Masarykova univerzita
Fakulta informatiky

Portál pre písanie a vyhodnocovanie diktátov

Diplomová práca

Bc. Jakub Rumanovský

Brno, Jeseň 2016

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. Žiakom bude poskytovať možnosť trénovať diktáty a okamžite vidieť výsledky a štatistiky. Učiteľom umožní spravovať pridelené skupiny žiakov rozdelených do skupín - školských tried a nahrávať diktáty. Portál bude tiež poskytovať verejné rozhranie slúžiace na opravu diktátov, ktoré bude zároveň použiteľné ako služba.

Abstract

The aim of this Master thesis is to create a dictate web system, which can be used mainly by pupils of elementary school and high school and their teachers. For pupils, it will offer the possibility of training dictates and they will see the results and statistics immediately. For teachers, it will provide the grouping of pupils to school classes that will be administrated by teachers and possibility for upload of dictate. The web portal will also offer public REST API for correcting the dictates, which will be also usable as a service.

Kľúčové slová

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

Obsah

1 **Úvod** 1
1.1 *Ciele práce* 2
1.2 *Štruktúra práce* 3
2 **Analýza systému** 4
2.1 *Existujúce riešenia* 4
2.1.1 Štruktúra existujúcich riešení 4
2.1.2 Problémy v existujúcich riešeniach 5
2.1.3 Výhody a nevýhody existujúcich riešení 5
2.2 *Navrhovaný systém* 5
2.2.1 Špecifikácia požiadaviek 5
2.2.2 Výhody a nevýhody navrhovaného systému . . 7
3 **Návrh a implementácia systému** 8
3.1 *Problémy a slepé vetvy pri návrhu systému* 9
3.1.1 Java Server Faces 9
3.1.2 JSON Web Token (JWT) 9
3.1.3 Atribúty popis chyby a typ chyby v samostatnej tabuľke 10
3.1.4 Integračné testy 10
3.2 *Diagram prípadov použitia* 11
3.1.5 Návrhové chyby zistené pri implementácii . . . 11
3.3 *Dátový model* 12
3.3.1 Entity - charakteristika 12
3.4 *Rozdelenie aplikácie na moduly* 14
3.5 *Štruktúra aplikácie* 15
3.5.1 Vrstva prístupu k databáze 15
Prístup k entitám 15
3.5.2 Bussiness vrstva aplikácie 15
3.5.3 Vrstva rozhraní - REST 16
Nahrávanie diktátov 16
3.5.4 Prezentačná vrstva 16
Angular JS JavaScript framework 16
CSS a Bootstrap 17
HTML5 17
3.6 *Ukladanie dát popisujúcich diktát* 18
3.7 *Prehrávanie diktátov* 19

3.7.1 Diktát 19
3.8 *Nahrávanie diktátu na strane používateľského rozhrania* 19
3.9 *Detekcia hraníc jednotlivých viet vrámci diktátu* 20
3.10 *Zabezpečenie aplikácie* 20
3.10.1 Zabezpečená komunikácia pomocou HTTPS . . 20
3.10.2 Basic autentifikácia a AngularJS 22
3.10.3 Zabezpečenie ciest na úrovni AngularJS frameworku 22
3.10.4 Autentifikácia pomocou sociálnych sietí 23
Autorizačný protokol OAuth 23
Možnosti integrovania sociálnej autentifikácie . 23
Zvolený prístup pre Diktátový systém 25
Implementácia sociálnej autentifikácie 25
3.10.5 Cross Domain Resource Sharing (CORS) 26
3.11 *Správa používateľov systému* 28
3.12 *Návrh používateľského rozhrania* 28
4 **Modul na opravu diktátov** 29
4.1 *Analýza* 29
4.2 *Návrh a implementácia* 29
4.3 *Značkovanie chýb* 31
4.4 *Spracovanie chýb podľa definovaných pravidiel* 32
5 **Testovanie systému** 34
6 **Nasadenie systému** 35
6.1 *Databázový systém* 35
6.2 *Aplikačný server* 35
6.3 *Nasadenie a web-hosting* 35
7 **Záver** 36
A **Používateľský manuál** 42
A.1 *Rozhranie slúžiace na opravu diktátov* 42
A.2 *Manuál ku grafickému užívateľskému rozhraniu* 43
B **Snímky obrazoviek** 44
C **Zoznam jednotlivých typov chýb** 45
D **Obsah CD** 47

1 Úvod

Informačné technológie sa v súčasnosti dostávajú do stále väčšieho množstva odvetví ľudskej činnosti. Sú medzi nimi aj také oblasti, v akých by sme ich ešte pred pár rokmi nevedeli predstaviť. V mnohých z nich majú nepopierateľne obrovský potenciál rozvoja. Jedným z príkladov takéhoto odvetvia je e-learning, alebo slovensky vzdelávanie prostredníctvom informačných technológií. V posledných rokoch došlo k rapídnemu rozvoju tejto oblasti. Prispeli k tomu nemalou mierou aj niektoré renomované univerzity [1], ktoré poskytujú svoje kurzy zdarma alebo za menší poplatok. Táto evolúcia vzdelávania poskytuje nespočetné možnosti 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. Z horeuvedených príkladov je vidno, že e-learning v akejkoľvek forme má zmysel aj 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. Jeden z jeho najpodstatnejších aspektov je správna gramatika a pravopis. Tie sa žiaci učia na školách od najútlejšieho veku a osvojovanie gramatiky prebieha už niekoľko desaťročí rovnakým spôsobom. Ku klasickým nástrojom na trénovanie správneho pravopisu patrí písanie diktátov. Do tejto oblasti moderné technológie ešte celkom nestihli preniknúť. Diktát sa píše na papier a následne je opravovaný a hodnotený učiteľom.

