prednosťou systému GIT oproti iným verzovacím systémom je zavedenie tzv. lokálnych zmien prítomných výhradne na počítači používateľa, ktoré je neskôr možné nahrať na vzdialený server a zdieľať s inými používateľmi.

Portál na písanie a vyhodnocovanie diktátov je dostupný na serveri `GitHub`, ktorý poskytuje verejným projektom vzdialené úložisko zadarmo. Projekt je k nahliadnutiu na adrese `https://github.com/xrumanov/dictatetrainer/`.

14. Viac na: `https://github.com/danialfarid/ng-file-upload`
15. Viac na: `http://getbootstrap.com/`

3.4.7 Správa softvérového projektu pomocou nástroja Maven

Nástroj `Apache Maven` slúži na správu softvérového projektu a pridružených externých knižníc. Prostredníctvom `POM` súboru[16] je možné spravovať zostavovanie projektu, jeho štruktúrovanie, závislosti na iných projektoch a jeho dokumentáciu. Portál používa nástroj `Maven` vo vrstve prístupu a správy dát (*back-end*) na zostavovanie aplikácie, definíciu jej štruktúry a správu závislostí.

3.4.8 Správa balíčkov pomocou nástroja Bower

Nástroj `Bower` je používaný na správu balíčkov a optimalizovaný pre použitie v prezentačnej vrstve (*front-end*). Zabezpečuje, aby všetky použité balíčky boli pri inštalácii a nasadení webovej aplikácie na server v najnovších verziách vrátane závislostí[12]. Pri pridaní nového balíčka do aplikácie Bower jednoducho pridá jeho definíciu do špeciálneho súboru vo formáte JSON[17]. (podobne ako `POM` súbor v nástroji `Maven` spomínanom vyššie). Pri inštalácii potom iba stiahne všetky definované balíčky z internetu. Navrhovaný portál používa Bower na získavanie modulov pre rámec AngularJS a CSS rámec Bootstrap.

16. POM – skratka pre *project object model* - definícia softvérového projektu vo formáte *XML*
17. JSON – skratka z anglického JavaScript Object notation – je jednoduchý formát na výmenu dát medzi klientami

3.5 Navrhnuté riešenia vybraných problémov

3.5.1 Správa používateľov a skupín v systéme

Jednou z funkčných požiadaviek na aplikáciu (kapitola 2.2.2, požiadavka FP2) bola implementácia správy používateľov systému. Všetkých používateľov systému spravuje používateľ role `ADMINISTRATOR`, ktorý je po nasadení aplikácie na server ručne pridaný do databázy. Má povolené upravovať alebo zmazať akéhokoľvek používateľa systému (s výnimkou seba samého). Jeho druhou úlohou je na požiadanie vytvoriť používateľa role `TEACHER` (učiteľ), pretože toho nie je možné registrovať priamo cez grafické rozhranie. V rámci registrácie nového učiteľa je administrátor taktiež poverený prideľovať skupiny používateľov konkrétnemu učiteľovi.

Každý učiteľ v systéme má pridelených práve jednu školskú triedu v systéme. Tú vytvára zároveň s novým učiteľom administrátor systému. Školská trieda je definovaná atribútmi názov triedy (napr. 4.A, alebo Kvinta) a názov školy (napr. Gymnázium Ľ. Štúra, Trenčín). Táto dvojica atribútov je vrámci systému unikátna. V aplikácii momentálne nemôžu existovať viacerí učitelia spravujúci rovnakú skupinu používateľov. Používateľ role `STUDENT` (žiak) patrí vždy do práve jednej školskej triedy. Učiteľ môže upravovať profil žiakov v pridelených skupinách takisto ako ich mazať. Rovnako má možnosť vidieť štatistiky každého žiaka jednotlivo, alebo celej triedy.

3.5.2 Autentizácia pomocou sociálnych sietí

Existuje hneď niekoľko možností, ako vyriešiť prihlasovanie do webovej aplikácie pomocou sociálnych sietí, píše Carvajal[13] na svojom technologickom blogu. Popis technológie a porovnanie jednotlivých možností jej implementácie takisto pochádza z uvedeného zdroja.

Autorizačný protokol OAuth

`OAuth` je autorizačný protokol používaný v prípade potreby prístupu webovej služby `A` k informáciám webovej služby `B` (ako je napríklad sociálna sieť). Protokol vznikol, aby používateľ nemusel do služby **A** ručne zadávať prihlasovacie údaje služby **B**. Aby vývojári nemuseli

pre každú službu, od ktorej požadovali informácie integrovať nové autorizačné schémy, vznikol štandardný protokol. OAuth, dnes už vo svojej druhej verzii, nie je autentizačný[18] protokol, ale protokol slúžiaci na autorizáciu[19]. Nižšie je uvedené ako je možné OAuth použiť aj na autentizáciu používateľa.

Možnosti integrovania sociálnej autentizácie

Carvajal na svojom blogu rozoberá tri autentizačné schémy.

Prvá možnosť je nechať používateľa prihlásiť sa k aplikácii pomocou zvolenej sociálnej siete. Používateľ prihlásením povolí prístup k niektorým jeho informáciám vrátane tzv. identifikátora (skrátene *id*) vrámci sociálnej siete. Aplikácia potom vygeneruje unikátny identifikátor používateľa na základe získaného id zo sociálnej siete a prípadne iných informácií o používateľovi. Heslo je generované pomocou hashovacej funkcie na základe identifikátora aplikácie v sociálnej sieti (tzv. `client id` aplikácie), tajomstva aplikácie a získaných používateľských informácií. Najdôležitejšie pri tejto autentizačnej metóde je, aby nebolo zverejnené id a tajomstvo aplikácie resp. hashovacia funkcia. Výhody uvedeného prístupu spočívajú v tom, že používateľ si nepotrebuje pamätať ďalšie heslo, riešenie je relatívne jednoducho implementovateľné a tiež, že heslo (resp. v tomto prípade tajomstvo) môže byť uložené v aplikácii alebo vygenerované znova na základe získaných informácií. Nevýhoda je, že tajomstvo je rovnako jednoducho generovateľné treťou stranou v prípade znalosti postupu.

Druhú variantu sociálnej autentizácie je možno nazvať hybridnou, pretože používa `OAuth` protokol na získanie používateľských informácií zo sociálnej siete a heslo zadáva používateľ ručne, rovnako ako pri prihlasovaní emailom. Tento prístup je iba o málo lepší ako klasické prihlasovanie pomocou mena a hesla, konkrétne o to, že používateľ nemusí ručne zadávať svoje osobné informácie ako napr. meno a email. Výhodou oproti predchádzajúcemu riešeniu je, že je vyžadované explicitné overenie identity, postup sa neuspokojí iba s prístupom k sociálnej sieti. Nesporná výhoda je tiež jednoduchosť

18. Autentizácia = identifikácia používateľa na základe mena a hesla
19. Autorizácia = povolenie prístupu k určitým informáciám o používateľovi

implementácie a možnosť expanzie z klasickej prihlasovacej schémy (`BasicAuth`). Nevýhodou je nutnosť pamätať si ďalšie heslo.

Posledným zmieneným autentizačným postupom je tzv. Reverzná `OAuth` autentifikácia. Od sociálnej siete je získaný prístupový token, ktorý je následne odoslaný aplikácii cez jej REST rozhranie. Server aplikácie sa následne spojí s autorizačným serverom sociálnej siete a overí používateľské údaje za použitia prístupového tokenu. Hlavné výhody: Nie je potrebné ďalšie heslo. Nie je nutné ukladať heslá alebo prístupové tokeny. Používateľsky prívetivé. Z vývojárskeho pohľadu však veľmi ťažko implementovateľné a neintuitívne[20].

Zvolený prístup pre Diktátový systém

Pre navrhovaný portál bola zvolená hybridná autentizácia pomocou sociálnych sietí. Dôvodov je hneď niekoľko. Prvým dôvodom sú problémy v návrhu systému popisované v kapitole 3.6.2. Taktiež bol tento prístup zvolený preto, že používateľ (žiak) musí spolu s heslom zadávať pri prvom prihlasovaní i školu a triedu, pretože sa tieto informácie zo sociálnych sietí získať nedajú. Prihlasovanie pomocou sociálnych sietí prebieha nasledovne:

1. Používateľ klikne na ikonu prihlasovania cez zvolenú sociálnu sieť
2. Ak ešte nie je prihlásený k zvolenej sociálnej sieti, táto ho požiada o prihlásenie
3. V prípade úspechu pošle sociálna sieť aplikácii požadované informácie od používateľa (meno a email)
4. Aplikácia overí, či používateľ so získaným emailom v systéme existuje
   (a) Ak áno, spýta sa na heslo do aplikácie ktoré potom používateľ zadá
   (b) Ak nie, požiada o voľbu hesla, triedu a školu žiaka

20. Viac ohľadom implementácie na `http://digitalleaves.com/blog`

Ďalšou výhodou tohto riešenia je, že zjednocuje prihlasovanie a používateľ sa po prvom prihlásení cez sociálnu sieť môže prihlásiť klasickým spôsobom email/heslo, pokiaľ použije svoj email zo sociálnej siete.

