VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

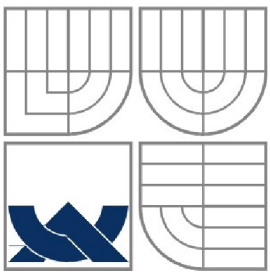
WEBOVÝ PROHLÍŽEČ AUDIO/VIDEO ZÁZNAMŮ PŘEDNÁŠEK: PŘEVOD PROHLÍŽEČE NA MYSQL DATABÁZI

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

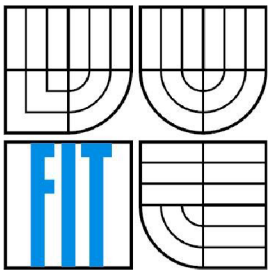
AUTOR PRÁCE
AUTHOR

Bc. JAKUB JANOVIČ

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

WEBOVÝ PROHLÍŽEČ AUDIO/VIDEO ZÁZNAMŮ PŘEDNÁŠEK: PŘEVOD PROHLÍŽEČE NA MYSQL DATABÁZI

WEB BASED AUDIO/VIDEO LECTURE BROWSER: PORTING ON THE BROWSER TO MYSQL DATABASE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAKUB JANOVIČ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. IGOR SZÖKE

BRNO 2010

Abstrakt

Tato práce se zabývá webovým prohlížečem, jehož cílem je zjednodušit získávání znalostí s využitím multimédií. Je tu představený již existující prohlížeč přednášek vytvořený jako diplomový projekt v rámci FIT VUT Brno. Demonstrované jsou technologie, které jsou v něm využity, a které budou použity na převod prohlížeče na MySQL databázi a vztvoření modulu úprav přepisu řeči. Čtenář se dozví o analýze a návrhu nového modelu aplikace. Dále jsou probrány implementační metody vývoje a následného testování systému. Na konci práce je zhodnocení s budoucím vývojem webového prohlížeče přednášek.

Abstract

This project deals with a web-based lecture browser, whose goal is to simplify the gaining of knowledge with the use of multimedia. It presents an existing lecture browser that was created for a diploma thesis at FIT VUT Brno. Demonstrated are the technologies that are used and which will be used to migrate the browser to a MySQL database and to develop a transcription module for speeches. The reader will be acquainted with an analysis and model of the new application. Furthermore, implementation methods for development and subsequent testing are discussed. At the end of the project is a conclusion about the future development of web-based lecture browsers.

Klíčová slova

zpracovnání řeči, streaming, multimédia, MySQL, XML, parser, rozdíl řetezců

Keywords

speech processing, streaming, multimedia, MySQL, XML, parser, strings difference

Citace

Bc. Jakub Janovič: Webový prohlížeč audio/video záznamů přednášek: převod prohlížeče na MySQL databázi. Brno, rok, diplomová práce, FIT VUT v Brně.

Webový prohlížeč audio/video záznamů přednášek: převod prohlížeče na MySQL databázi

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Igora Szökeho
Další informace mi poskytl Ing. Jozef Žižka
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Bc. Jakub Janovič
24.5.2010

Poďakovanie

Ďakujem svojmu vedúcemu Ing. Igorovi Szökemu a Ing. Jozefovi Žižkovi za rady, ktoré mi poskytli a za čas, ktorý mi venovali.

Obsah

Obsah....1
1 Úvod....3
2 Prehliadač....5
2.1 Použité technológie....5
2.1.1 Video....5
2.1.2 Streaming....6
2.1.3 Rečové technológie Speech@FIT....6
2.1.4 Spracovanie slajdov z videa....7
2.2 Doposiaľ realizovaný prehliadač prednášok....8
2.2.1 Komponenty prehliadača prednášok....8
2.2.2 Spôsob reprezentácie dát....8
2.2.3 Užívateľské rozhranie prehliadača prednášok....10
2.3 Technológie na prevod prehliadača na MySQL databázu....10
2.3.1 Návrh, modelovanie, UML....10
2.3.2 XML Parser....11
2.3.3 MySQL....11
2.4 Technológie na úpravu prepisov reči....11
2.4.1 Levenshtein distance....12
2.4.2 A*....13
2.4.3 Titulky....15
3 Analýza a návrh MySQL databáze....17
3.1 Analýza XML súborov....17
3.1.1 Jazyková mutácia....17
3.1.2 Kategórie....17
3.1.3 Prednášky....17
3.1.4 Slajdy a dokumenty....19
3.1.5 Transcript....19
3.2 Analýza ostatných súborov....19
3.3 Návrh štruktúry databázy....20
4 Analýza a návrh vytvárania korektúr prepisu reči....22
4.1 Korektúra prepisu reči....22
4.2 Analýza problému....22
4.2.1 Analýza vytvárania upravených segmentov....23
4.3 Návrh riešenia úpravy segmentov....23

4.3.1 Výber korigovanej časti....23
4.3.2 Úprava segmentu....25
4.4 Návrh ohodnocovania upravených segmentov....26
4.4.1 Hodnotenie opravy prepisov....27
4.5 Návrh vytvorenia nového segmentu....27
4.5.1 Nájdenie rozdielu reťazcov – levenshtein algoritmus....28
4.5.2 Hľadanie najefektívnejšej trasy – A*....29
4.5.3 Zmena trasy na transformačný vektor....30
4.5.4 Vytváranie výstupného reťazca....31
4.5.5 Pridanie časovania....31
4.6 Návrh vytvorenia titulkov....33
5 Implementácia....34
5.1 Prevod obsahu súborov do MySQL databáze....34
5.1.1 Browser parser....35
5.2 Prevod prehliadača prednášok na MySQL....36
5.3 Úprava transkriptov....37
5.3.1 Grafické užívateľské rozhranie....37
5.3.2 Výber a prepis segmentu....38
5.3.3 Hodnotenie segmentu....38
5.3.4 Implementácia úpravy segmentov....39
5.3.5 Vytvorenie nových segmentov....40
5.3.6 Vytvorenie titulkov....40
6 Testovanie a výsledky....41
6.1 Testy vytvárania nových prednášok....41
6.2 Testy vytvárania nových transkriptov....42
6.3 Prieskum verejnej mienky....43
6.4 Zhodnotenie....45
6.4.1 Testy vytvárania nových prednášok....45
6.4.2 Testy vytvárania nových transkriptov....45
6.4.3 Prieskum verejnej mienky....45
7 Záver....46
Literatúra....47
Zoznam príloh....49
Príloha 2 – Obrazový manuál pre vkladanie nového segmentu a jeho hodnotenie....50