Pretože v súčasnej dobe neexistuje aplikácia, ktorá by používateľovi 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 v písaní. Aplikácia si kladie za cieľ rozšíriť písanie a korekciu diktátov, ktorá sa momentálne píše žiakmi a opravuje učiteľmi, o výhody, ktoré ponúkajú informačné technológie. Konkrétne 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 niekoľko záujemcov, ktorí sa budú podieľať na funkčnom testovaní. Jednak sú to vybrané základné školy a jednak sa bude systém 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, ktorá bude dostupná cez definované rozhranie a bude implementovať pravidlá nájdené počítačovými lingvistami. Tieto pravidlá detekujú 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. Plynule prechádza k popisu navrhovaného systému, popisuje jeho výhody oproti súčasnému stavu a špecifikuje požiadavky na aplikáciu.
Kapitola 3 - Návrh a implementácia systému obsahuje popis návrhu diktátového systému, poprípade popisuje niektoré implementačné detaily, pokiaľ nie sú jasné. Na začiatku sa kapitola venuje problémom a slepým vetvám vývoja systému s odôvodnením, prečo konkrétny popisovaný jav alebo požiadavka bola nakoniec riešená iným spôsobom. Kapitola tiež zahŕňa popis štruktúry aplikácie a postupne sa venuje zvlášť každej vrstve.
Kapitola 4 - Modul na opravu diktátov obsahuje popis samostatného modulu na opravu diktátov, ktorý je integrovaný spoločne s aplikáciou. Rozhodol som sa ho, kvôli väčšej prehľadnosti, presunúť do samostatnej kapitoly. Tá zahŕňa analýzu a návrh modulu ako aj popis fungovania a v prípade potreby aj implementačné detaily.
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

2.1.1 Štruktúra existujúcich riešení

Ako bolo zmienené vyššie, pri výuke gramatiky hrá ešte stále významnú úlohu zbierky diktátov, z ktorých je potom diktát diktovaný učiteľom v triede počas vyučovania. Zbierka diktátov slúži ako doplnok k učebniciam českého jazyka, pretože učebnice (kvôli obmedzenému rozsahu), nemôžu prinášať dostatočnú zásobu učiva k precvičovaniu pravopisu.

Taktiež existuje veľká skupina cvičení zadarmo dostupných na internete. Prášilová[22] ich však vo svojej diplomovej práci nazýva skôr ako diktátmi doplňovacími cvičeniami. Výhodou takejto formy je podľa nej okamžité vyhodnotenie, ktoré však nemusí byť príliš dobrý motivačný prvok. V svojej diplomovej práci tiež uvádza, že by doplňovacie cvičenia mali slúžiť hlavne k precvičovaniu, nie ku klasifikácii.

Ďalšou skupinou existujúcich riešení sú audiodiktáty, ktoré sú postavené na rovnakých textoch ako zbierky diktátov. Nevýhodou týchto diktátov sú technické problémy, ku ktorým môže dochádzať počas diktovania (napr. šum, preskakovanie). Ku komplikáciám môže podľa Prášilovej tiež dochádzať vo chvíli keď žiak zle rozumel, chce sa spýtať, tým sa dostane do sklzu a stráca pozornosť. Hlavnú nevýhodu však vidí v nemožnosti prispôsobiť tempo danej skupine. Výhodu potom vidí v zvýšenej pozornosti, ktorú môže učiteľ žiakom venovať namiesto čítania textu.

V súčasnosti sa na internete objavujú aj celé portály zamerané na precvičovanie gramatických javov. Jedným z zo zástupcov 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. audiodiktáty. Tieto 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 pri skončení cvičenia. Ďalším portálom dostupným na internete je web *umimcesky.cz*. Na tomto portáli si žiak precvi-

čuje gramatické javy pomocou testových otázok (väčšinou typu vyber správnu možnosť) a po označení odpovede sa zobrazí, či bola správna a ak áno, prečo boli ostatné nesprávne.

2.1.2 Problémy v existujúcich riešeniach

2.1.3 Výhody a nevýhody existujúcich riešení

2.2 Navrhovaný systém

2.2.1 Špecifikácia požiadaviek

Požiadavka je neformálna špecifikácia niečoho, čo má byť implementované. Požiadavky sa delia na funkčné a nefunkčné. Funkčné požiadavky (FP) určujú aké chovanie bude systém bude systém ponúkať. Nefunkčné požiadavky (NFP) špecifikujú vlastnosti alebo obmedzujúce podmienky daného systému. Požiadavky sú základom všetkých systémov, sú v podstate vyjadrením toho, čo by mal systém robiť, avšak nie toho ako by to mal robiť[25].

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 dané a ich výber je špecifikovaný neskôr v texte.

NFP2 Jednoduchá rozšíriteľnosť
Webový systém na komplexnú prácu s diktátmi by mal byť ľahko rozšíriteľný o novú funkcionalitu, najmä ďalej zmienený samostatný modul určený na opravu používateľom napísaného textu.

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

- ADMINISTRATOR - používateľská rola je slúžiť na správu systému. Jeho hlavnou úlohou je pridávať používateľov role učiteľ (definovanej nižšie) a priradiť im skupiny používateľov do ich správy. Budú mať tiež možnosť spravovať portál, jeho používateľov, diktáty a podobne.

- TEACHER - rola reprezentujúca učiteľa. Má možnosť pridávať nový diktát, spravovať priradenú skupinu používateľov.
- STUDENT - má možnosť listovať dostupnými diktátmi, trénovať diktát, vidieť vlastné štatistiky a upravovať vlastný profil.

FP2 Správa skupín používateľov
Učiteľovi je pridelená skupina, alebo skupiny používateľov (triedy v rámci školy), ktorú môže spravovať a vidieť ich štatistiky chýb v diktátoch, počet pokusov a podobne.