Implementácia sociálnej autentizácie pozostávala z dvoch častí. Na strane servera je vytvorené REST rozhranie pre každú sociálnu sieť zvlášť. Rozhranie reprezentuje logiku autentizácie popísanej v predchádzajúcej sekcii. Na strane užívateľského rozhrania je použitý modul frameworku AngularJS s názvom *Satellizer*. Je to autentizačný modul so vstavanou podporou pre Facebook, Google, LinkedIn a iné sociálne siete[14].

3.5.3 Zabezpečená komunikácia pomocou HTTPS

Základom bezpečnosti akéhokoľvek webového systému je zabezpečená komunikácia. Akékoľvek dáta vrátane mena, hesla a iných citlivých informácií, posielané používateľom cez sieť pri interakcii so systémom, by bez boli bez zabezpečenia vystavené potenciálnemu riziku odcudzenia. Útok vykonávaný za takýmto účelom sa nazýva *Man in the middle attack*[21]. Je tomu však možné zabrániť zašifrovaním komunikácie použitím protokolu `HTTPS`, ktorý je bezpečnou verziou `HTTP`. Cezeň sa posielajú dáta medzi internetovým prehliadačom a webovou stránkou, ku ktorej je prehliadač pripojený.

HTTPS typicky využíva jeden z dvoch protokolov na šifrovanie komunikácie - TLS alebo SSL. Oba používajú tzv. asymetrickú šifru, ktorá pracuje s dvoma typmi kľúčov. Privátny kľúč je bezpečne uložený na webovom serveri a zabezpečuje šifrovanie posielaných dát. Verejný kľúč je distribuovaný komukoľvek, kto chce rozšifrovať informácie zašifrované pomocou privátneho kľúča. Verejný kľúč obdrží používateľ v SSL certifikáte webovej stránky pri prvotnom požiadavku o spojenie [15].

Vo vyvíjanej aplikácii, ktorá je nasadená na serveri používajúcom službu `Openshift`, je vynútený protokol HTTPS. Openshift ponúka svoj SSL certifikát, takže nie je nutné vytvárať vlastný. Je iba potrebné správne nastaviť `web.xml` a `jboss-web.xml` podľa dokumentácie[16].

21. Typ útoku, kedy útočník napadne komunikáciu medzi užívateľom a serverom s cieľom ukradnúť citlivé dáta

3.5.4 Cross Domain Resource Sharing (CORS)

Jednou z požiadaviek na aplikáciu (kap. 2.2.2, FP8) je vytvoriť rozhranie prístupné ostatným doménam, ktoré bude schopné opraviť používateľský text podľa zadaného originálu. Túto funkciu bude môcť v budúcnosti využívať napr. Informačný systém Masarykovej univerzity v svojom projekte webového portálu na trénovanie diktátov [22]. Keďže ide o prístup k dátam a zdrojom aplikácie zvonku, je nutné riešiť tzv. Cross Domain Resource Sharing[23].

CORS HTTP požiadavka zdrojovej stránky je definovaná ako požiadavka na dáta alebo informácie z inej domény ako je doména zdroja. Dáta môžu byť v akejkoľvek forme, napr. obrázky, css štýly, skripty atď. Z bezpečnostných dôvodov sú CORS požiadavky vo všetkých webových prehliadačoch implicitne zakázané. Explicitne je však možné prístup pre určité rozhrania a domény povoliť. Ku konkrétnej odpovedi servera sa pridajú hlavičky ktoré popisujú množinu domén a HTTP metód, ktorým je povolený prístup. HTTP metódy, ktoré môžu spôsobiť vedľajšie efekty na odosielané používateľské dáta (predovšetkým GET a POST), je najprv vykonaná tzv. predpožiadavka[24], ktorý pozdrží odoslanie odpovede servera do prehliadača až do okamihu, keď je povolený prístup na základe hlavičiek[17]. Správny formát hlavičiek znižuje bezpečnostné riziká spojené s CORS, predovšetkým CSRF útok[25][18]. Definícia hlavičiek spolu s dokumentáciou sa nachádza na stránkach pracovnej skupiny W3C[19].

Povolenie prístupu musí byť definovaná na strane servera. Z návrhu aplikácie vyplýva, že je nutné povoliť rozhranie slúžiace na opravu diktátov, využívajúce metódu POST. Java ponúka hneď niekoľko možností.

Prvá možnosť je použitie filtru, ktorý implementuje rozhranie `ContainerResponseFilter`[20][21]. Tento spôsob však povoľuje prístup pre všetky zdroje, preto je pre aplikáciu nevhodný. Druhé riešenie je pridať prístupové hlavičky priamo pri budovaní odpovede

22. Viac na: `https://is.muni.cz/auth/el/1433/test/s_zakazky/rozprac15/040_zizkova/diktaty/diktaty.html,zabezpečenýzdroj`
23. V preklade: Zdieľanie zdrojov naprieč doménami
24. angl. Preflight request
25. Cross-Site Request Forgery, v preklade: Falšovanie požiadaviek cudzími doménami je typ útoku, ktorý vykoná požiadavku z inej domény ako doména servera využívajúc aktuálne prihláseného používateľa

spolu s telom odpovede a návratovým kódom. Výhoda tohto riešenia je, že umožňuje vybrať podmnožinu dostupných zdrojov u ktorých bude povolený CORS. Bohužiaľ, tento postup nie je možné použiť pre HTTP metódu POST, pretože pokiaľ je požiadavka odoslaná z inej domény, hlavičky sa nepridajú a odpoveď skončí s chybou[22].

Riešenie spĺňajúce obmedzenia kladené na aplikáciu využíva externú knižnicu[23], ktorá definuje prístupové hlavičky priamo v súbore `web.xml`. Tento postup jednak ponúka možnosť vybrať z dostupných zdrojov tie, ktoré budú povolené, a taktiež funguje s HTTP metódou POST[22].

3.5.5 Ukladanie dát popisujúcich diktát

Dátami popisujúcimi diktát rozumieme všetky dáta uložené spolu s diktátom, ktoré ho nejakým spôsobom definujú. Konkrétne sa jedná o textový prepis diktátu (tzv. transkript), značky koncov jednotlivých viet, východzí počet opakovaní prehrávania viet ako aj celého diktátu a dĺžka pauzy medzi jednotlivými vetami. V mojej bakalárskej práci bol systém navrhnutý tak, že boli tieto informácie uložené v rámci mp3 súboru s diktátom [24]. Tento prístup má však niekoľko nevýhod. Jednak je použitie zvukových formátov obmedzený na `mp3`, pretože sa informácie ukladajú do `id3` tagu[26]. Druhou nevýhodou je, že tento návrh komplikuje prípadnú budúcu úpravu systému a spracovanie napr. pre účely štatistiky, pretože v prípade budovania štatistických informácií by bolo nutné najprv pristúpiť ku každému súboru s diktátom zvlášť a získať z neho potrebné dáta. V neposlednom rade je nevýhodou chýbajúca flexibilita takéhoto riešenia. Každý súbor s diktátom by totiž pred nahratím na server musel do súboru nahrať všetky dáta pomocou špeciálneho nástroja implementovaného v systéme.

Toto všetko sú dôvody, prečo bolo nakoniec rozhodnuté ukladať všetky popisné dáta v databáze spolu s autorom a názvom súboru s diktátom a zvukový záznam uložiť zvlášť na serveri.

Existujú dve možnosti ako ukladať zvukové súbory a všeobecne akékoľvek dáta. Prvý prístup ukladá dáta priamo do databázy ako

26. Do súboru mp3 je možné uložiť rôzne atribúty, nazývané tagy, ktoré obsahujú napr. názov interpreta, názov skladby, rok vydania atď.

BLOB[27]. Nevýhoda tejto alternatívy je v tom, že neúmerne zväčšuje databázu, spomaľuje dotazy a všeobecne degraduje jej rýchlosť.Druhá možnosť je ukladať súbory priamo na server a do databázy pridať iba odkaz, resp. meno súboru ako jeden stĺpec v tabuľke. Tento prístup je pri veľkých súboroch odporúčaný [25]. Rozhodol som sa preto oddeliť súbory s diktátmi od databázy a ponechať iba referenciu na súbor. Viac o konkrétnom riešení s použitím databázy Postgres a aplikačného servera Wildfly je uvedené v kapitole 6. Nasadenie systému.

3.5.6 Nahrávanie diktátov

Nahrávanie diktátov učiteľom pomocou grafického užívateľského rozhrania a pridávať k nim pomocné dáta, ktoré budú diktát popisovať (ako počet opakovaní jednotlivých viet, počet opakovaní celého diktátu, dĺžka pauzy medzi vetami...), bolo jednou z hlavných požiadaviek na aplikáciu2.2.2. Ako je spomenuté v kapitole 3.5.5, samotný súbor s diktátom je uložený oddelene od metadát.

Návrh nahrávania diktátov bol rozdelený na tri etapy:

1. Sprístupnenie zložky určenej pre ukladanie diktátov na aplikačnom serveri
2. vytvorenie REST rozhrania pristupujúcemu k tejto zložke
3. navrhnutie nahrávania diktátu v prezentačnej vrstve na strane klienta

Návrh počíta s existenciou zložky na serveri určenej pre nahrávanie diktátov, ktorá bude umiestnená mimo samotný kontext aplikácie. Hlavným dôvodom je nutnosť zachovania nahratých súborov v prípade opätovného nasadenia aplikácie kvôli vylepšeniam resp. oprave chýb.