1 Úvod

Metodika z pedagogického hľadiska je odpradávna skúmaná náuka, ktorá sa postupom času vyvíjala. Z počiatku sa človek vyučoval pomocou predávania skúseností z osoby na osobu, neskôr sa využívala papierová forma. Vynálezom Gutenbergovej kníhtlače sa vedomosti začali predávať pomocou kníh, a rôznych publikácií. Ľudia sa vždy snažili vzdelávanie zjednodušovať a zefektívňovať.

V dnešnej modernej dobe po nástupe elektronických zariadení ako sú napríklad počítače, sa snažíme využiť ich silu aj pri vzdelávaní. Využívajú sa hypertextové dokumenty, e-knihy, animácie, atď.

V tejto práci je čitateľovi predstavený doposiaľ nerozšírený spôsob výuky kombináciou multimediálnych prvkov. Ak študent navštevuje nejaký kurz, má možnosť absolvovať prednášky. Ak sa ale z nejakých dôvodov na prednášku nemôže dostaviť, má možnosť pozrieť si ju z videozáznamu, samozrejme ak je takýto videozáznam dostupný. Ten je prínosom aj pre opakovacie účely.

Tento systém začína byť ale značne zaujímavý, ak má užívateľ k prednáške ľahko dostupné študijné materiály ako sú slajdy z prezentácie, prepis reči prednášajúceho, rôzne dokumenty s prednáškou súvisiace a pod.

A práve o takomto systéme je táto práca. Ide teda o združenie, rôznych textových a multimediálnych materiálov. Aplikácia bola vyvíjaná v rámci diplomového projektu na FIT VUT Brno, a touto prácou je prevádzaná na MySQL databázu a doplnená o možnosť úpravy prepisu reči (transkriptov).

V úvode sa čitateľ oboznámi s technológiami, ktoré sú v prehliadači použité a ktoré v rámci migrácie na MySQL a úpravy transkriptov použité budú. Popísané sú tu techniky parsovania XML dát, modelovania systémov a algoritmov na prácu s reťazcami pri vzájomnej transformácií jedného na druhý.

Nasleduje kapitola, ktorá súčasný systém analyzuje a vytvára nový model prehliadača prednášok, ktorý je založený na databáze. V nadväznosti na spomínané techniky je vytvorený návrh štruktúry databázy.

V ďalšej časti sa čítajúci oboznámi o návrhu spôsobu vytvárania úprav prepisov reči. Popísaná je tu analýza danej problematiky, rôzne návrhy riešenia a detailne analyzovaný ten, ktorý bude v aplikácii nakoniec použitý.

Kapitola 5 sa zaoberá implementačnou časťou diplomovej práce. Rozoberá sa Browser parser, čo je entita, ktorá prevádza XML dáta do databázy MySQL a prevodom prehliadača prednášok na aplikáciu používajúcu databázový systém. Nastolené sú implementačné postupy, ktoré sú použité na správny chod úpravy prepisov reči.

Obsahom záverečnej kapitoly je súbor testov, ktorými je vypracovaná časť prehliadača prednášok podrobená. Súčasťou je aj menší prieskum verejnej mienky, keďže ide o aplikáciu určenú širokej verejnosti.

Záverom sa práca zhodnotí a spomenú sa návrhy na možnú ďalšiu realizáciu.

Ako sa spomína v zadaní diplomovej práce, ide o dva ucelené prvky, a to prevod prehliadača na MySQL databázu a pridanie funkcionality pre prehliadač. Tieto dva celky aj rozdelili prácu na tomto projekte na časti, ktoré boli riešené samostatne, aj keď medzi nimi súvis bol. Prvá časť: prevod prehliadača na MySQL databázu bola spracovaná ako semestrálny projekt, a druhá ako diplomová

práca, aj keď pri realizácii jednej, či druhej museli byť uvažované prvky preväzujúce tieto časti dokopy.

Diplomová práca nadväzuje na semestrálnu v tom, že aplikačná časť využíva návrhy, ktoré sú spomenuté v semestrálnej práci. Ide najmä o návrh databázy MySQL.

V práci je využitá časť mojej bakalárskej práce, konkrétne vyhľadávacie postupy v stavovom priestore. Ostatné časti práce boli vytvorené v nadväznosti na poznatky z preštudovania materiálov spomínaných v časti Literatúra a vlastných skúseností.

Cieľom je vytvorenie fungujúceho prehliadača prednášok s dátami uloženými v databáze, ktorá bude nápomocná pri štúdiu. Pridaný bude modul vytvárania úpravy prepisov reči. Správny prepis reči ďalej poslúži na zlepšenie rozpoznávača, ktorý je vyvíjaný v skupine spracovania reči SpeechFIT na FIT VUT v Brne.

2 Prehliadač

Cieľom tejto kapitoly je prezentovať čitateľovi techniky na vytvorenie a uvedenie do prevádzky webový prehliadač prednášok. Taktiež sa zaoberá aplikáciou, ktorá bola v rámci FIT VUT Brno spracovaná, a z ktorej táto práca vychádza. V poslednom rade budú spomenuté prostriedky pomocou ktorých, bude aktuálny prehliadač migrovaný na databázu MySQL.

Práca je vo svojej podstate rozsiahly projekt, na ktorom pracuje veľa ľudí, skupina spracovania reči SpeechFIT, študenti bakalárskeho či magisterského štúdia a vedúci ich prác.

Aktuálna verzia je prehliadač, ktorý ako Diplomovú prácu vytvoril Ing. Jozef Žižka v roku 2009. Ide o funkčnú aplikáciu, ktorá je momentálne dostupná na webovej adrese http://www.lectures.cz/ resp. http://www.prednasky.com/.