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

FP4 Používateľské štatistiky
Je možné určeným skupinám používateľov zobraziť základné štatistiky používateľov a diktátov.

FP5 Prihlasovanie používateľov pomocou mailu alebo sociálnych sietí
Každý používateľ má možnosť sa prihlásiť klasicky pomocou mailovej adresy, alebo sociálnych sietí.

FP6 Registrácia používateľov
Používateľ má možnosť sa pri prvom vstupe na portál zaregistrovať zvoleným emailom a heslom.

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
Portál obsahuje samostatný modul slúžiaci na opravu diktátov, ktorý je prístupný cez známe rozhranie. Prístupný je komukoľvek, kto ho chce použiť. Rozhranie aj 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 detekovať a pridávať zdôvodnenia pre vybrané jazykové javy v češtine.

2.2.2 Výhody a nevýhody navrhovaného systému

3 Návrh a implementácia systému

Webový systém je rozdelený na tri časti. *Dátovú vrstvu*, označovaná anglickým slovom backend, ktorá sa stará o ukladanie dát a navonok je možné k nej pristupovať pomocou tzv. REST endpointov – známych adries, ktoré po obdržaní dát v správnom formáte v JSON vrátia požadované údaje v definovanej forme.

Druhou časťou je *prezentačná vrstva*, známa aj pod anglickým ekvivalentom frontend, ktorá bude komunikovať s vrstvou prístupu k dátam skrz REST rozhrania. Tieto sú zabezpečené proti neoprávnenému prístupu neautorizovaného používateľa. Ten sa bude musieť na získanie oprávnenia zaregistrovať. Registrovať je možné iba užívateľov role STUDENT (žiaci), učiteľ (rola TEACHER) musí požiadať o vytvorenie účtu administrátora (rola ADMINISTRATOR). Overenie prístupu prebieha zavolaním špeciálnej adresy, definovanej v jednej z následujúcich kapitol spolu s užívateľským menom a heslom a následným povolením/odmietnutím prístupu na základe týchto údajov. Prezentačná vrstva je navrhnutá ako tzv. jednostránková aplikácia (skrátene SPA – Single Page Application). SPA sú webové aplikácie, ktoré načítajú jediný HTML stránku a potom ju dynamicky obnovujú podľa toho, ako s ňou používateľ komunikuje[20].

Pri implementácii systému je kladený dôraz na použitie moderných technológií a štruktúra 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 podľa príslušnosti k vrstve systému.

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

3.1 Problémy a slepé vetvy pri návrhu systému

Pri návrhu systému sa vyskytlo niekoľko problémov, ktorým bolo možné jednoducho zamedziť dôkladnejšie vykonanou analýzou a/alebo hlbšími znalosťami technológií. V tejto podkapitole budú spomenuté tie najzávažnejšie.

3.1.1 Java Server Faces

Z hľadiska použitých technológií v prezentačnej vrstve aplikácie (viď štruktúra aplikácie - kapitola 3.5) sa najprv uvažovalo o použití Java Server Faces (skrátene JSF), resp. niektorej z jeho variant. JSF je framework pre programovací jazyk Java, ktorý zjednodušuje tvorbu webových užívateľských rozhraní za pomoci znovupoužiteľných komponent. Tie sú uložené na strane servera v pamäti a pri užívateľskej požiadavke vytvoria kompletný vzhľad stránky, ktorý je potom odoslaný klientovi. To si vyžaduje jednu vrstvu abstrakcie naviac[26]. Tento koncept bol opustený z dôvodu zbytočnej rozsiahlosti a zložitosti frameworku pre navrhovanú aplikáciu.

3.1.2 JSON Web Token (JWT)

Počas implementácie funkčnej požiadavky 5 - Prihlasovanie pomocou sociálnych sietí sa objavila možnosť implementovať prihlasovanie iným spôsobom – pomocou autentifikácie na základe tzv. tokenu. Nižšie je krátke porovnanie s klasickou autentifikáciou a sú uvedené výhody a nevýhody jednotlivých prístupov.

Klasická autentifikácia prebieha tak, že sa používateľ prihlási na server a ako odpoveď dostane tzv. buď priamo údaje o používateľovi, ktoré sa potom uložia do cookie, aby s nimi aplikácia mohla pracovať (tento postup sa však s bezpečnostného hľadiska neodporúča), alebo je odpoveďou cookie s identifikátorom sedenia (angl. session). Problém sedenia je v prvom rade jeho neuniverzálnosť, pokiaľ chceme použiť viac spôsobov prihlasovania. Riešením je tzv. autentifikácia založená na tokene[27].

JSON Web token (skrátene JWT) je autentifikačný token reprezentovaný pomocou JSON objektu. Zakódovaný token pozostáva z

troch podreťazcov zakódovaných pomocou kódovania *Base64*[1] a oddelených od seba znakom ".". Prvý podreťazec nesie zakódované 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. že nie je potrebné udržiavať sedenie pre každého používateľa na strane servera. Overenie prihlásenia je možné jednoduchou kontrolou, či existuje token v úložisku prehliadača[28]. Ďalšia nesporná výhoda spočíva v univerzálnosti riešenia pokiaľ používame viacero prihlasovacích metód (ako napr. sociálne siete).

Vzhľadom na pokročilý stav implementácie systému bol tento koncept napriek jeho nesporným výhodám opustený a výsledný systém používa Basic-Auth prihlasovanie. Z tohoto dôvodu bolo tiež nutné upraviť spôsob, akým bude fungovať prihlasovanie pomocou sociálnych sietí. Viac v sekcii 3.10.4.