Nahrávanie diktátov na strane rozhrania

K implementácii nahrávania diktátu na strane používateľského rozhrania bol použitý modul pre framework AngularJS s názvom *ng-*

27. dátový typ pre bližšie nešpecifikované binárne dáta v databáze

file-upload. Medzi jeho hlavné funkcie patrí ukazovateľ postupu nahrávania, možnosť nahrať súbor presunutím na zvolené miesto na stránke (tzv. *drag and drop* funkcia), možnosť filtrovania zvolených typov súborov podľa koncovky a maximálnej veľkosti súboru, jednoduché prepojenie s REST rozhraním na strane servera a podpora starších prehliadačov[26].

V portáli pre písanie a vyhodnocovanie diktátov bolo použité filtrovanie na zvukové súbory `mp3`, `ogg` a `wav` a funkcia *drag and drop*.

3.5.7 Prehrávanie diktátov

Výber knižníc, modulov a rôznych iných možností použiteľných k implementácii prehrávania zvukového súboru v prehliadači je veľký. Funkcie, ktoré musel prehrávač obsahovať boli nasledovné: implementácia požadovaných funkcií musí byť jednoduchá a intuitívna. Mal by obsahovať funkcie umožňujúce opakovanie prehrávania a počiatočný resp. konečný čas prehrávania definovateľný pomocou atribútov. Jediná knižnica, ktorá spĺňa všetky požiadavky je `wavesurfer`, ktorý prináša aj mnoho dodatočných zaujímavých funkcií ako napr. automatická detekcia medzery medzi vetami, zobrazenie zvukového grafu a podobne. Je to však bohužiaľ na úkor jednoduchosti. Z tohto dôvodu bolo rozhodnuté v súčasnej dobe neimplementovať niektoré funkcie a rozšíriť tak okruh potenciálne použiteľných knižníc. Konkrétne v prezentačnej vrstve zatiaľ nie je implementovaná funkcionalita opakovania jednotlivých viet podľa parametrov uložených v databáze. V budúcnosti je však plánovaná implementácia hlavne kvôli tomu, že sa s ňou na úrovni *back-end* vrstvy počíta. Na nasadenú aplikáciu to nebude mať žiaden vplyv, pretože dáta o dĺžke medzier medzi vetami, počte opakovaní viet ako aj informácie o koncoch jednotlivých viet je možné editovať priamo z používateľského prostredia. Momentálne je pri všetkých nových diktátoch nahratých cez používateľské rozhranie implicitne nastavená dĺžka pauzy medzi vetami na `0`, konce viet `0` a počet opakovaní jednotlivých viet `1`.

Diktát a pravidlá pri diktovaní

Diktát je podľa Prášilovej[4] jeden z mnohých druhov pravopisných cvičení. Je to veľmi špecifická možnosť, ako skontrolovať u žiaka

úroveň jeho pravopisného prejavu. Je to podľa nej však aj cvičebný prostriedok, pretože sa z jeho pomocou upevňuje prebraté učivo.

Diktát je vo väčšine prípadov diktovaný učiteľom. Toto diktovanie má taktiež svoje zásady a predpisy. Podľa Hausera[11] sa riadi nasledovnými pravidlami:

1. Učiteľ najprv prečíta celý text (a podľa potreby vysvetlí menej užívané výrazy)
2. Diktuje po vetách. Najprv prečíta celú vetu a potom diktuje po častiach. Tempo žiakov prispôsobí veku žiakov a ich výkonnosti
3. Diktuje presne a zreteľne, dodržuje pravidlá spisovnej výslovnosti, nezdôrazňuje nadmerne pauzy ani `i/y` `s/z` atď.
4. Pri diktovaní sleduje učiteľ ako žiaci píšu. Stojí pred žiakmi a neprechádza sa po triede.
5. Nakoniec prečíta učiteľ ešte raz pomaly celý text.

Vybrané pravidlá, ktoré dávajú zmysel aj pri trénovaní diktátu prostredníctvom počítača, sú jednoducho uplatniteľné a implementovateľné v systéme. Viac v kapitole 3.5.7 Návrh a implementácia prehrávania diktátov.

3.5.8 Basic autentizácia a AngularJS

Podľa špecifikácie aplikácie uvedenej v kapitole 2.2.2 má navrhnutý systém obsahovať administračnú časť pre pedagógov a používateľskú časť pre žiakov. Z tohoto dôvodu je nutné zabezpečiť riadenie prístupu k zdrojom. Ako riešenie je použitá autentizačná metóda `BasicAuth`.

Podľa vývojárskeho portálu `spaghetti.io`[27] je Basic autentizácia jedna z najrozšírenejších a najviac používaných autentizačných metód. Aby bolo možné implementovať túto funkcionalitu, musí byť na strane klienta do každej požiadavky pridaná hlavička obsahujúca Base64[28] reťazec vo formáte `Authorization:Basic username:password`.

28. Číslovací systém založený na 64 rôznych znakoch využívajúci veľmi jednoduchý kódovací/dekódovací algoritmus. Neponúka jednosmernú šifrovaciu funkciu ako napr. SHA1

Napriek tomu, že sa meno a heslo nachádza v nezašifrovanej podobe v hlavičke, je protokol bezpečný, vyžaduje však zabezpečenú komunikáciu medzi klientom a serverom, teda HTTPS (implementácia je popísaná v kapitole 3.5.3).

Implementácia Basic autentizácie na strane klienta vychádza z návodu od Watmora[28]. Funguje to tak, že sa najprv odošle požiadavka na konkrétny zdroj prostredníctvom konkrétnej HTTP metódy. Pri odpovedi prezentačná vrstva overí, či požiadavka skončila úspešne, ak nie zobrazí chybové hlásenie. Ak je dotaz úspešný, uložia sa informácie jednak do globálnej premennej, aby boli prístupné ostatným stránkam resp. im prislúchajúcim skriptom a jednak do HTTP cookie[29]. Toto riešenie bolo zvolené z toho dôvodu, aby sa pri náhodnom alebo cielenom obnovení stránky nestratili prihlasovacie dáta a užívateľ sa nemusel znova prihlasovať do systému. Odhlásenie používateľa znamená presmerovanie na prihlasovaciu stránku a zmazanie HTTP cookie s prihlasovacími údajmi.

Som si vedomý, že toto riešenie nie je ideálne. V budúcnosti je v pláne prejsť na autentizáciu pomocou tokenu (viac v kapitole 3.6.2).

3.5.9 Zabezpečenie ciest na úrovni AngularJS frameworku

Takisto ako je potrebné zabezpečiť aplikáciu proti neoprávnenému prístupu k dátam prostredníctvom rozhraní, je to nutné urobiť aj na úrovni prezentačnej vrstvy i keď tá je jednoduchšie napadnuteľná a podvrhnuteľná (preto dvojité zabezpečenie na oboch vrstvách aplikácie). Momentálne riešenie funguje nasledovne: ak sa používateľ, vedome či nevedome, pokúša dostať na stránku ku ktorej nemá prístup, je automaticky odhlásený a presmerovaný na stránku s prihlásením.

3.6 Problémy a slepé vetvy pri návrhu systému

Pri návrhu systému sa vyskytlo niekoľko vývojových vetiev častí systému, ktoré sa v neskorých fázach návrhu, alebo až počas implementácie ukázali ako nepoužiteľné pre navrhovaný portál. Dôvodov bolo niekoľko - napr. nedostatok informácií, či nedostatočná znalosť

29. dáta získané od webovej stránky, uložené v prehliadači používateľa

technológií pred návrhom a implementáciou. Tieto vývojové vetvy bolo potom nutné nahradiť inými a použiť iné riešenia. Celý proces je u najdôležitejších problémov popísaný nižšie, vrátane technológie alebo riešenia, ktoré boli napokon použité.

3.6.1 Java Server Faces

Z hľadiska technológií v prezentačnej vrstve aplikácie (viď štruktúra aplikácie – kapitola 3.3) sa najprv uvažovalo o použití niektorej s implementácií špecifikácie Java Server Faces (skrátene JSF[30]. JSF je *framework*[31] pre programovací jazyk Java, ktorý zjednodušuje tvorbu webových užívateľských rozhraní za pomoci znovupoužiteľných komponent. Tie sú uložené na strane servera v pamäti a v prípade požiadavky od používateľa vytvoria kompletný vzhľad stránky, ktorý je potom odoslaný klientovi. To si vyžaduje jednu vrstvu abstrakcie naviac[9].

Vzhľadom na konečnú verziu návrhu prezentačnej vrstvy (viac v kapitole 3.3.4), ktorý sa snaží byť minimalistický, považujem znovupoužiteľné komponenty a budovanie ich stromu na strane servera za zbytočnú záťaž a zneprehľadnenie aplikácie. Spolu s tým prináša JSF zvýšenú réžiu v podobe pridania ďalšej vrstvy abstrakcie. Preto bola nakoniec táto technológia opustená v prospech frameworku AngularJS.

3.6.2 JSON Web Token (JWT)

Počas implementácie funkčnej požiadavky 5 – Prihlasovanie pomocou sociálnych sietí sa objavila možnosť implementovať prihlasovanie iným spôsobom – pomocou autentizácie na základe tzv. tokenu[32].