Sú tu implementované hlavné prvky, ako je prehrávanie videa, titulky k nemu, prepis reči, zobrazenie slajdov k danej prednáške, posun videa na základe časovej osi alebo textu prepisu reči. Ďalším komponentom je vyhľadávanie vo videách, presnejšie v prepisoch reči s našepkávačom.

2.1 Použité technológie

Na vytvorenie webového prehliadača je potrebné analyzovať a použiť rôzne technológie na prehrávanie videa, jeho streaming, spracovanie reči, prípadne spracovanie slajdov a pod. Podrobnejší popis technológií je možné nájsť v technickej dokumentácii práce Ing. Jozefa Žižku, preto v tejto podkapitole nie sú rozoberané natoľko dopodrobna.

2.1.1 Video

Video je technika pre zachytávanie, zaznamenávanie, prehrávanie, prenos a obnovu pohyblivých obrázkov, používajúce elektronické signály alebo digitálne dáta. Spoločne označuje analógové a digitálne spôsoby ukladania obrazových záznamov. My sa budeme špecifikovať na digitálne záznamy [2].

Medzi jeho vlastnosti patrí:

- **frame rate** (počet snímkov za jednotku času): zvyčajne sa používa počet snímkov za sekundu(fps). Dnešný štandard sa pohybuje okolo hodnoty 25fps. Ľudské oko považuje ako plynulý pohyb hodnotu väčšiu ako 10fps
- **interleaving** (prekladanie): video je buď progresívne alebo prekladané – to znamená, že každý snímok je rozdelený na dva pol-snímky trvajúce polovicu času jedného snímku
- **resolution** (rozlíšenie): počet bodov (pixelov) v jednom snímku. Udáva sa v počte bodov šírky a výšky snímku pre digitálne formáty, pre analógové sa používa len počet bodov v riadku. Príklady: Televízia 576, 480 pre PAL resp. NTSC, 720 resp. 1080 pre HDTV
- **aspect ratio** (pomer strán): pomer šírky a výšky snímku použitého vo videu. Väčšinou 4:3, ale dnes je tento formát vytláčaný novým 16:9
- **bit rate** (dátový tok): množstvo digitálnych dát za časovú jednotku, zvyčajne Mb/s

Dnes sa do povedomia dostávajú tzv. multimediálne kontajnery, ktoré ako z názvu vyplýva obsahujú multimediálne dáta viacerých kódovaní. Napríklad video-sekvencia v mpg, viaceré audio bloky v mp3 (originálny zvuk + dabing), titulky a pod.

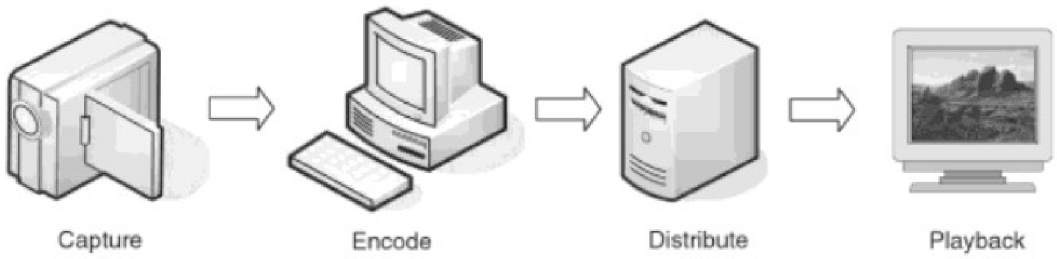
2.1.2 Streaming

Streaming médií prezentuje profesionálnu komunikáciu s novou formou doručovania informácií, správ a zábavy užívateľovi. S rozmachom počítačových systémov, je možné multimediálne dáta doručovať jednoduchšie a lacnejšie, než transport cez rôzne média [4].

Dochádza ku kontinuálnemu prenosu audiovizuálneho materiálu medzi zdrojom a koncovým užívateľom. V súčasnej dobe sa streaming využíva najmä na prenášanie audiovizuálneho obsahu po internete, ide o tzv. webcast.

Streaming prebieha v dvoch prípadoch:

- **On demand** (na vyžiadanie): audiovizuálny obsah je prenášaný z nejakého úložiska. Tento obsah je zaznamenaný už v minulosti, a ide teda len o „prehratie". Príkladom sú videoservery ako je Youtube prípadne videoserver FIT VUT Brno dostupný na https://video1.fit.vutbr.cz/
- **Realtime** (v reálnom čase): obsah je prenášaný „tak ako je zaznamenávaný", príkladom sú rôzne prenosy live udalostí, ako športové podujatia, prednášky a pod.



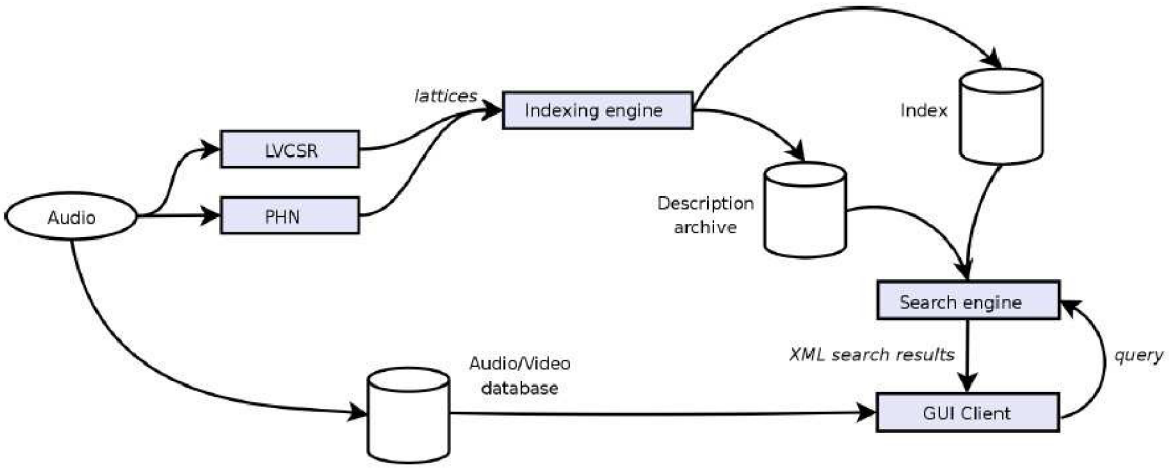
Obr. 2.1: 4 kroky streamovania audiovizuálneho obsahu [prevzaté z 4]

2.1.3 Rečové technológie Speech@FIT

Skupina spracovania reči na FIT VUT – Speech@FIT je aktívna na Ústave počítačovej grafiky a multimédií FIT pod vedením Doc. Jána Černockého.