3.1.3 Atribúty popis chyby a typ chyby v samostatnej tabuľke

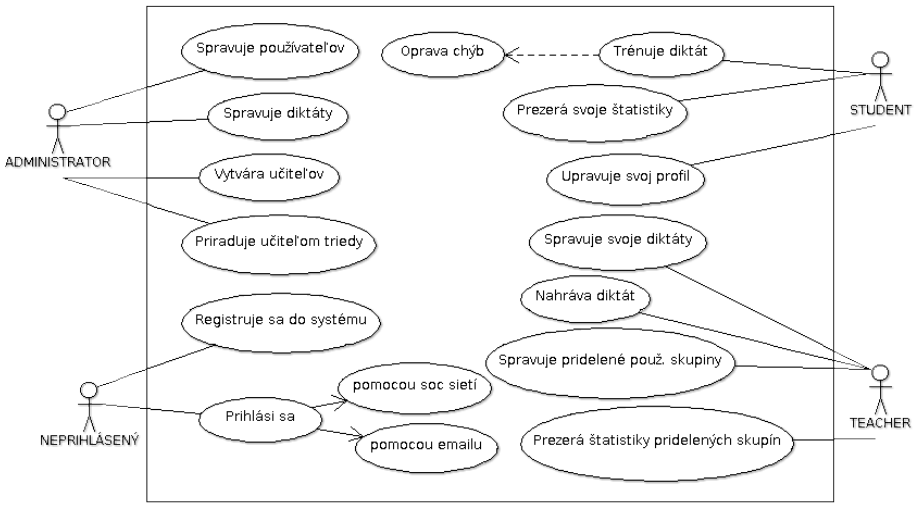
V neskoršom návrhu sa tiež počítalo s oddelením atribútu `errorType` a `errorDescription` do samostatných tabuliek. Takéto riešenie by bolo omnoho lepšie, pretože by nedochádzalo k duplikácii informácií v databáze a neplytvalo by sa diskovým priestorom. V súčasnej podobe však oba atribúty ostali vrámci tabuľky entity `Error`. Dôvodom bol nedostatok času na zmenu návrhu.

3.1.4 Integračné testy

Integračné testy, na rozdiel od jednotkových testov, ktoré testujú každú časť systému zvlášť, testujú ako dobre fungujú jednotlivé časti spolu[32]. Napriek tomu, že v aplikácii je pripravený modul v budúcnosti slúžiaci tomuto účelu, nakoniec je systém testovaný výhradne jednotkovými testami. Viac v kapitole 5.

1. Kódovanie ktoré prevádza binárne dáta (v tomto prípade oktety UTF-8) na postupnosť tlačiteľných znakov

3.2 Diagram prípadov použitia

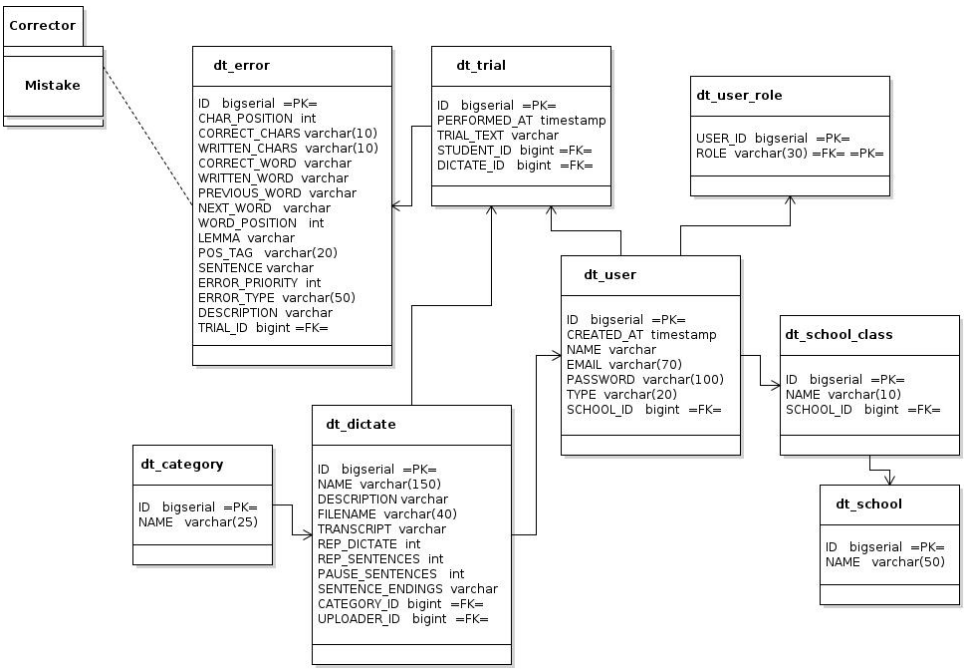


Obr. 3.1: Najvrchnejšia vrstva diagramu prípadov užitia

3.1.5 Návrhové chyby zistené pri implementácii

Počas návrhu a implementácie sa vyskytli chyby kvôli ktorým bolo nutné znova sa vrátiť k analýze a zakomponovať riešenie problému. Konkrétne sa jednalo o implementáciu funkčnej požiadavky 2 (Práca so skupinami používateľov), kde bolo po úvahe nutné pridať dve nové entity reprezentujúce školu resp. triedu. Druhý problém kvôli ktorému bolo nutné sa vrátiť k návrhu, bola funkčná požiadavka 4 (Používateľské štatistiky), kde bolo nutné pridať entitu chyba, ktorá obsahuje všetky informácie o konkrétnej chybe. Nižšie je vyobrazený dátový diagram.

3.3 Dátový model



Obr. 3.2: Dátový diagram

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

3.3.1 Entity - charakteristika

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 gramatický jav, ktorý má diktát v prvom rade precvičiť. Napr. vybrané slová, veľké a malé písmená, spodobovanie atď. Kategórie definuje správca systému.