---

30. Viac o jednotlivých implementáciách Java Server Faces frameworku: `http://www.masterthebo ss.com/jboss-web/richfaces/primefaces-vs-richfaces-vs-icefaces`

31. Softwarový framework (slovensky rámec alebo rozhranie) je univerzálne, znovupoužiteľné prostredie upraviteľné pomocou dodatočného kódu, poskytujúce konkrétnu funkcionalitu. Môže obsahovať prídavné programy na zjednodušenie použitia.

32. Token je v kontexte autentizácie a bezpečnosti zariadenie alebo softvérový kľúč slúžiaci na preukázanie identity prihlasovaného používateľa.

Nižšie je krátke porovnanie s klasickou autentizáciou a sú uvedené výhody a nevýhody jednotlivých prístupov.

Klasická autentizácia prebieha tak, že sa používateľ prihlási na server a ako odpoveď dostane tzv. buď priamo údaje o používateľovi, ktoré sa potom uložia do cookie[33], aby s nimi aplikácia mohla pracovať (tento postup sa však s bezpečnostného hľadiska neodporúča), alebo je odpoveď'ou cookie s identifikátorom sedenia (angl. session). Problém sedenia je v prvom rade jeho neuniverzálnosť, pokiaľ chceme použiť viac spôsobov prihlasovania. Riešením je tzv. autentizácia založená na báze tokenu[29].

JSON Web token (skrátene JWT) je autentizačný token reprezentovaný pomocou JSON objektu[34]. Token pozostáva z troch podreťazcov zakódovaných pomocou kódovania *Base64*[35] a oddelených od seba znakom ".". Prvý podreťazec nesie informácie definujúce samoté JWT (tzv. dáta o dátach alebo metadáta), druhá časť tokenu obsahuje samotné dáta a posledná časť obsahuje tzv. tajomstvo. Overenie je možné vykonať jednoducho dekódovaním dát na strane servera. Token je uložený lokálne v prehliadači klienta, tzn. nie je potrebné udržiavať sedenie pre každého používateľa na strane servera. Overenie prihlásenia je možné jednoduchou kontrolou, či existuje token v lokálnom úložisku prehliadača[30]. Ďalšia nesporná výhoda spočíva v univerzálnosti riešenia pokiaľ používame viacero prihlasovacích metód (ako napr. sociálne siete).

Koncept je kvôli jeho nesporným výhodám určený na neskoršiu implementáciu, ktorá tak v budúcnosti nahradí v súčasnosti použitú autentizačnú metódu *Basic Auth* (Viac v kapitole 3.5.8). V momentálnom návrhu bolo nutné zvoliť iný návrh riešenia prihlasovania pomocou sociálnych sietí (Viac v kapitole 3.5.2).

---

33. Cookie (slovensky koláčik) je malé množstvo dát posielaných z webovej stránky a uložené v prehliadači
34. JSON je formát ukladania dát, v súčasnosti veľmi rozšírený, dá sa povedať jednoduchšia verzia formátu XML
35. Kódovanie ktoré prevádza binárne dáta (v tomto prípade oktety UTF-8) na postupnosť tlačiteľných znakov

3.6.3 Atribúty popis chyby a typ chyby v samostatnej tabuľke

Ďalším dôležitým problémom určeným na zmenu implementácie v blízkej budúcnosti je ukladanie atribútov `errorType` a `errorDescription` do samostatných tabuliek. Toto riešenie by zamedzilo duplikácii informácií a tým aj plytvaniu diskovým priestorom. V súčasnej podobe oba atribúty ostali vrámci entity `Error`.

3.6.4 Detekcia hraníc jednotlivých viet vrámci diktátu

Hranica dvoch viet v zvukovom súbore s diktátom je definovaná ako dva súvislé zvukové signály vysokej intenzity oddelené od seba súvislým zvukovým signálom nízkej intenzity[31]. Čo znamená súvislý resp. aká je definícia vysokej a nízkej intenzity signálu je možné zvoliť pomocou konkrétnych atribútov. Detekciu je možné vykonať automaticky pomocou rozšírenia s názvom *wavesurfer* pre jazyk JavaScript.

3.6.5 Integračné testy

Na rozdiel od jednotkových testov, ktoré testujú každú časť systému zvlášť, sa integračné testy zaoberajú tým, ako dobre fungujú jednotlivé časti spolu[32]. Napriek tomu, že v aplikácii je pripravený modul v budúcnosti slúžiaci tomuto účelu, nakoniec bolo rozhodnuté systém testovať výhradne jednotkovými testami. Viac v kapitole 5.

3.6.6 Návrhové chyby zistené pri implementácii

Počas ranej fáze implementácie funkčnej požiadavky 2 – práca so skupinami používateľov bolo nutné z dôvodu nevhodnosti pôvodného navrhovaného riešenia znova vykonať návrh. Pôvodne sa počítalo s ukladaním číselného atribútu označujúceho príslušnosť používateľa ku konkrétnej skupine. Ukázalo sa však, ako nedostatočné z hľadiska používateľského rozhrania, konkrétne vytvárania nového používateľa (volenie čísla skupiny považujem za používateľsky neakceptovateľné). Preto boli dodatočne vytvorené dve nové entity – `School`, reprezentujúca školu a `SchoolClass` obsahujúca referenciu na školu a triedneho učiteľa (správcu skupiny). Viac o jednotlivých entitách v kapitole 3.1.

Druhý problém sa vyskytol v ranej fáze implementácie funkčnej požiadavky 4 – používateľské štatistiky. Pôvodný návrh nepočítal s ukladaním chýb v databáze, to sa ale ukázalo ako nevhodné, preto bola navrhnutá entita `Error` nesúca atribúty reprezentujúce konkrétnu chybu (viac v kapitole 3.1 a v definícii modulu na opravu diktátov, kapitola 4).

4 Modul na opravu diktátov

4.1 Návrh a implementácia

Modul na opravu diktátov je nezávislý od ostatných modulov aplikácie a pozostáva zo značkovača vstupného používateľského textu a definície pravidiel. Prístup k modulu zabezpečuje verejné REST rozhranie. Vstupom je správny text diktátu z databázy a text zadaný používateľom. Výstup vo formáte JSON obsahuje celkový počet chýb a list jednotlivých chýb zoradených podľa pozície chyby v diktáte (viac o definícii rozhrania, formáte vstupu a výstupu v prílohe A.1).

Opravu diktátu možno rozdeliť do niekoľkých fáz. Najprv sa vstupné reťazce porovnajú za pomoci knižnice `diff-match-patch`[33] a vytvorí sa označkovaný text. Z takto predspracovaného textu sa v ďalšom kroku vygeneruje pre každú chybu objekt typu `Mistake`. Tento objekt je určený na uchovanie dát popisujúcich jednotlivé chyby. Štruktúra atribútov vychádza z návrhu popísaného v bakalárskej práci Vojtěcha Škvařila [34]. V nasledujúcom texte bude ako správny označený znak alebo slovo vyskytujúce sa v správnom texte diktátu z databázy a chybným znakom alebo slovom budeme rozumieť podreťazec textu vloženého používateľom, ktorý sa nezhoduje so správnym znakom resp. slovom. Slovo je v tomto texte synonymum k výrazu token[1] s výnimkou kedy slovo môže pozostávať s dvoch tokenov, a to keď obsahuje interpunkčný znak. Posledná fáza aplikuje definované pravidlá a na ich základe pridáva definíciu chyby, typ chyby a prioritu.

Každá inštancia chyby obsahuje nasledovné atribúty:

- `id` - jednoznačný identifikátor objektu
- `mistakeCharPosInWord` - pozícia chybného znaku, pri viacerých chybných znakoch označených za sebou ako jedna chyba vracia pozíciu prvého chybného znaku, v prípade nadbytočného znaku je pozícia 0, v prípade chýbajúceho je to záporná hodnota pozície.

1. Token je kategorizovaný blok textu, obyčajne pozostáva z nedeliteľných znakov

- `correctChars` - správny znak resp. podreťazec. Ak sa jedná o nadbytočný znak, môže byť prázdny
- `writtenChars` - chybný znak resp. podreťazec označený ako jedna chyba. Ak sa jedná o chýbajúci znak, môže byť prázdny
- `correctWord` - správne slovo
- `writtenWord` - chybné slovo bez značiek označujúcich chybu
- `previousWord` - slovo bezprostredne predchádzajúce chybnému slovu
- `nextWord` - slovo bezprostredne nasledujúce po chybnom slove
- `wordPosition` - pozícia chyby v diktáte, použiteľná napr. pri výpise chýb konkrétneho slova. Pozícia je definovaná ako číslo tokenu v rámci diktátu, číslovaná od 1, 0 znamená nadbytočné slovo, záporná pozícia chýbajúce slovo na pozícii `wordposition` v absolútnej hodnote
- `lemma` - základný tvar slova získaný z výstupu morfosyntaktického analyzátoru Majka[35], ak je vo výstupe viac ako jeden tvar, berie sa vždy prvý v poradí
- `posTag` - značka popisujúca slovné druhy a iné morfosyntaktické kategórie, rovnako získané z výstupu analyzátoru Majka. Popis značiek je dostupný na webovej stránke NLP FI[36], pričom znova platí, že ak je z výstupu dostupných viacero možností, je braná vždy prvá v poradí
- `sentence` - veta v ktorej sa vyskytla chyba. Ukladá sa veta z výstupu značkovača, teda s označenými chybami. Tento atribút je určený pre budúce využitie v prípade popisu kontextových chýb, ktoré sa v tejto práci neriešia
- `priority` - prioritu slova. Priorita je číslo od 1 do 10, pričom 10 je najväčšia priorita a získava sa spolu s typom chyby a jej popisom ako výstup definície pravidiel

- `mistakeType` - typ chyby, tzn. zaradenie chyby do určitej kategórie. Úplný zoznam možných typov chýb je dostupný v prílohe C.
- `mistakeDescription` - popis chyby na základe daných pravidiel.