Cieľom skupiny a projektu je rozpoznávanie jazyka a reči z audio záznamu. Systém používa pre spracovanie audia kombináciu prístupov založených na LVCSR a fonémovom rozpoznávači. Výstupné dáta sú indexované a uložené do databáz. Súbory s indexami sú uložené do *Index* databáze a lattice (orientovaný graf) do *Description archive* databáze. V systéme je ešte multimediálna databáza, z ktorej je možné načítať audio/video dáta.

Vyhľadávací stroj môže fungovať buď ako samostatná aplikácia, alebo neblokujúci server. V server móde klient pošle dotaz na adresu serveru (špecifikovaná IP adresou a portom) a ten mu odpovie výsledky uložené v XML formáte [1].



Obr. 2.2: Návrh vyhľadávacieho stroja v reči [prevzaté z 1]

2.1.4 Spracovanie slajdov z videa

Slajdy z videa sú extrahované pri splnení určitých podmienok. Ide o automatické spracovanie tým, že sa v obraze (videu) nájde priestor videoprojekcie slajdov. Tento priestor sa vyznačuje obdĺžnikovou plochou, ktorá môže byť určitým spôsobom transformovaná z dôvodu rôzneho umiestnenia kamery. Intenzita plochy plátna je zväčša výrazne odlišná od okolia. Takto získaná plocha sa premietne do 2D priestoru, čím sa získa postupnosť obrázkov. Následne je treba odstrániť duplicitné výskyty.

Obr. 2.3: Transformácia obrazu z videozáznamu do obrazu slajdu [prevzaté z 1]

Výstupom je časová os prednášky so zvýraznenými bodmi, kde došlo k zmene slajdu. Pre jednotlivé body sa extrahujú snímky a uložia do súboru. Týmto výstupom je možné spárovať aj slajdy vo formáte PDF s časovou osou. To sa realizuje tak, že jednotlivé slajdy sa prevedú na obrázok a porovnajú s extrahovanými súbormi. Touto kombináciou dostávame správne načasovanie zmien slajdov a obrázok samotného slajdu v lepšej kvalite (oproti slajdu získanému z videozáznamu).

2.2 Doposiaľ realizovaný prehliadač prednášok

V tejto podkapitole sa spomenie už realizovaný prehliadač prednášok od Ing. Jozefa Žižku. Je to na prvý pohľad intuitívne ovládateľná webová aplikácia, ktorá spĺňa typické vlastnosti, ktoré od prehliadača prednášok natívne požadujeme. Možnosti aplikácie môžeme rozdeliť do určitých blokov zvané komponenty.

2.2.1 Komponenty prehliadača prednášok

Ide o komponenty, ktoré postačia na základný chod prehliadača. Keďže sa jedná o rozsiahlejší projekt, ostatní spolutvorcovia prehliadača vyvíjajú iné komponenty. Návrh a interpretácia ďalších rozšírení je aj obsahom tejto práce, ale tým sa budeme venovať až v nasledujúcich kapitolách.

- **Videozáznam**: Ide o kľúčový komponent, keďže ide o samotné prehrávanie prednášky. Je riešená použitím JW FLV Media Player, čo je prehrávač vo formáte Flash (formálnejšie Macromedia Flash). Je možné k videu pripojiť titulky, ktoré sú uložené vo externom súbore vo formáte XML.
- **Hľadanie v audiu**: Pre tento komponent bola použitá technológia AJAX, ktorá zabezpečuje komunikáciu zo serverom, aj bez potreby znovu-načítať nepotrebný obsah web stránky. Vyhľadávanie sa teda vykonáva na pozadí.
- **Prepis reči**: Je to užitočný komponent hlavne na to, aby užívateľ pri nedostatočnom porozumení audio záznamu mohol pracovať s prednáškou a mal text prednášky v jednom súvislom celku. Ako výrazné plus tohto komponentu je to, že aktuálne prednášaný text je zvýraznený, a užívateľ v každom okamihu vie, kde sa prednášajúci vzhľadom na text prednášky nachádza. Na základe tohto komponentu je možné sa posúvať v časovej osi prednášky.
- **Slajdy**: Ide o zobrazenie slajdu, ktorý vo videu slúži študentom na ilustráciu prednášanej látky. Pomocou slajdov je možné sa vzhľadom na časovú os v prednáške tak isto pohybovať.
- **Dokumenty**: Zoznam dokumentov, ktoré súvisia s danou prednáškou
- **Príbuzné prednášky:** zobrazenie 2 prednášok, ktoré súvisia s danou prednáškou