- `Dictate` reprezentuje samotný diktát. Obsahuje základné informácie o diktáte: meno, krátky popis diktátu, názov súboru, pod ktorým je uložený na serveri a odkaz na kategóriu, do ktorej diktát spadá. Keďže jedna z požiadaviek na systém bola možnosť nahrávať diktáty, obsahuje tiež odkaz na používateľa, ktorý diktát vytvoril. Entita `Dictate` tiež ukladá dáta popisujúce samotný zvukový súbor s diktátom, menovite časové značky koncov viet, počet opakovaní diktátu a 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 (USER resp. STUDENT) a aké role môže zastávať (rola ADMINISTRATOR a TEACHER pre učiteľa resp. STUDENT pre študenta).

- Entita `Trial` obsahuje informácie o každom dokončenom diktáte. Pozostáva s neopraveného textu napísaného študentom, časovej známky vykonania pokusu, odkazu na trénovaný diktát a rovnako odkazu na trénujúceho študenta. Tieto informácie môžu byť použité predovšetkým za účelom vytvárania štatistických dát.

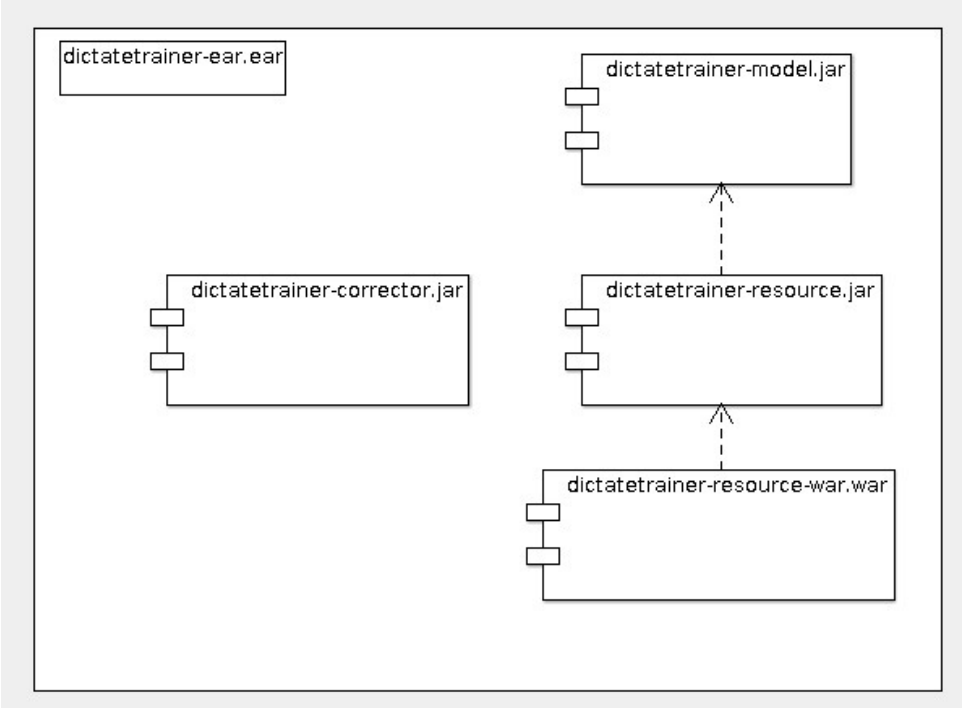
- Entita `Error` reprezentuje jednu chybu používateľa v diktáte. Chyba nesie jednak všetky popisné informácie o konkrétnej chybe (ktoré sú totožné s výstupom modulu `Corrector`, pretože ho k svojmu vytvoreniu využíva). K tomu obsahuje odkaz na entitu `Trial`, odkaz na diktát, v ktorom sa chyba vyskytla a odkaz na používateľa, ktorý ju urobil. Rovnako ako u predchádzajúcej entity, aj tieto dáta sú predovšetkým určené na účely štatistiky.

- Entita `School` združuje objekty reprezentujúce jednotlivé školy v systéme. Slúži v prvom rade na reprezentáciu skupín používateľov, prípadne pre štatistické účely.

- Entita `SchoolClass` reprezentuje konkrétnu školskú triedu v určitej škole. Spolu s entitou `School` tvoria v systéme jedinečné dvojice, do jednej z ktorých musí patriť každý registrovaný žiak.

3.4 Rozdelenie aplikácie na moduly

Systém je rozdelený na 6 modulov. Prvý s názvom `dictatetrainer-model` obsahuje triedy zabezpečujúce prístup k databáze, byznis vrstvu aplikácie a k nim prislúchajúce testy. Modul `dictatetrainer-resource` obsahuje verejne prístupné rozhrania spolu s ich jednotkovými testami, `dictatetrainer- corrector` je zvláštny modul zabezpečujúci opravu diktátu a poskytuje verejne prístupné rozhranie. Modul `dictatetrainer- resource-war` pozostáva s prezentačnej vrstvy aplikácie a nakoniec modul `dictatetrainer-ear` zabezpečuje správne zbalenie celého systému, oddeľuje závislosti do zvláštneho adresára a backend do zvláštneho archívu tak, aby bola štruktúra aplikácie čo najprehľadnejšia.



Obr. 3.3: Štruktúra modulov aplikácie

3.5 Štruktúra aplikácie

3.5.1 Vrstva prístupu k databáze

Aby bolo možné abstrahovať definíciu tabuliek a relácii medzi nimi od konkrétnej implementácie databázového systému (ako je napr. Postgres, MySQL, Oracle a pod.), je nutné použiť framework 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 vzťahy medzi entitami, musí obsahovať aj preťažené metódy `equals` a `hashCode`. Štandardný framework umožňujúci ORM je Java Persistence API (skrátene JPA). Je to vlastne sada rozhraní definujúcich ako by malo ORM fungovať. Existuje viacero implementácií JPA - ako napr. EclipseLink, OpenJPA alebo Hibernate. Aplikácia používa poslednú spomínanú knižnicu.