Celý modul je navrhnutý s dôrazom na modifikovateľnosť a nahraditeľnosť jednotlivých častí. Stačí vymeniť implementácie daných Java rozhraní. Modul je používaný aj samotnou navrhovanou aplikáciou, kde využíva objekty typu `Mistake` z popisovaného modulu a ukladá ich do databázy spolu s informáciami o používateľovi, diktáte a prístupe k diktátu ako objekt typu `Error`. Návrh bol zvolený z toho dôvodu, aby bolo rozhranie opravy diktátu čo najjednoduchšie použiteľné aplikáciami tretích strán (preto má objekt typu `Mistake` iba atribúty, ktoré je možné vyčítať z daných vstupných dát). Zamýšľané použitie objektu typu `Error` je na generovanie štatistických dát o jednotlivých používateľoch a diktátoch.

4.2 Značkovanie chýb

Formát značkovania vychádza z návrhu prezentovanom v mojej bakalárskej práci [24]. Je definovaný nasledujúcim spôsobom:

- Ak sa v slove vyskytuje chyba, prípadne je chybné celé slovo či slová, označí sa znak, reťazec, slovo alebo slová špeciálnymi značkami definovanými nižšie
- Správny znak resp. reťazec je uzavretý do párových zátvoriek *(* a *)*. Chybný znak alebo reťazec je označený zátvorkami < a > nasledujúcimi bezprostredne po správnej variante (napr. *r(y)<i>ba*)
- Chýbajúci znak alebo reťazec znakov je označený zátvorkami *(* a *)* a bezprostredne za nimi pokračuje dané slovo (napr. *strun(n)ý*) resp. sa môže nachádzať koniec slova (napr. *košil(e)*), čo sa väčšinou vyskytuje pri preklepoch.
- Nadbytočný znak v slove je označený zátvorkami < a > a bezprostredne za nimi slovo pokračuje (napr. *ran<n>ý*) resp. sa

môže nachádzať koniec slova (napr. *košile<e>*), čo sa väčšinou vyskytuje pri preklepoch.

- Chýbajúce slovo alebo slová sú celé uzavreté do zátvoriek a za ním nenasledujú párové zátvorky < a >, ale koniec slova, prípadne interpunkčný znak[2] (napr. *Stěhuje se (z domu).*)
- Nadbytočné slovo resp. slová sú uzavreté do zátvoriek < a > a za uzavieracou zátvorkou nasleduje koniec slova, prípadne interpunkčný znak (napr. *Mává <s> kapesníkem.*)

Slovo je v mojej práci definované ako reťazec znakov v texte oddelený medzerami. Slovo vo veľkej väčšine obsahuje jeden token. Existujú dve výnimky. Ak je slovo na konci vety, (Veta je definovaná ako reťazec slov ukončených interpunkčným znakom) obsahuje aj interpunkčný znak, ktorý je súčasťou slova. Ak sa v užívateľskom vstupe objaví chýbajúca alebo nadbytočná medzera, sú tokeny naľavo a napravo od chyby kvôli jednoduchšiemu spracovaniu brané ako jedno slovo. Chybný znak, reťazec znakov, chýbajúce a nadbytočné slová sú získané zo vstupu zadaného užívateľom. Správny znak alebo reťazec znakov je daný podľa prepisu diktátu uloženého v databáze aplikácie. Z označkovaného textu je pri každom výskyte chyby vytvorený nový objekt typu `Mistake`. Jeho štruktúra je popísaná vyššie.

4.3 Spracovanie chýb podľa definovaných pravidiel

Po spracovaní textu a vytvorení `Mistake` objektu pre každú chybu spracovanie prechádza do ďalšej fázy. Prostredníctvom dát uložených v objekte a popisujúcich chybu sa ju systém pokúsi čo najpresnejšie zaradiť do jednej z kategórií (popísaných v prílohe C) a priradiť mu odpovedajúce vysvetlenie a prioritu. Priorita je váha, ktorú majú jednotlivé typy chýb a je možné ju napr. použiť pri výpočte výslednej známky.

Implementácia pravidiel, ktoré dokážu chybu správne zaradiť, bola jednou z požiadaviek na výslednú aplikáciu. Zoznamy vysky-

2. Interpunkčným znakom rozumieme znak označujúci koniec vety (*. ! ?*) a špeciálne znaky (*", ; :*)

tujúce sa v rámci definície pravidiel sú umiestnené v samostatnej statickej triede. Statická trieda bola zvolená z toho dôvodu, aby sa pri každom prístupe nevytvárala nová inštancia triedy obsahujúcej zoznamy. Trieda s pravidlami k nim potom pristupuje priamo pomocou názvu triedy a názvu zoznamu.

Pravidlá sú prepísané z pseudokódu, ktorý bol súčasťou bakalárskej práce Vojtěcha Škvařila[34]. Bolo v nich však vykonaných zopár zmien kvôli zlepšeniu fungovania pravidiel a ich presnosti. Ak sa napr. vysvetlenie chyby skladá z viacerých častí, bolo najviac špecifické vysvetlenie presunuté na začiatok a ďalej boli zobrazené všeobecnejšie definície. V pseudokóde to nebolo vždy pravidlom. Bolo tiež zabezpečené, aby všetky dáta typu `String` popisujúce chybu začínali malým písmenom, s jedinou výnimkou, a to ak by šlo o chybu vo veľkosti písmen.

5 Testovanie systému

V aplikácii je každá vrstva systému testovaná zvlášť jednotkovými testami, ktoré majú overiť správne fungovanie jednotlivých funkcii. V zvýšenej miere sa pritom dbá na to, aby sa jednotlivé vrstvy testov navzájom neovplyvňovali a nespôsobovali tak propagáciu prípadných chýb do ďalších vrstiev. Tento cieľ je dosiahnutý dôsledným tzv. *napodobovaním* (anglicky mocking) funkcionality nižších vrstiev. *Napodobňovanie* funguje tak, že sa, pomocou špeciálnej knižnice, priradí metóde z nižšej vrstvy aplikácie očakávaná hodnota, s ktorou potom vyššia vrstva pracuje tak, ako by ju obdržala pri skutočnom behu aplikácie.

Dohromady je k dispozícii viac ako 200 testov, čo umožňuje jednoducho overiť zachovanie funkčnosti jednotlivých častí systému po pridaní resp. upravení niektorých jeho funkcií.

6 Nasadenie systému

6.1 Databázový systém

Ako databázový systém bol použitý PostgreSQL vo verzii 9.4. Je to voľne šíriteľný objektovo-relačný databázový systém[37]. Jedným z hlavných dôvodov použitia bola aj jednoduchá integrácia s aplikačným serverom v rámci web-hostingovej platformy Openshift. Jediný problém v chovaní databázového systému vznikal pri použití dátového typu LOB.

6.2 Aplikačný server

Ako aplikačný server bol zvolený Wildfly vo verzii 9, hlavne kvôli používateľskej prívetivosti rozhrania a jednoduchému použitiu, ale aj z dôvodu toho, že projekt je voľne šíriteľný.

6.3 Nasadenie a web-hosting

Za hosťovaciu platformu bol zvolený projekt OpenShift, hlavne kvôli jednoduchej správe a množstvu dostupných prídavných modulov, ktoré spĺňali požiadavky uvedené v predošlých sekciách. Vytvorenie novej aplikácie, priradenie modulov a nasadenie reálneho systému bolo otázkou rádovo niekoľkých minút. Systém je dostupný na adrese `http://dictatetrainerweb-jrumanov.rhcloud.com/`.

V neplatenom režime sa nasadená aplikácia po 48 hodinách bez návštevníka prepne do režimu nečinnosti a stane sa nedostupnou, potom je nutné aplikáciu znova nasadiť. Z tohto dôvodu bolo nutné prejsť na spoplatnený balík[1].

1. Viac na `https://www.openshift.com/pricing/index.html`

7 Záver

Cieľom diplomovej práce bolo vytvoriť portál pre písanie a vyhodnocovanie diktátov. Mal sa skladať s administračnej časti pre učiteľov a používateľskej časti pre žiakov. Administračná časť mala obsahovať správu skupín používateľov a možnosť nahrávať vlastné diktáty. Mala tiež obsahovať základné štatistiky o využívaní diktátov. Žiak po prihlásení mal mať možnosť vybrať si diktát na trénovanie, nechať si opraviť chyby a zobraziť svoje štatistiky. Používatelia by mali mať možnosť sa prihlásiť pomocou mena a hesla ako aj cez sociálne siete. Druhou časťou mal byť samostatný modul integrovaný s webovou aplikáciou, ktorý mal byť dostupný cez definované rozhranie a bude implementovať pravidlá nájdené počítačovými lingvistami.

Návrh aplikácie vychádzal z analýzy súčasného stavu a snažil sa dbať na budúcu rozšíriteľnosť a použitie voľne dostupných a otvorených technológií. Samostatný modul na opravu diktátov je vďaka použitiu vhodných nástrojov dostupný aplikáciám tretích strán.