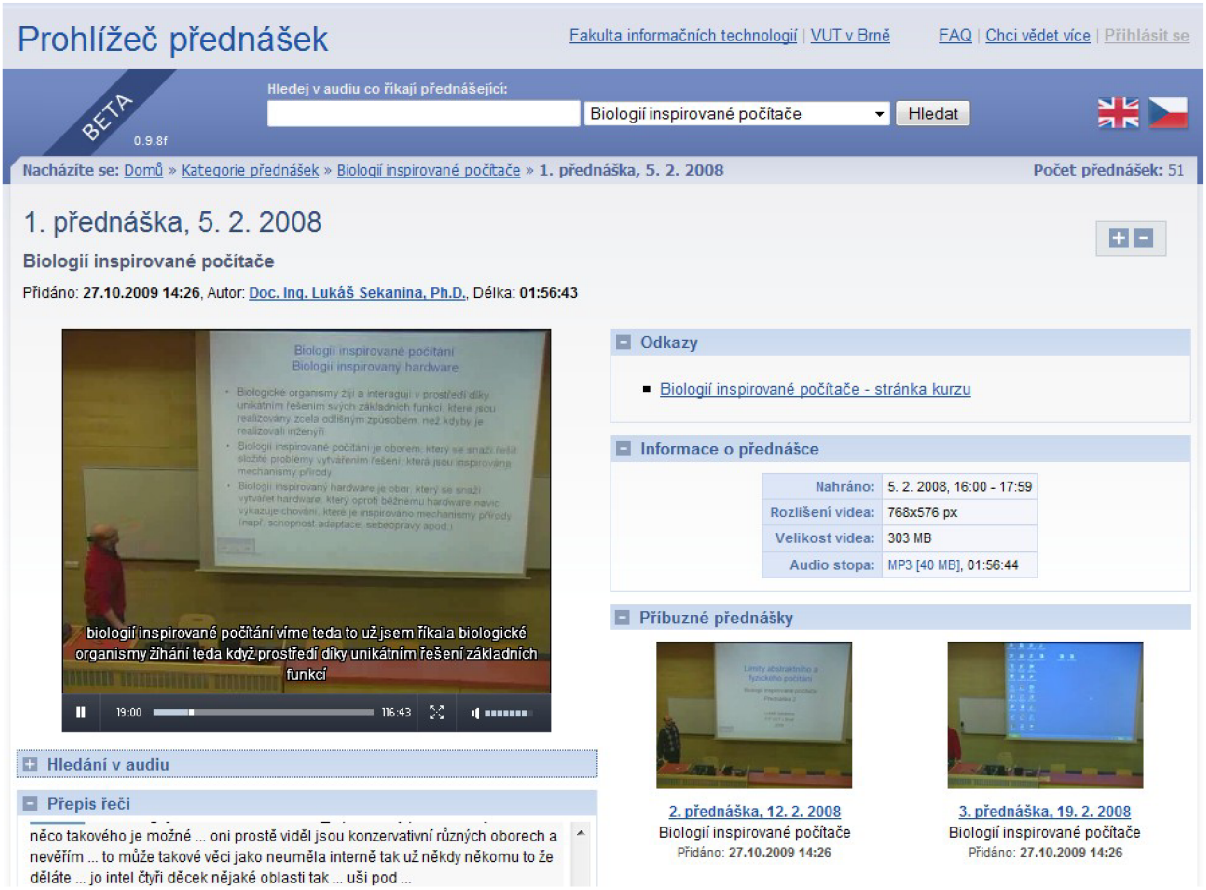
2.2.2 Spôsob reprezentácie dát

Všetky dáta sú rozdelené do hierarchie v súborovom systéme. Jedným z hlavných cieľov práce je práve migrácia textových dát do databáze MySQL kvôli lepšej manipulácii s nimi.

Hlavné nastavenia prehliadača sa nachádzajú v PHP skriptoch. Ostatné relevantné dáta sa sú v rôznych textových alebo multimediálnych formátoch.

Jazykové mutácie

V koreňovom adresári sa nachádza zložka *lang,* v ktorej sa vyskytujú jazykové mutácie prehliadača prednášok. Ide o štruktúrovaný dokument vo formáte XML. Celý dokument sa skladá z reťazcov, ktoré reprezentujú výraz v danom jazyku. Tento reťazec je uzavretý v XML tagu, ktorý je daný jednotne pre všetky jazykové mutácie.



Obr. 2.4: Ukážka grafického užívateľského rozhrania

```
<?xml version="1.0" encoding="utf-8" ?>
 <strings>
 ...
     <title>Prohlížeč přednášek</title>
     <contact>Kontakt</contact>
 ...
 </strings>
```

Kód 2.1: Spôsob zápisu jazykových mutácií

Ostatné

Ostatné súbory, ktoré korešpondujú s chodom prehliadača prednášok sa nachádzajú v zložke *data. T*u sú dáta pre prednášky a dva súbory *category_list.xml* a *lecture_list.xml.* V nich sú obsiahnuté vlastnosti kategórií resp. vlastnosti samotných prednášok.

V podzložkách pre dáta prednášok môžno nájsť súbory a zložky s vlastnosťami prednášky:

- slides (zložka): obsahuje obrázky slajdov prednášky a dokumenty patriace k prednáške
- video súbor: audiovizuálny súbor s prednáškou vo formáte mp3
- audio súbor: audio záznam prednášky vo formáte MP3
- textové súbory s rozličnými vlastnosťami, v ktorých sa nachádza prepis reči
- obrázky náhľadu prednášky

2.2.3 Užívateľské rozhranie prehliadača prednášok

Jadro prehliadača je implementované skriptovacím programovacím jazykom PHP s použitím značkovacieho jazyka XHTML a jazyka CSS na tvorbu štýlov. Tento postup je typický pre takmer všetky zložitejšie webové aplikácie. Použitie alternatívnych metód ako je Flash, by bolo neefektívne a zbytočne náročné.

Pozitívne možno hodnotiť použitie technológie AJAX, ktorá umožňuje komunikáciu klienta (webového prehliadača) so serverom bez nutnosti znovu-načítania obsahu stránky. Práca s prehliadačom je teda jednoduchšia. To je implementované pre vyhľadávanie, takže užívateľ nemusí prerušiť chod videoprezentácie prednášky a vyhľadávanie sa realizuje na pozadí.

Aplikácia je na prvý pohľad intuitívne ovládateľná. Používa typické rozloženie, aké užívateľ očakáva pri projekte ako je tento. Interakcia medzi užívateľom a systémom je realizovaná hlavne „klikateľnou“ formou (použitie myši). Použitie klávesnice je nutné len pri vyhľadávaní prednášok, alebo pri vyhľadávaní v rámci prednášky.

Použitá farebná škála (odtiene modrej a šedej) pôsobí nerušivo, a graficky dopĺňa celkový vzhľad prehliadača. Nevyskytujú sa tu zbytočné prvky, ktoré by boli zbytočne veľké a pod.