Prístup k entitám

Entity je nutné nielen vytvoriť, ale k nim aj vedieť pristúpiť a vykonať s nimi základné operácie. Základnými operáciami rozumieme tzv. CRUD operácie[2]. Tieto operácie sú definované v špeciálnej triede vlastnej pre každú entitu, ktorá tvorí akýsi medzičlánok spájajúci entity a databázu. K základným operáciám je pre každú entitu pridaná možnosť filtrovania výsledkov na základe určených atribútov (resp. z databázového pohľadu stĺpcov tabuľky).

3.5.2 Bussiness vrstva aplikácie

Servisná vrstva aplikácie vytvára ďalší stupeň abstrakcie aplikácie. Spolupracuje s vrstvou prístupu k dátam a poskytuje svoje rozhranie vrstve nad ňou.

2. **C**reate = vytvorenie, **R**ead = čítanie, **U**pdate = aktualizácia, **D**elete = zmazanie

3.5.3 Vrstva rozhraní - REST

Nahrávanie diktátov

Jednou zo zadaných požiadaviek na aplikáciu bolo umožniť učiteľom nahrávať diktáty pomocou užívateľského grafického rozhrania a pridať k nim pomocné dáta, ktoré budú diktát popisovať (počet opakovaní jednotlivých viet, počet opakovaní celého diktátu, dĺžka pauzy medzi vetami...). Ako je spomenuté v kapitole **??**, samotný súbor s diktátom je uložený oddelene od metadát.

Návrh tejto časti bol rozdelený na tri etapy - sprístupnenie zložky určenej pre ukladanie diktátov na aplikačnom serveri, vytvorenie REST rozhrania pristupujúcemu k tejto zložke a navrhnutie nahrávania diktátu v prezentačnej vrstve na strane klienta.

Zložka sprístupnená učiteľom pre nahrávanie diktátov musí byť umiestnená mimo samotný kontext aplikácie. Hlavným dôvodom je nutnosť zachovania nahratých súborov v prípade opätovného nasadenia aplikácie kvôli vylepšeniam resp. oprave chýb.

3.5.4 Prezentačná vrstva

Angular JS JavaScript framework

AngularJS je framework založený na jazyku JavaScript a implementujúci návrhový vzor *MV**. Skratka *MV** označuje začiatočné písmená anglických výrazov *Model* (reprezentácia dát - pomocou premenných jazyka JavaScript), *View* (vrstva používateľského rozhrania - deklarovaná pomocou HTML a špeciálnych atribútov pridaných AngularJS frameworkom) a hviezdičkou sa rozumie mechanizmus, ktorý umožňuje komunikáciu medzi týmito dvoma vrstvami. Tento mechanizmus sa nazýva *Digest Loop* a pridáva sledovanie zmien atribútov deklarovaných v modeli v jazyku JavaScript, na strane používateľského rozhrania[30].

Hlavnou výhodou frameworku AngularJS oproti napr. JSF je, že sa logika prezentačnej vrstvy presúva na stranu klienta. [26] 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

CSS a Bootstrap

Vzhľad používateľského rozhrania je implementovaný pomocou kaskádových štýlov a *Bootstrap* frameworku. Bootstrap je kolekcia nástrojov pre vytváranie webových aplikácií [31]. Pomocou HTML atribútov (predovšetkým atribútov `class` a `id`[4]) je možné jednoduchým spôsobom definovať vzhľad jednotlivých komponent používateľského rozhrania.

HTML5

3.6 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 bakalárskej práci bol systém navrhnutý tak, že boli tieto informácie uložené v rámci mp3 súboru s diktátom [8]. Tento prístup má však niekoľko nevýhod. Jednak je použitie zvukových formátov obmedzený na mp3, pretože sa informácie ukladajú do ID3v2 tagu[3]. Druhá nevýhoda 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 zvlášť a získať z neho potrebné dáta. V neposlednom rade je nevýhodou neflexibilita takéhoto riešenia. Každý súbor s diktátom by totiž pred nahratím na server musel najprv integrovať všetky dáta pomocou špecialneho nástroja dostupného 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 je uložiť zvlášť na serveri.

Existujú dve možnosti ako ukladať audio súbory a všeobecne akékoľvek dáta. Prvý prístup ukladá dáta priamo do databázy ako BLOB[4]. 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 [7]. Rozhodol som sa preto oddeliť súbory s diktátmi od databázy a ponechať tam iba odkazy. Viac o konkrétnom riešení s použitím databázy Postgres a aplikačného servera Wildfly je uvedené v kapitole 6. Nasadenie systému.

---

3. Do súboru mp3 je možné uložiť rôzne atribúty, nazývané tagy, ktoré obsahujú napr. názov interpreta, názov skladby, rok vydania atď.
4. dátový typ pre bližšie nešpecifikované binárne dáta v databáze

3.7 Prehrávanie diktátov

3.7.1 Diktát

Diktát je podľa Prášilovej[22] jeden z mnohých druhov pravopisných cvičení. Je to veľmi špecifická možnosť, ako skontrolovať u žiaka úroveň jeho pravopisného prejavu. Je to podľa nej však i prostriedok cvičebný, 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[**?**] 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 precvičovaní diktátu na internete, sú uplatnené aj pri vývoji portálu.

3.8 Nahrávanie diktátu na strane používateľského rozhrania

K implementácii nahrávania diktátu na strane používateľského rozhrania bol použitý modul pre framework AngularJS s názvom *ng-file-upload*. Medzi jeho hlavné funkcie patrí ukazovateľ postupu nahrávania, možnosť nahrať súbor presunutím na zvolené miesto na stránke (tzv. *drag and drop*, možnosť filtrovania zvolených typov súborov podľa koncovky a maximálnej veľkosti súboru, jednoduché

prepojenie s REST rozhraním na strane servera a podpora starších prehliadačov[33].