Zadanie bolo až na pár drobností naplnené a v niektorých prípadoch výrazne prekročené.

Možností rozšírenia a plánov do budúcnosti je množstvo, niektoré sú spomenuté v kapitole 3.6. Hlavne sa bude jednať o vylepšenie resp. prepracovanie prihlasovania, pridania štatistík na základe agregovania väčšieho množstva rôznych dát vyprodukovaných portálom a vylepšenie prehrávania diktátov a rozšírenie funkcií prehrávača zvuku. Vďaka kvalitnému návrhu vrstvy správy dát (*back-end*) má aplikácia veľký potenciál rozšíriteľnosti do budúcnosti.

Literatúra

[1] Cross, Jay. *An informal history of eLearning* On the Horizon, vol 12, no 3, 2004, p. 103-110.

[2] Fahs, C. Ramsey. *EdX Overtakes Coursera in Number of Ivy League Partners* [online]. 2015 [cit. 2015-10-27]. Dostupné z: <http://www.thecrimson.com/article/2015/10/2/edx-ivy-league-coursera/>.

[3] Coursera Inc. *Course Certificate* [online]. 2015 [cit. 2016-01-09]. Dostupné z: <https://www.coursera.org/signature/>.

[4] Prášilová, Romana. *Diktáty v didaktice českého jazyka na 2. stupni* České Budějovice, 2014 Dostupné z: <http://theses.cz/id/29k3zb/DP-Prilov.pdf>. Diplomová práca. Jihočeská univerzita v Českých Budějovicích.

[5] ARLOW, J., NEUSTADT *UML 2 a unifikovaný proces vývoje aplikací.* [online]. I. 2011. 28 s. Computer Press. ISBN 978-80-251-1503-9.

[6] Wasson, Mike. *ASP.NET - Single-Page Applications: Build Modern, Responsive Web Apps with ASP.NET* [online]. 2013 [cit. 2015-12-15]. Dostupné z: <https://msdn.microsoft.com/en-us/magazine/dn463786.aspx>.

[7] Oracle. *The JAX-RS client API* [online]. 2015 [cit. 2016-01-10]. Dostupné z: <https://docs.oracle.com/javaee/7/api/javax/ws/rs/client/package-summary.html>.

[8] Alicea, Anthony. *Learn and Understand AngularJS* [online]. 2015 [cit. 2016-01-01]. Dostupné z: <https://www.udemy.com/learn-angularjs/>.

[9] Cavalheiro, Vasco. *The Java Origins of Angular JS: Angular vs JSF vs GWT* [online]. 2014 [cit. 2015-12-29]. Dostupné z: <http://blog.jhades.org/the-java-origins-of-angular-js-angular-vs-jsf-vs-gwt/>.

[10] W3C School. *Bootstrap 3 Tutorial* [online]. 2015 [cit. 2016-01-01]. Dostupné z: <http://www.w3schools.com/bootstrap/>.

[11] Hauser *Doplnit* 1. vyd. Berkely, California, USA: Apress, 2009. ISBN 1430218339.

[12] Bower. *Bower A package manager for the web* [online]. 2015 [cit. 2016-01-10]. Dostupné z: <http://bower.io/>.

[13] Carvajal, Ignacio Nieto. *Building your own REST API with OAuth 2.0* [online]. 2014 [cit. 2015-12-30]. Dostupné z: <http://digitalleaves.com/blog/2014/05/building-your-own-rest-api-with-oauth-2-0-i-the-basics/>.

[14] Yalkabov, Sahat. *Build an Instagram clone with AngularJS, Satellizer, Node.js and MongoDB* [online]. 2014 [cit. 2015-10-26]. Dostupné z: <https://hackhands.com/building-instagram-clone-angularjs-satellizer-nodejs-mongodb/>.

[15] Comodo CA Limited. *What is HTTPS* [online]. 2015 [cit. 2015-10-26]. Dostupné z: <https://www.instantssl.com/ssl-certificate-products/https.html>.

[16] Red Hat Inc. *Troubleshooting FAQs - How do I redirect traffic to HTTPS?* [online]. 2015 [cit. 2015-10-26]. Dostupné z: <https://developers.openshift.com/en/troubleshooting-faq.html#_how_do_i_redirect_traffic_to_https>.

[17] Mozilla Developer Network. *HTTP access control (CORS)* [online]. 2015 [cit. 2015-11-01]. Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS>.

[18] Stack Exchange Inc. *When is it safe to enable CORS?* [online]. 2015 [cit. 2015-11-01]. Dostupné z: <http://stackoverflow.com/questions/9713644/when-is-it-safe-to-enable-cors>.

[19] W3C. *Cross-Origin Resource Sharing* [online]. 2014 [cit. 2015-11-01]. Dostupné z: <http://www.w3.org/TR/cors/>.

[20] Matei, Adrian. *How to add CORS support on the server side in Java with Jersey* [online]. 2014 [cit. 2015-11-01]. Dostupné z: <http://www.codingpedia.org/ama/how-to-add-cors-support-on-the-server-side-in-java-with-jersey/>.

[21] Stack Exchange Inc. *How to enable Cross domain requests on JAX-RS web services?* [online]. 2014 [cit. 2015-11-01]. Dostupné z: <http://stackoverflow.com/questions/23450494/how-to-enable-cross-domain-requests-on-jax-rs-web-services>.

[22] Stack Exchange Inc. *CORS angular js + restEasy on POST* [online]. 2014 [cit. 2015-11-01]. Dostupné z: <http://stackoverflow.com/questions/22849972/cors-angular-js-resteasy-on-post>.

[23] Dzhuvinov, Vladimir. *CORS Filter : Cross-Origin Resource Sharing for Your Java Web Apps* [online]. 2015 [cit. 2015-11-01]. Dostupné z: <http://software.dzhuvinov.com/cors-filter-configuration.html>.

[24] Rumanovský, Jakub. *Systém na trénovanie diktátov* Brno, 2012 Dostupné z: <http://is.muni.cz/th/359581/fi_b/Bakalarka_FI_nal.pdf>. Bakalárska práca. Masarykova Univerzita.

[25] Stack Exchange Inc. *What is best way to store mp3 files in server ? Storing it in database (BLOB) , is right?* [online]. 2014 [cit. 2015-10-26]. Dostupné z: <http://stackoverflow.com/questions/11958465/what-is-best-way-to-store-mp3-files-in-server-storing-it-in-database-bl

[26] GitHub, Inc. *Ng-file-upload* [online]. 2015 [cit. 2016-01-03]. Dostupné z: <https://github.com/danialfarid/ng-file-upload>.

[27] Mitica, Gabrielle. *AngularJS and Basic Auth* [online]. 2014 [cit. 2015-11-01]. Dostupné z: <http://spaghetti.io/cont/article/angularjs-and-basic-auth/12/1.html>.

[28] Watmore, Jason. *AngularJS Basic HTTP Authentication Example* [online]. 2014 [cit. 2015-11-02]. Dostupné z: <http://jasonwatmore.com/post/2014/05/26/AngularJS-Basic-HTTP-Authentication-Example.aspx>.

[29] Woloski, M., Gontovnikas, M. *Make your Angular app a max security prison* [online]. 2014 [cit. 2015-12-29]. Dostupné z: <https://www.youtube.com/watch?v=lDb_GANDR8U>.

[30] Jones, M., Bradley, J. B., Sakimura, N. *JSON Web Token (JWT) RFC-7519* [online]. 2015 [cit. 2015-12-29]. Dostupné z: <http://self-issued.info/docs/draft-ietf-oauth-json-web-token.html>.

[31] GitHub, Inc. *DetectRegions - Issue #307 - wavesurfer.js* [online]. 2014 [cit. 2016-01-03]. Dostupné z: <https://github.com/katspaugh/wavesurfer.js/issues/307>.

[32] Stack Exchange Inc. *What is an integration test exactly?* [online]. 2015 [cit. 2016-01-01]. Dostupné z: <http://programmers.stackexchange.com/questions/48237/what-is-an-integration-test-exactly>.

[33] Google, Inc. *API Google-diff-match-patch* [online]. 2011 [cit. 2015-10-29]. Dostupné z: <http://code.google.com/p/google-diff-match-patch/wiki/API>.

[34] Škvařil, Vojtěch. *Návrh algoritmu pro vyhodnocení bezkontextových pravopisných chyb* Brno, 2014 Dostupné z: <http://is.muni.cz/th/399486/ff_b/BAKALARSKA_PRACE_27._6..pdf>. Bakalárska práca. Masarykova Univerzita.

[35] NLP FI MUNI. [online]. 2010 [cit. 2015-11-13]. Dostupné z: <https://nlp.fi.muni.cz/ma/majka.html>.

[36] NLP FI MUNI. *Ajka tagset* [online]. 2050 [cit. 2015-11-13]. Dostupné z: <https://nlp.fi.muni.cz/projekty/ajka/tags.pdf>.

[37] PostgreSQL. *PostgreSQL: About* [online]. 2015 [cit. 2015-12-15]. Dostupné z: <http://www.postgresql.org/about/>.

[38] Samwell, Jon. *URL Route Authorization and Security in Angular* [online]. 2014 [cit. 2015-10-26]. Dostupné z: <http://jonsamwell.com/url-route-authorization-and-security-in-angular/>.