Použitie na mobilných zariadeniach ako sú mobilné telefóny PDA sa nepredpokladá, takže sa nemuselo brať ohľad na rozlíšenia týchto zariadení. Použité rozlíšenie minimálne 1024x768 je už dnes štandardom.

2.3 Technológie na prevod prehliadača na MySQL databázu

Na prevod prehliadača prednášok na prehliadač s použitím MySQL databázy sa museli zvoliť metódy a technológie, pomocou ktorých by bola zachovaná sémantika už vytvoreného systému s pridaním novo navrhnutých prvkov

2.3.1 Návrh, modelovanie, UML

UML je grafický jazyk pre vizualizáciu, špecifikáciu, navrhovanie a dokumentáciu programových systémov. Ponúka štandardný spôsob tvorby softvéru. Podporuje objektovo orientovaný prístup k analýze, návrhu a popisu programovacích systémov [5].

Návrh reprezentácie dát je jeden z kľúčových krokov prevodu prehliadača na MySQL, pretože spätné menenie štruktúry databázy by bolo časovo náročné a vysoko neefektívne. Preto je treba analýze a modelovaniu systému venovať veľkú pozornosť.

Pre modelovanie je možné použiť jeden z mnoho softvérov na modelovanie pomocou UML. Pre vývoj aplikácie bol použitý softvér Argo UML [6], konkrétne modul na tvorbu návrhu schémy databázy.

2.3.2 XML Parser

Prehliadač prednášok je postavený na XML súboroch. Na prevod do databázy je možné použiť prácu s reťazcami, ale pre lepšiu efektivitu sa používajú XML parsery [7].

- PHP ponúka široké možnosti práce s dokumentami XML:
- DOM extension – umožňuje operovať s XML dokumentami pomocou DOM API pomocou PHP5
- libxml – od PHP 5.1
- SimpleXML – ponúka jednoduchý a použiteľný prevod XML dokumentu na objekt
- XML parser – umožňuje jednoduché parsovanie s podporou kódovaní UTF-8, ktoré je aj použité v prehliadači prednášok

Na implementácie parsovania XML dokumentov bol použitý posledný spomenutý: XML parser. Bolo to z dôvodu jeho jednoduchosti, dostupnosti materiálov a najmä osobných skúseností. Ďalšou výhodou bolo, že je súčasťou inštalácie PHP modulu pre Apache, takže nie je nutná inštalácia pri migrácii prehliadača na iné servery.

2.3.3 MySQL

Ide o databázový systém, vytvorený švédskou firmou MySQL AB. Je považovaný za úspešného priekopníka v dvojnom licencovaní – je k dispozícií pod bezplatnou licenciou GPL, ale aj pod platenou licenciou [3].

Komunikácia s MySQL prebieha pomocou jazyka SQL. Je použiteľná multiplatformne, prečo sa stala veľmi obľúbenou. Jej hlavnou vlastnosťou je rýchlosť, a to aj za cenu niektorých zjednodušení: má jednoduché spôsoby zálohovania a donedávna ani neboli podporované pohľady, triggery a úložné procedúry [12].

V aplikácii je použitá na uchovávanie dát získaných z XML súborov. Pre jednoduchšiu správu MySQL databázy bolo použité rozhranie phpMyAdmin. Použitá je verzia MySQL 5.1, so znakovou sadou UTF-8-Unicode.

2.4 Technológie na úpravu prepisov reči

Pre korekcie prepisov reči bude použitý skriptovací programovací jazyk PHP, ktorý sa bude dotazovať na tabuľky transkriptov v databáze MySQL. Komunikáciu medzi klientom a serverom bude zaisťovať najmä vývojová technológia AJAX, aby zmeny vykonané na strane servera sa premietli užívateľovi v čo najprívetivejšej forme bez nutnosti znovu-načítania celých stránok.

Po vytvorení korekcie, presnejšie textového segmentu z prepisu reči, je nutné túto časť správne načasovať, aby slová, ktoré prednášajúci rozpráva časovo korešpondovali s textom reči. Na túto problematiku bude použitý algorimtus Edit distance známy ako Leveshtein distance algoritmus s použitím ďalších metodík popísaných nižšie.

Nakoniec bude vygenerovaný súbor pomocou PHP a MySQL, ktorý bude slúžiť ako vstup pre rozpoznávač reči.

2.4.1 Levenshtein distance

Levenshtein distance (voľne preložené ako Levenshteinova vzdialenosť) je metrika, ktorá je vypočítaná na základe podobností dvoch reťazcov. V jednoduchosti, vzdialenosť *D(x,y)* medzi reťazcom *x* a *y* je minimálny počet znakov vložených (insertion), zamenených (substitution), prípadne zmazaných (deletion) potrebný na transformáciu reťazca *x* na *y* [11].

Príklad

Na demonštráciu je uvedený príklad výpočtu Levenshteinovej vzdialenosti pri transformácii slova „*kitten*“ na slovo „*sitting*“. Výsledná hodnota je 3. Prepis sa teda udeje v 3 nasledovných krokoch[10]:

- kitten → sitten : použitá zámena (substitution) k za s
- sitten → sittin : použitá zámena (substitution) e za i
- sittin → sitting : použité vloženie (insertion) znaku g

		k	i	t	t	e	n
	0	1	2	3	4	5	6
s	1	1	2	3	4	5	6
i	2	2	1	2	3	4	5
t	3	3	2	1	2	3	4
t	4	4	3	2	1	2	3
i	5	5	4	3	2	2	3
n	6	6	5	4	3	3	2
g	7	7	6	5	4	4	3

		S	a	t	u	r	d	a	y
	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5	6	7
u	2	1	1	2	2	3	4	5	6
n	3	2	2	2	3	3	4	5	6
d	4	3	3	3	3	4	3	4	5
a	5	4	3	4	4	4	4	3	4
y	6	5	4	4	5	5	5	4	3

Obr. 2.5: Príklad výpočtu Levenshteinovej vzdialenosti [prevzaté z 10]