V diktátovom systéme bolo použité filtrovanie a drag and drop

3.9 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[34]. 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.

3.10 Zabezpečenie aplikácie

3.10.1 Zabezpečená komunikácia pomocou HTTPS

Základom bezpečnosti akéhokoľvek webového systému je zabezpečená komunikácia. Dáta posielané užívateľom cez sieť v rámci interakcie so systémom vrátane mena, hesla a iných citlivých informácií, by bez nej boli vystavené potenciálnemu riziku odcudzenia. Takýto útok sa nazýva Man in the middle attack[5]. Je tomu však možno zabrániť zašifrovaním komunikácie s 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 [2].

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

Vo vyvíjanej aplikácii, ktorá je nasadená na službe 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 [3].

3.10.2 Basic autentifikácia a AngularJS

Podľa špecifikácie aplikácie uvedenej v kapitole 2.2.1 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 Basic autentifikácia používateľa medzi klientom a serverom.

Podľa servera `spaghetti.io`[18] je Basic autentifikácia jedna z najrozšírenejších a najviac používaných autentifikačných metód. Aby bolo možné implementovať túto funkcionalitu, musí byť na strane klienta do každej požiadavky pridaná hlavička obsahujúca Base64[6] reťazec vo formáte `Authorization:Basic username:password`. 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 (kapitola 3.10.1).

Implementácia Basic autentifikácie na strane klienta vychádza z návodu od Watmorea[19]. Je realizovaná tak, že sa najprv odošle požiadavka na konkrétny zdroj s danou HTTP metódou. 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[7]. 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.

3.10.3 Zabezpečenie ciest na úrovni AngularJS frameworku

Zabezpečenie prezentačnej vrstvy pred neoprávneným prístupom používateľa je rovnako dôležité ako zabezpečenie časti aplikácie starajúcej sa o prístup k dátam. Ak je používateľ neprihlásený, alebo prihlásený pod rolou, ktorá k uvedenej adrese nemá mať prístup,

6. Číslovací systém založený na 64 rôznych znakoch využívajúci veľmi jednoduchý kódovací/dekódovací algoritmus. Neponúka jednosmernú šifrovaciu funkciu ako napr. SHA1
7. dáta získané od webovej stránky, uložené v prehliadači používateľa

musí byť zamedzený a používateľ primerane oboznámený.

3.10.4 Autentifikácia pomocou sociálnych sietí

Existuje hneď niekoľko možností, ako vyriešiť prihlasovanie do webovej aplikácie pomocou sociálnych sietí, píše Carvajal[29] na svojom technologickom blogu. Popis technológie ako aj 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 autentifikačný[8] protokol, ale protokol slúžiaci na autorizáciu[9]. Nižšie je uvedené ako je možné OAuth použiť aj na autentifikáciu používateľa.

Možnosti integrovania sociálnej autentifikácie

Carvajal na svojom blogu rozoberá tri autentifikačné schémy.

Prvá možnosť je nechať užívateľa 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`), tajomstva aplikácie a získaných používateľských informácií. Najdôležitejšie pri tejto autentifikačnej metóde je, aby nebolo zverejnené id a tajomstvo aplikácie resp. hashovacia

8. Autentifikácia = identifikácia používateľa na základe mena a hesla
9. Autorizácia = povolenie prístupu k určitým informáciám o používateľovi

funkcia. Výhody uvedeného prístupu spočívajú v tom, že používateľ si nepotrebuje pamätať ďalšie heslo, riešenie je relatívne jednoducho implementovateľné a tiež to, 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 autentifikácie 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. Výhoda oproti predchádzajúcemu riešeniu je v tom, že je nutné explicitne overiť identitu, nestačí prístup k sociálnej sieti. Nesporná výhoda je tiež jednoduchosť implementácie a expanzie z klasickej prihlasovacej schémy meno/heslo. Nevýhodou je nutnosť pamätať si ďalšie heslo.

Posledným zmieneným autentifikačným mechanizmom 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[10].

10. Viac ohľadom implementácie na `http://digitalleaves.com/blog/2014/07/building-your-own-rest-api-with-oauth-2-0-iii-hands-on/`

Zvolený prístup pre Diktátový systém

Pre navrhovaný systém bola zvolená hybridná autentifikácia pomocou sociálnych sietí. Dôvodov je hneď niekoľko. Prvým dôvodom sú problémy v návrhu systému popisované v kapitole 3.1.2. Taktiež bol tento prístup zvolený preto, že používateľ (žiak) musí spolu s heslom zadávať pri prvom prihlasovaní i školu a triedu, keďže tieto informácie sa zo sociálnych sietí získať nedajú (viac v kapitole 3.11). Prihlasovanie pomocou sociálnych sietí teda 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 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

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 email zo sociálnej siete.

Implementácia sociálnej autentifikácie

Implementácia sociálnej autentifikácie pozostávala z dvoch častí. Na strane servera bolo nutné vytvoriť REST rozhranie pre každú sociálnu sieť zvlášť. Rozhranie reprezentuje logiku autentifikácie popísanej v predchádzajúcej sekcii. Na strane užívateľského rozhrania je použitý modul frameworku AngularJS s názvom *Satellizer*. Je to autentifikačný modul so vstavanou podporou pre Facebook, Google, LinkedIn a iné sociálne siete[5].

3.10.5 Cross Domain Resource Sharing (CORS)

Jedným z požiadaviek na aplikáciu (kap. 2.2.1) je vytvoriť rozhranie prístupné ostatným doménam, ktoré bude schopné opraviť užívateľský text podľa zadaného originálu. Túto funkciu bude môcť v budúcnosti využívať napr. Informačný systém Masarykovej univerzity vo svojich projektoch. Keďže ide o prístup k dátam a zdrojom aplikácie zvonku, je nutné riešiť tzv. Cross Domain Resource Sharing[11].