[39] Mikołajczyk, Michal. *Top 18 Most Common AngularJS Developer Mistakes* [online]. 2015 [cit. 2015-10-26]. Dostupné z: <http://www.toptal.com/angular-js/top-18-most-common-angularjs-developer-mistakes>.

A Používateľský manuál

A.1 Rozhranie slúžiace na opravu diktátov

V nasledujúcej podkapitole bude popísané REST rozhranie spolu s korektným telom požiadavky a telom odpovede. V tabuľke nižšie sú uvedené dve verejné rozhrania, jedno ako odpoveď vracia všetky dostupné informácie o chybách, zatiaľ čo druhé je minimalistické a vracia iba id, popis chyby, typ chyby a prioritu. Je preto iba na používateľovi verejného rozhrania, ktoré lepšie spĺňa jeho predstavy a požiadavky.

HTTP metóda	Cesta	Povolené role	Návratový kód
POST	/api/correctDictate	neautorizovaný	201
POST	/api/correctDictate/nometa	neautorizovaný	201

Tabuľka A.1: Popis rozhrania na opravu diktátu

```
{
        transcriptText: String,
        userText: String
}
```

Listing A.1: Telo požiadavky

```
{
        totalMistakes: Integer,
        mistakes:
                [{
                  id: Long,
                  mistakeCharPosInWord: Integer,
                  correctChars: String,
                  writtenChars: String,
                  correctWord: String,
                  writtenWord: String,
                  wordPosition: Integer,
                  lemma: String,
```

```
                    posTag: String,
                    sentence: String,
                    priority: Long,
                    mistakeType: String,
                    mistakeDescription: String
                }
                {
                ...
                }]
}
```

Listing A.2: Telo odpovede `/api/correctDictate`

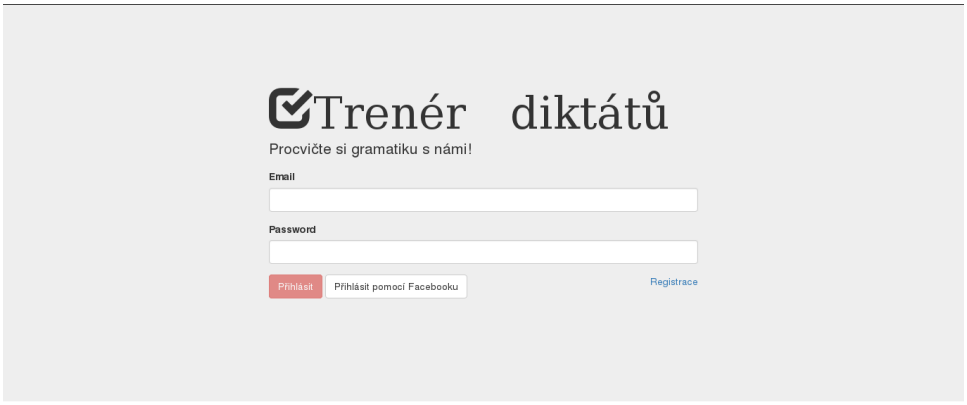
```
{
        totalMistakes: Integer,
        mistakes:
                [{
                    id: Long,
                    priority: Long,
                    mistakeType: String,
                    mistakeDescription: String
                }
                {
                ...
                }]
}
```

Listing A.3: Telo odpovede `/api/correctDictate/nometa`

Viac o tom, čo jednotlivé atribúty znamenajú je uvedené v kapitole 4.

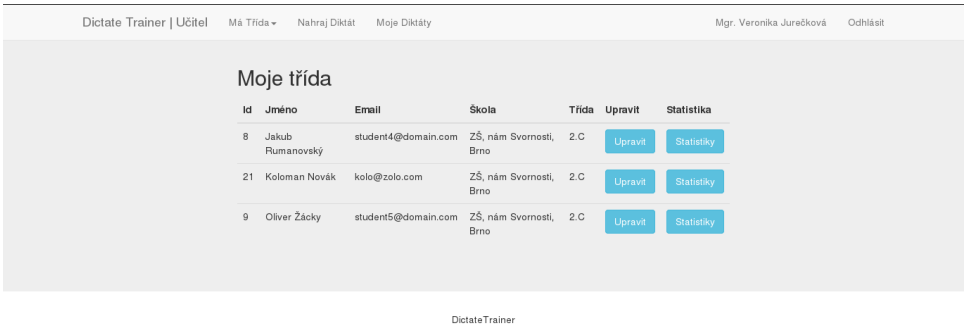
B Snímky obrazoviek

V tejto prílohe bude ukázaných niekoľko dôležitých obrazoviek všetkých troch používateľských rolí s krátkym popisom.

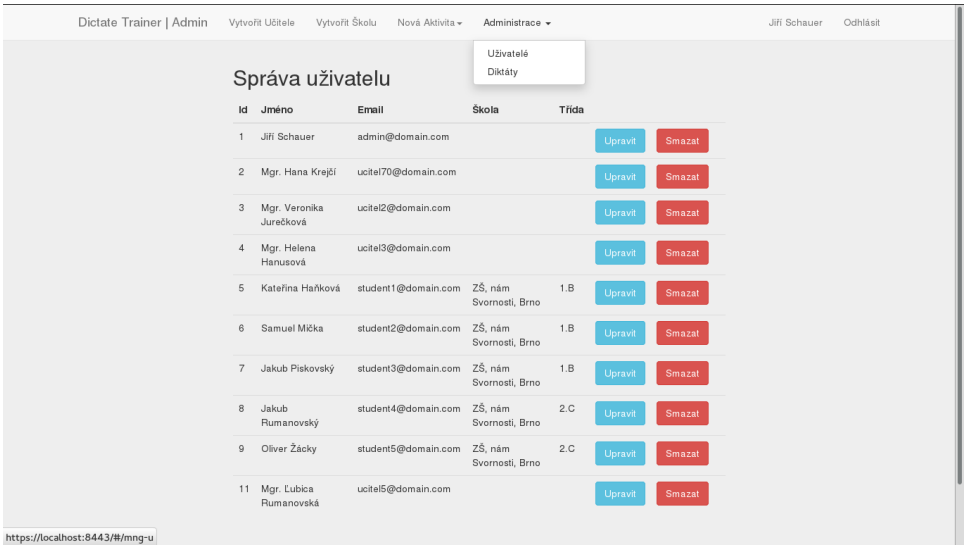


Obr. B.1: Prihlasovacia obrazovka, vľavo sú tlačidlá na prihlásenie, vpravo link na registráciu nového žiaka

Obr. B.2: Stránka trénovania diktátu, tlačidlom trénuj sa spúšťa diktát, tlačidlom oprav je spustená oprava a žiak je presmerovaný na výsledky

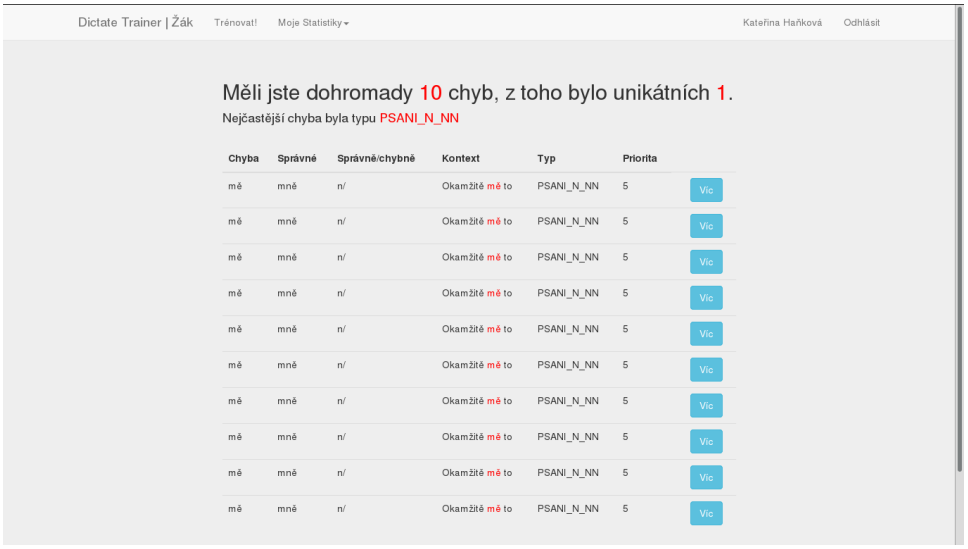


Obr. B.3: Správa skupiny žiakov učiteľom, učiteľ môže upavovať pofil žiaka, prípadne vidieť jeho štatistiky