Na obrázku č. 2.5 je znázornený výpočet Levenshteinovej vzdialenosti medzi slovami *kitten* a *sitting*, resp. *Sunday* a *Saturday*. Výpočet začína v ľavom hornom rohu a porovnávajú sa zmeny v každom smere pohybu. Vkladanie, zámena aj mazanie majú rovnakú váhu, a to 1. Ak sú znaky totožné, tento krok sa ohodnotí nulou. Vyhodnotí sa každý prvok matice, aby sme sa dostali do pravého dolného rohu. Vtedy základný výpočet končí.

```
int LevenshteinDistance(char s[1..m], char t[1..n])
{
  // d is a table with m+1 rows and n+1 columns
  declare int d[0..m, 0..n]

  for i from 0 to m
    d[i, 0] := i // deletion
```

```
    for j from 0 to n
      d[0, j] := j // insertion

    for j from 1 to n
    {
      for i from 1 to m
      {
        if s[i] = t[j] then
          d[i, j] := d[i-1, j-1]
        else
          d[i, j] := minimum
                     (
                       d[i-1, j] + 1,  // deletion
                       d[i, j-1] + 1,  // insertion
                       d[i-1, j-1] + 1 // substitution
                     )
      }
    }

    return d[m,n]
}
```

Kód. 2.2: Algoritmus výpočtu Levenshteinovej vzdialenosti [prevzaté z 10]

2.4.2 A*

Z algoritmu Leveshtein distance je návratová hodnota počet zmien nutných na transformáciu jedného reťazca na druhý. Pre potreby generovania nových prepisov reči je ale potrebná celá matica, ktorá sa pri výpočte využíva. V nej sa potom vyhľadáva najkratšia cesta z pravého dolného rohu do ľavého horného. Na vyhľadanie trasy sa použil algoritmus A* [12].

Metóda A* (čítaj A s hviezdičkou, A star) je najznámejšou a najpoužívanejšou v oblasti riešenia úloh prehľadávaním stavového priestoru. Hľadá pomocou heureistiky a skúša vždy najsľubnejší neprebádaný stav.

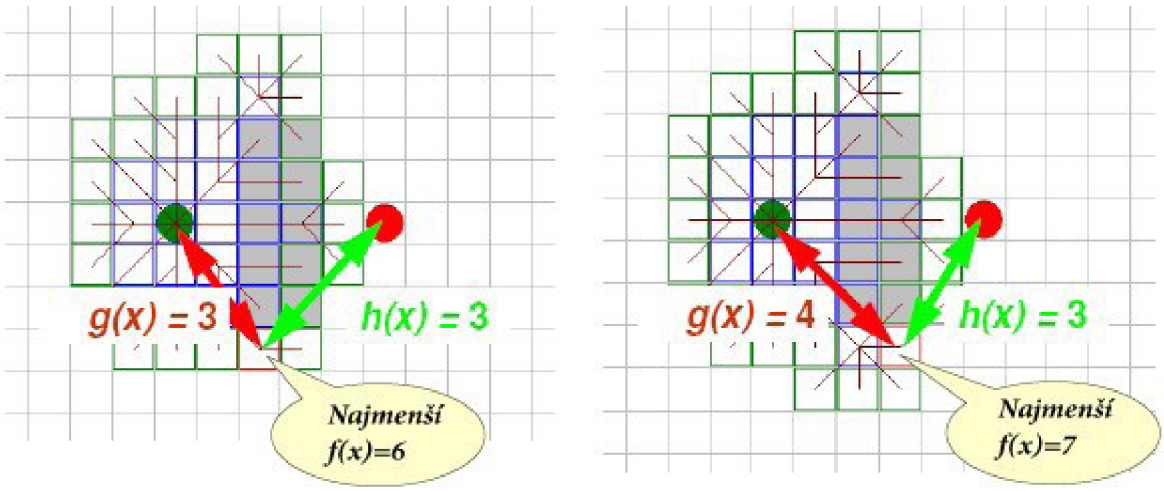
Pre každý navštívený vrchol x sa spočíta ohodnotenie najlepšej cesty $f(x)$, ktorá cez neho vedie podľa vzorca:

$$f(x) = g(x) + h(x)$$

Kde $g(x)$ je súčet ohodnotenia najkratšej cesty z počiatočného bodu cez doposiaľ navštívené vrcholy do x a $h(x)$ je odhad dĺžky cesty z x do cieľového bodu.

$$h(x) = 0 \quad \text{pre cieľový bod}$$
$$h(x) \geq 0 \quad \text{pre každý bod}$$

Na obrázku 2.6 je možné vidieť hodnoty $g(x)$ a $h(x)$ počas prehľadávania stavového priestoru. A zmena $f(x)$ počas tohoto prehľadávania.



Obr. 2.6: Príklad postupu algoritmu A* [prevzaté z 14]

V algoritme A* sa používajú dve dátové štruktúry a to open a closed. V štruktúre open (ďalej len OPEN) sa nachádza zoznam navštívených vrcholov, ktoré je potreba prebrať a štruktúra closed (ďalej len CLOSED) obsahuje už prejdené vrcholy.

Algoritmus A* teoreticky

1. Počiatočný bod vložíme do OPEN
2. Opakujeme následujúce:
    - Nájdeme bod s najmenším f(x) ohodnotením z OPEN a poznačíme ho ako aktuálny bod
    - Vyberieme ho z OPEN a vložíme do CLOSED
    - Pre každé okolité body preveríme následujúce:
        - Pokiaľ nie je priechodný, alebo sa nachádza v CLOSED ignorujeme ho
        - Pokiaľ sa nenachádza v OPEN vložíme ho do OPEN a aktuálny bod zaznamenáme u vkladaného ako rodičovský a zaznamenáme hodnoty h(x) a f(x)
        - Pokiaľ sa už v OPEN nachádza, pozrieme či táto trasa k tomuto bodu je lepšia. Použijeme na zistenie hodnotu g(x) – bod s nižšou hodnotou g(x) znamená lepšiu cestu. Pokiaľ dôjdeme na lepšiu cestu rodiča tohoto bodu zmeníme na aktuálny a prepočítame hodnoty g(x)
    - zastavíme pokiaľ:
        - Vložíme cieľový bod do CLOSED, v tomto prípade bola trasa úspešne nájdená
        - Cesta sa nenájde, a OPEN je prázdny, v tomto prípade neexistuje cesta medzi danými dvomi bodmi

3. Zistíme si celú trasu a to tak, že prechádzame CLOSED od cieľového bodu a zisťujeme jeho rodiča. Týmto spôsobom sa dostaneme až na počiatočný bod. Otočením tohoto poľa získame hľadanú trasu. Podrobnejší popis v [12].

