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. Zahŕňa to taktiež vytvorenie samostatného modulu vrámci aplikácie, ktorý bude verejne prístupný skrz rozhranie. To bude zabezpečovať opravu diktátu a bude použiteľné aj pre prípadné iné systémy.

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. It also contains creation of an independent module inside the app, that will be public and will correct the dictate based on the input text using proper endpoint. This module will also be reusable for other systems.

Kľúčové slová

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

Obsah

1 Úvod

Informačné technológie sa v súčasnosti dostávajú do stále väčšieho množstva odvetví ľudskej činnosti. Aj do takých, v akých by sme ich ešte pred pár rokmi nečakali. A v mnohých 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. Celá táto evolúcia vzdelávania poskytuje doteraz netušené možnosti a to skutočne pre každého kto má záujem. Je dokonca možné absolvovať vopred vybrané kurzy a získať certifikát, ktorý si 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. Samozrejme už v súčasnosti existujú viac či menej kvalitné alternatívy (viď kapitola 2).

Kvôli neexistencii dostačujúcej aplikácie pre trénovanie diktátov za pomoci počítača, je v následujúcom texte navrhnutý webový portál, ktorý slúži ako 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 podujmú na funkčnom testovaní. Jednak sú to vybrané základné školy a jednak sa bude systém využívať na Pedagogickej fakulte Masarykovej univerzity pri výuke budúcich pedagógov.

2 Analýza systému

2.1 Definícia problému

2.2 Existujúce riešenia

2.2.1 Štruktúra existujúcich riešení

2.2.2 Problémy v existujúcich riešeniach

2.2.3 Výhody a nevýhody existujúcich riešení

2.3 Navrhovaný systém

2.3.1 Ciele navrhovaného systému

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.

2.3.2 Rozsah navrhovaného systému

2.3.3 Výhody a nevýhody navrhovaného systému

3 Návrh systému

Webový systém bude rozdelený na dve úplne oddelené časti.

Vrstvu prístupu k dátam, označovaná anglickým slovom backend, ktorá sa bude starať o ukladanie dát a navonok bude možné k nej pristupovať pomocou tzv. REST endpointov – známych adries, ktoré po prijatí dát v správnom formáte v JSON vrátia požadované údaje v definovanej forme. Rovnakým spôsobom sa bude pristupovať aj k samostatnému modulu slúžiacemu na opravu chýb. Adresy budú zdokumentované v jednej z následujúcich kapitol, rovnako ako správny formát vstupných respektíve výstupných dát.

Druhou časťou bude prezentačná vrstva, známa aj pod anglickým ekvivalentom frontend, ktorá bude komunikovať s vrstvou prístupu k dátam skrz REST rozhrania. Tieto budú zabezpečené aj na strane backendu aj na strane front-endu proti neoprávnenému prístupu neautorizovaného používateľa. Ten sa bude musieť na získanie oprávnenia zaregistrovať. Registrovať sa budú môcť iba užívatelia role STUDENT (žiaci), učiteľ (rola TEACHER) bude musieť požiadať o vytvorenie účtu administrátora (rola ADMINISTRATOR). Overenie prístupu bude prebiehať 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 bude navrhnutá ako tzv. jednostránková aplikácia[1].

1. Single Page Application, skrátene SPA

3.1 Ukladanie servisných informácií o diktátoch

Servisnou informáciou o diktáte rozumieme všetky dáta uložené spolu s diktátom, ktoré ho nejakým spôsobom definujú. Ide konkrétne o textový prepis diktátu, značky koncov jednotlivých viet, východzí počet opakovaní viet ako aj celého diktátu a dĺžka pauzy medzi jednotlivými vetami. V bakalárskej práci bol systém navrhnutý tak, že boli všetky tieto informácie uložené vrámci mp3 súboru s diktátom [8]. Tento prístup má však niekoľko nevýhod. Jednak je obmedzená použiteľnosť iných zvukových formátov na mp3, pretože sa informácie ukladajú do ID3v2 tagu[2]. Druhá nevýhoda je, že tento návrh komplikuje prípadnú budúcu úpravu systému a spracovanie napr. pre účely štatistiky. V neposlednom rade je nevýhodou nemožnosť použitia iného zdroja pre nahrávanie diktátu do systému skrz rozhranie kvôli nutnosti nakoniec uložiť všetky dáta do mp3.

Toto všetko sú dôvody, prečo sú nakoniec všetky servisné dáta uložené v databáze spolu s autorom a názvom súboru s diktátom a zvukový záznam je uložený zvlášť na serveri. Viac o uložení súboru na serveri v kapitole 4.2.1.