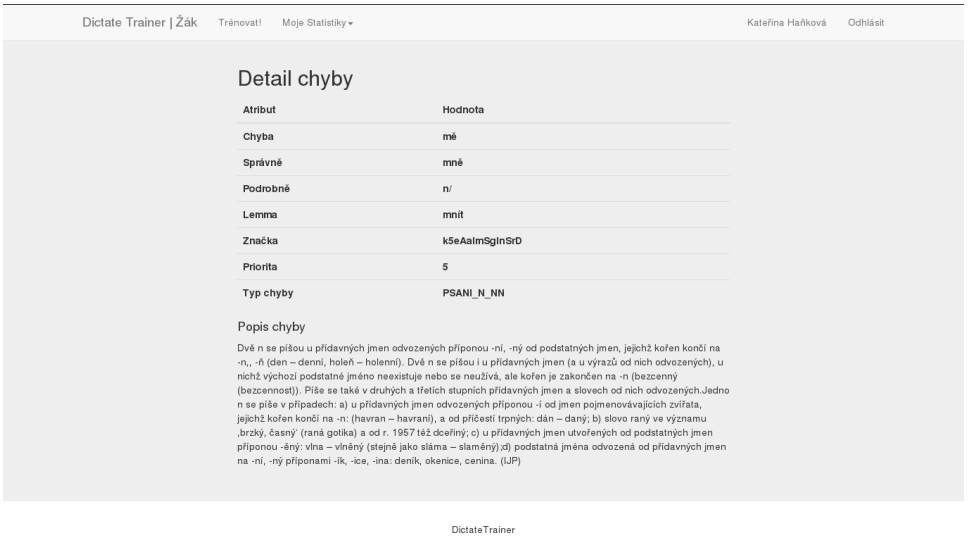


Obr. B.4: Správa všetkých používateľov prostredníctvom administrátora

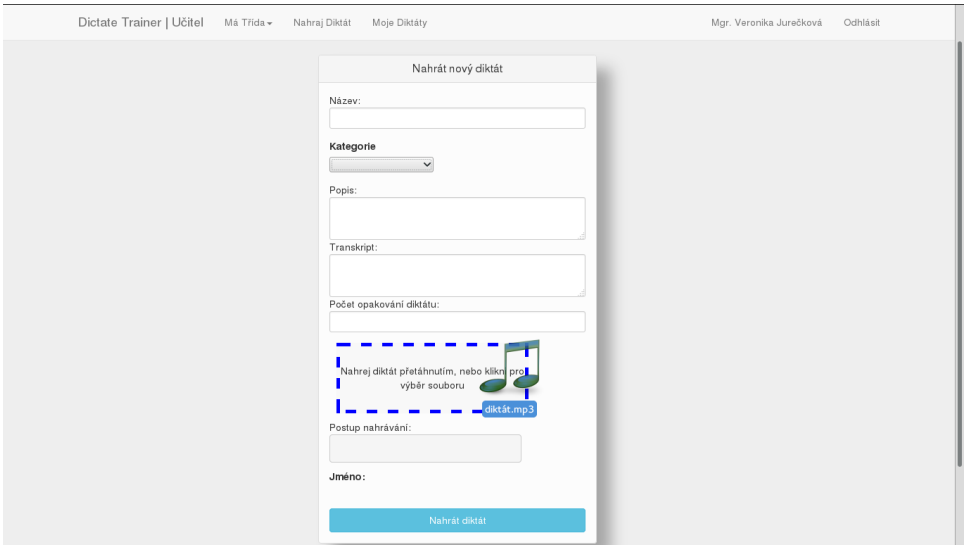
Obr. B.5: Štatistiky v praxi – zobrazia sa iba diktáty, ktoré žiak už trénoval, po kliknutí sa prejde na štatistiku



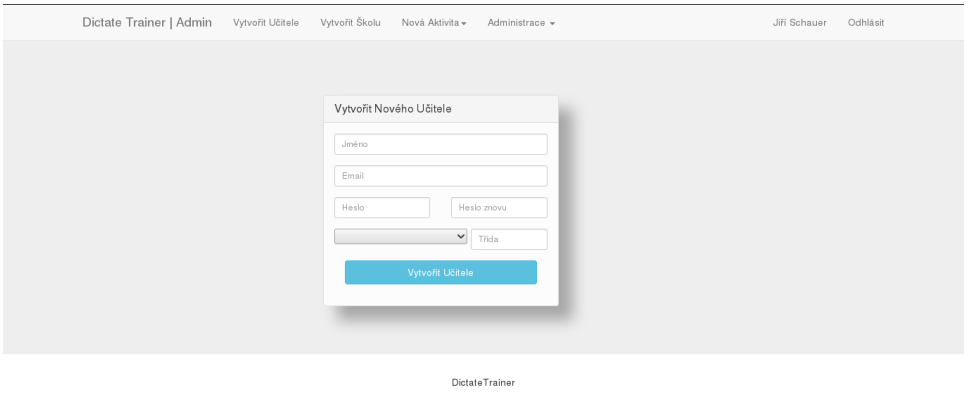
Obr. B.6: Štatistika vybudovaná na základe diktátu. Vidieť všetky chyby ktoré žiak v diktáte kedy urobil a tiež koľko unikátnych – teda rôznych – chýb z je z celkového počtu



Obr. B.7: Detailné zobrazenie konkrétnej chyby s popisom



Obr. B.8: Nahrávanie diktátu do systému učiteľom



Obr. B.9: Vytváranie nového učiteľa spolu s jeho triedou prostredníctvom administrátora

C Zoznam jednotlivých typov chýb

Nižšie je uvedený zoznam možných typov chýb, ktorý bol jedným z výstupov bakalárskej práce Vojtěcha Škvařila[34] v poradí, v akom sa vyskytol v priloženom pseudokóde popisujúcom jednotlivé pravidlá.

Pravopis:

- Vyjmenovaná slova
- Psaní i/y po písmenu c
- Psaní předpon s-, z-
- Psaní předložek s, z (z postele, s knihou, s sebou)
- Pravopis a výslovnost přejatých slov se s – z
- Psaní dis-, dys-
- Psaní slov se zakončením -manie
- Psaní n – nn
- Psaní spřežek a spřahování
- Složená přídavná jména - První část přídavného jména je zakončena na -sko, -cko, -ně nebo –ově
- Velká písmena

Slovotvorba:

- Přídavná jména zakončená na -icí – -ící
- Přídavná jména zakončená na -ní – -ný
- Typ ačkoli – ačkoliv, kdokoli – kdokoliv
- Vokalizace předložek

Jevy, které nejsou v Akademické příručce českého jazyka přímo uvedeny:

- Zájmena typu vaši/vaší, ji/jí, ni/ní
- Psaní bě/bje, vě/vje, pě
- Souhlásky párové
- Diakritika
- i/í po měkkých a obojetných souhláskách
- Zájmena mně x mě a slova obsahující mě a mně

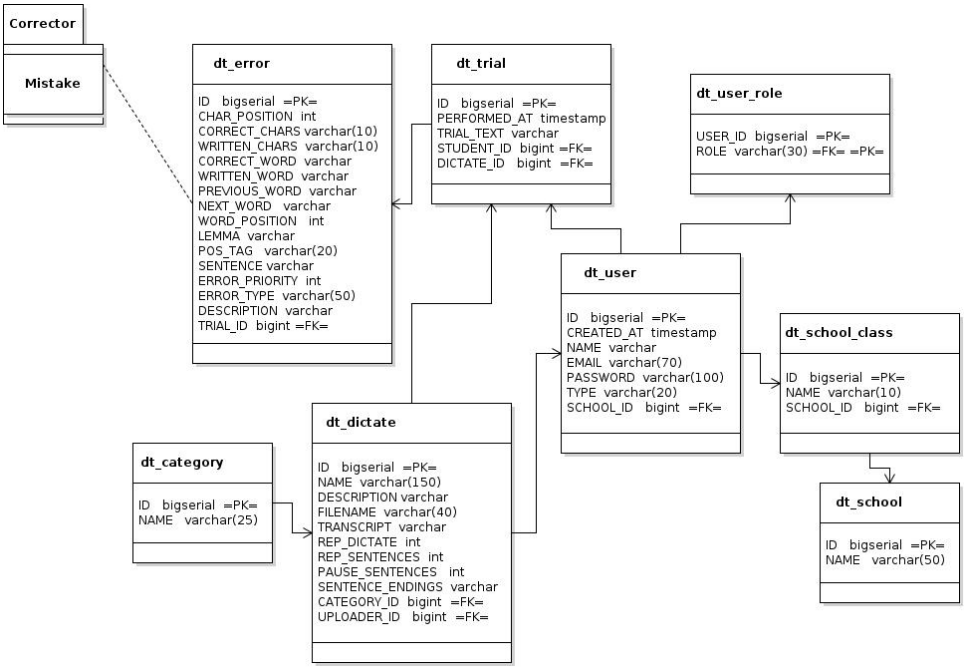
K týmto typom je naviac pridaný typ nadbytočného resp. chýbajúceho slova.

Pridané typy ktoré nie sú v bakalárskej práci uvedené:

- Chybějící slovo
- Nadbytečné slovo
- Ostatní

D Diagramy

D.1 Dátový model



Obr. D.1: Dátový diagram

`Mistake` nie je entita a preto nie je súčasťou dátového diagramu. Je v ňom však uvedená pre ilustráciu celkovej štruktúry systému.

E Obsah CD

Súčasťou diplomovej práce je priložené CD, ktoré obsahuje:

- text diplomovej práce (dictatetrainer_rumanovsky.pdf)
- zdrojový kód portálu pre písanie a vyhodnocovanie diktátov (archív dictatetrainer.zip)
- skompilovaná aplikácia pripravená na nasadenie podľa návodu (achív dictatetrainer-ear-0.0.1-SNAPSHOT.ear)
- pokyny k inštalácii portálu pre písanie a vyhodnocovanie diktátov (pokyny_instalacia.txt)
- súbor s SQL príkazmi na vytvorenie databázy (db.sql)
- upravený konfiguračný súbor aplikačného servera `Wildfly` (standalone-dt.xml)