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[12], ktorý pozdrží odoslanie odpovede servera do prehliadača až do okamihu, keď je povolený prístup na základe hlavičiek[11]. Správny formát hlavičiek znižuje bezpečnostné riziká spojené s CORS, predovšetkým CSRF útok[13][17]. Definícia hlavičiek spolu s dokumentáciou sa nachádza na stránkach pracovnej skupiny W3C[12].

Povolenie prístupu musí byť definovaná na strane servera. Z návrhu aplikácie vyplýva, že je nutné povoliť jediný zdroj slúžiaci na opravu diktátov, využívajúci metódu POST. Java ponúka hneď niekoľko možností.

Prvá možnosť je použitie filtru, ktorý implementuje rozhranie `ContainerResponseFilter`[16][15]. Tento spôsob však povoľuje prístup pre všetky zdroje, preto je pre aplikáciu nevhodný. Druhé riešenie je pridať prístupové hlavičky priamo pri budovaní odpovede spolu s telom odpovede a návratovým kódom. Výhoda tohto rieše-

11. V preklade: Zdieľanie zdrojov naprieč doménami
12. angl. Preflight request
13. 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

nia je, že umožňuje vybrať podmnožinu dostupných zdrojov u ktorých bude povolený CORS. Bohužiaľ, tento postup nie je možné použiť pre HTTP metódu POST, pretože ak je požiadavka vykonaná z inej domény, hlavičky sa nepridajú a odpoveď skončí s chybou[13].

Riešenie spĺňajúce obmedzenia aplikácie využíva externú knižnicu[14], 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[13].

3.11 Správa používateľov systému

Jednou z funkčných požiadaviek na aplikáciu (kapitola 2.2.1, požiadavka FP2) bola implementácia správy používateľov systému. Správu všetkých používateľov systému zabezpečuje administrátor systému, ktorý je po nasadení aplikácie na server ručne pridaný do databázy. Má povolené upravovať, poprípade 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 najmenej 0, najviac n školských tried, kde n je počet všetkých školských tried v systéme. Školská trieda je definovaná ako názov triedy (napr. 4.A, alebo Kvinta) a názov školy (napr. Gymnázium Ľ. Štúra, Trenčín). Táto dvojica atribútov je vrámci systému unikátna. V aplikácii môžu 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 mazať.

3.12 Návrh používateľského rozhrania

4 Modul na opravu diktátov

4.1 Analýza

4.2 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 vložený 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`[10] 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 [9]. V nasledujúcom texte bude ako správny označený znak alebo slovo vyskytujúce sa v správnom texte diktátu z databázy a chybným znakom alebo slovom rozumieme podreťazec textu vloženého používateľom, ktorý sa nezhoduje so správnym znakom resp. slovom. Slovo je v tomto texte synonymum k výrazu token[1] s výnimkou, že slovo môže pozostávať s dvoch tokenov, vlastného slova a interpunkčného znaku. 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

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

vracia pozíciu prvého chybného znaku, v prípade nadbytočného znaku je pozícia 0, v prípade chýbajúceho -1.

- `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` - pozíciu chyby v diktáte, použiteľnú napr. pri výpise chýb konkrétneho slova. Pozícia je definovaná ako číslo tokenu v rámci diktátu, číslovaná od 1, 0 znamená nadbytočné slovo, -1 chýbajúce slovo
- `lemma` - základný tvar slova získaný z výstupu morfosyntaktického analyzátoru Majka[23], 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[24], 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.

4.3 Značkovanie chýb

Formát značkovania vychádza z návrhu prezentovanom v mojej bakalárskej práci [8]. 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.

4.4 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[9]. 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 orientovaný databázový systém[21]. 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.

7 Záver

Čo sa podarilo, čo sa nepodarilo a prečo, splnilo sa oficiálne zadanie?

Literatúra

[1] 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/>.

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

[3] 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>.

[4] 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/>.

[5] 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/>.

[6] 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>.

[7] 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

[8] 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.

[9] 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.

[10] 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>.

[11] 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>.

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

[13] 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>.

[14] 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>.

[15] 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>.

[16] 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/>.

[17] 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>.

[18] 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>.

[19] 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>.

[20] 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>.

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

[22] Prášilová, Romana. *Návrh algoritmu pro vyhodnocení bezkontextových pravopisných chyb* České Budějovice, 2014 Dostupné z: <http://theses.cz/id/29k3zb/DP-Prilov.pdf>. Diplomová práca. Jihočeská univerzita v Českých Budějovicích.

[23] DOPLNIT CITACIU Dostupné z: <xyz>.

[24] DOPLN. *DOPLN* [online]. 2050 [cit. 2015-11-13]. Dostupné z: <https://nlp.fi.muni.cz/projekty/ajka/tags.pdf>.

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

[26] 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/>.

[27] 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>.

[28] 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>.

[29] 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/>.

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

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

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

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

http://goldfirestudios.com/blog/104/howler.js-Modern-Web-Audio-Javascript-Library
http://stackoverflow.com/questions/22684037/how-to-configure-wildfly-to-serve-static-content-like-images https://blog.openshift.com/multipart-forms-and-file-uploads-with-tomcat-7/

http://stackoverflow.com/questions/32008182/wildfly-9-http-to-https https://forums.openshift.com/changes-to-standalonexml-file

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.

HTTP metóda	Cesta	Povolené role	Návratový kód
POST	/api/correctDictate	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

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

A.2 Manuál ku grafickému užívateľskému rozhraniu

B Snímky obrazoviek

C Zoznam jednotlivých typov chýb

Nižšie je uvedený zoznam možných typov chýb, ktorý bol jedným z výstupov bakalárskej práce Vojtěcha Škvařila[9] 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 Obsah CD