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

4 Implementácia systému

Pri implementácii systému bol 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.

4.1 Štruktúra aplikácie

4.1.1 Rozdelenie aplikácie na moduly

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

4.1.2 Adresárová štruktúra

4.2 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 tabuliek v databáze na Java objekty, tzv. entity. Entita je trieda reprezentujúca typicky jednu tabuľku v databáze. Obsahuje atribúty, ktoré sú ekvivalentné stĺpcom v databázovej 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 je postavená na poslednej spomínanej knižnici.

4.2.1 Ukladanie audio súborov s diktátmi

Existujú dve možnosti ako ukladať audio súbory a všeobecne akékoľvek dáta. Prvý prístup ukladá dáta priamo do databázy ako BLOB[1]. 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.

4.3 Bussiness vrstva aplikácie

4.4 Vrstva rozhraní - REST

4.5 Prezentačná vrstva

4.5.1 Angular JS JavaScript framework

4.5.2 CSS a Bootstrap

4.5.3 HTML5

1. dátový typ pre bližšie nešpecifikované binárne dáta v databáze

4.6 Zabezpečenie aplikácie

4.6.1 Zabezpečená komunikácia pomocou HTTPS

Základom bezpečnosti akéhokoľvek webového systému je zabezpečená komunikácia. Dáta posielané užívateľom cez sieť vrámci 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[2]. 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].

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].

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

4.6.2 Basic autentifikácia a AngularJS

Podľa špecifikácie aplikácie uvedenej v kapitole 2.3.1 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[3] 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 4.6.1).

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 cookie[4]. 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.

4.6.3 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, musí byť zamedzený a používateľ primerane oboznámený.

---

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

4.6.4 Autentifikácia pomocou sociálnych sietí

4.6.5 Cross Domain Resource Sharing (CORS)

Jedným z požiadaviek na aplikáciu (kap. 2.3.1) 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[5].

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[6], 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[7][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

5. V preklade: Zdieľanie zdrojov naprieč doménami
6. angl. Preflight request
7. 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 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].

4.7 Modul na opravu diktátov

Modul na opravu diktátov je nezávislý od ostatných modulov aplikácie a pozostáva so značkovača vstupného textu od užívateľa a definície pravidiel, podľa ktorých bude určený typ chyby a jej popis. Značkovač je vo východzej verzii implementovaný za pomoci knižnice `diff-match-patch` od Google[10], slúžiacej na porovnávanie reťazcov. Definícia pravidiel vychádza z výstupu bakalárskej práce Vojtěcha Škvařila [9] a je prepísaný z pseudokódu do programovacieho Jazyka Java.

Prístup k modulu zabezpečuje verejné REST rozhranie, ktoré ako vstup berie dva reťazce - správny text diktátu a text napísaný užívateľom (viac v prílohe A.1). Výstup vo formáte JSON obsahuje list jednotlivých chýb zoradených podľa výskytu v diktáte.

Chyba je objekt typu `Mistake`. Obsahuje nasledovné atribúty:

- Jednoznačný identifikátor `id`
- Pozíciu chyby (Pozícia je definovaná ako číslo tokenu[8] vrámci diktátu.)
- správne slovo alebo frázu
- prioritu slova (číslo od 1 do 10, pričom 10 je najväčšia priorita)
- typ chyby - chýbajúce slovo, nadbytočné slovo, chyba
- 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` a ukladá ich do databázy spolu s informáciami o používateľovi a diktáte ako objekt typu `Error`. Návrh bol zvolený z toho dôvodu, aby bol opravovací modul čo najjednoduchšie použiteľný aplikáciami tretích strán (preto má objekt typu `Mistake` iba základné atribúty, ktoré je možné vyčítať z daných vstupných dát). Objekt typu `Error` je ukladaný do databáze

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

a zamýšľané použitie dodatočných informácii je na generovanie štatistických dát o jednotlivých používateľoch a diktátoch.

5 Testovanie systému

5.1 Jednotkové testy

5.2 Integračné testy

mvn clean install -PintegrationTests-wildfly

5.3 Manuálne testovanie

6 Nasadenie systému

6.1 Databázový systém

6.2 Aplikačný server

7 Záver

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>.

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,
                  wordPosition: String,
                  correctWord: String,
                  writtenWord: String,
                  priority: Long,
                  mistakeType: enum,
                  mistakeDescription: String
                }]
}
```

Listing A.2: Telo odpovede

Viac o tom, čo jednotlivé atribúty znamenajú je uvedené v kapitole 4.7.

A.2 Manuál ku grafickému užívateľskému rozhraniu

B Snímky obrazoviek