Algoritmus A* v pseudokóde

```
function A*(start,goal)
      var closed := the empty set
      var q := make_queue(path(start))
      while q is not empty
            var p := remove_first(q)
            var x := the last node of p
            if x in closed
                  continue
            if x = goal
                  return p
            add x to closed
            foreach y in successors(x)
                  enqueue(q, p, y)
      return failure
```

2.4.3 Titulky

Titulky sú textová verzia dialógov a monológov v audiovizuálnom obsahu, ktorá je zobrazená na spodnej časti obrazovky. Zväčša prezentujú hovorenú reč v cudzom jazyku v jazyku poslucháča, ale sú používané taktiež ako prezentácia hovoreného textu v rovnakom jazyku, s tým že v nich môže byť pridaná informácia o deji - najmä pre sluchovo postihnutých.

Titulky v rovnakom jazyku, ako je audiovizuálny obsah, sa používajú aj pri prednesoch zložitejších informácií.

Pre účely tejto práce sú titulky použité najmä pre to, aby bolo možné v prednáškach realizovať vyhľadávanie, keďže nie je možné vyhľadávať v audiovizuálnnom obsahu ako takom. Ďalším dôvodom je už spomínané prezentovanie obsahu, ktorý prednášajúci rozpráva počas prednášky – ide často o zložitý prednes, ktorý môže byť poslucháčovi lepšie pochopiteľný, ak je sprevádzaný titulkami.

U filmového obsahu sú titulky vyrábané leptaním, alebo laserom vpisované do filmu. Naopak u digitálneho formátu sú priložené k videu v podobe ďalšieho súboru. Titulkový súbor sa vkladá do softvérových prehrávačov, alebo pomocou kodeku, ktorý sa titulky pokúsi sám nájsť.

V projekte webový prehliadač prednášok sú použité titulky vo formáte .tt. Ide o XML formát použitý na prehrávanie titulkov v použitom flash prehrávači videí JW Player. Pokiaľ ale by poslucháč mal záujem pozerať prednášku v tzv. „offline“ móde – teda bez použitia webového prehliadača, má možnosť si prednášku stiahnuť z videoserveru FIT VUT v Brne. Tento ale neposkytuje titulky k daným prednáškam, preto je v rámci tejto práce vytvorená možnosť titulky stiahnuť.

Titulky vo formáte .tt ale nie sú bežné pre dnešné prehrávače audio/video súborov. Medzi najpoužívanejšie patria titulky vo formáte SubRip a MicroDVD.

SubRip

SubRip je súborový formát obsahujúci titulky k audiovizuálnemu obsahu vo formátoch DivX, XviD apod., používa príponu .srt

```
1
00:02:25,600 --> 00:02:29,400
tak já sem se dneska vešel do standardně akademické
pětiminutovky

2
00:02:29,600 --> 00:02:32,080
minule na první přednášce to bylo s hardwarem trochu
náročnější takže vás pěkně vítám
```

Kód 2.3 Súbor vo formáte SubRip

Formát SubRip sa od formátu MicroDVD líši tým, že udáva počiatočnú a koncovú značku pre zobrazenie jednotlivých titulkov vo formáte časovej značky. Naopak u MicroDVD formátu je zobrazenie/nezobrazenie titulku viazané na určitý snímok z videa, ku ktorému titulky náležia.

- Výhody: V prípade že sa zmení počet snímkov za sekundu (Framerate), nie je potreba počiatočné a koncové značky prepočítavať.
- Nevýhody: Titulky majú zbytočne veľkú veľkosť súboru a zložitejšie spracovanie, pretože obsahuje nadbytočné dáta pre strojové spracovanie:
    - Číslo titulku
    - Šípka z počiatočnej po koncovú časovú známku

3 Analýza a návrh MySQL databáze

Táto kapitola sa zaoberá analýzou problému, rozdelením na podproblémy a návrhom riešenia daných podproblémov. Najskôr je nutné previesť rozbor XML súborov, na základe ich štruktúry vytvoriť databázový návrh.

3.1 Analýza XML súborov

V aplikácii je využitých viacero XML súborov. Obsahujú údaje o vlastnostiach prednášok, ale aj jazykové mutácie prehliadača.

3.1.1 Jazyková mutácia

Prehliadač prednášok je možné lokalizovať pre viaceré jazyky. Slúži na to zložka *lang,* v ktorej sú obsiahnuté XML súbory s rôznymi prekladmi aplikácie.

Tento súbor je stále rovnaký, používa sa len na čítanie. Preto sa ani na prevod do databáze ani neuskutočňoval. Išlo by o zbytočne komplikovaný proces, ktorým by sa lepšia efektivita nedosiahla. XML dokumenty s jazykovými mutáciami teda ostávajú nezmenené.

Ostatné XML súbory, ako bude znázornené v nasledujúcich podkapitolách, sa viažu už priamo k prednáškam.

3.1.2 Kategórie

Kategórie sú reprezentované súborom *category_list.xml* v zložke *data*.

```
<?xml version="1.0" encoding="utf-8"?>
<data>
    <category>
      <id>1</id>
      <name>Signály a systémy</name>
    </category>
</data>
```

Kód 3.1: Spôsob rozdelenia kategórií

Vyskytuje sa tu názov kategórie a id (identifikačné číslo) kategórie. V prehliadači bude použitá v rámci administračného rozhrania správa kategórií a rozloženie z XML nebude použité.

3.1.3 Prednášky

K prednáškam sú asociované dva rozličné XML súbory. V jednom sú uložené prednášky, ktoré sú obsiahnuté v celej aplikácii., v druhom dáta o samotnej prednáške.

lecture_list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<data>
 <lecture>
       <id>ISS-2008-09-26</id>
       <categ_id>1</categ_id>
       <datafile>data/iss/ISS-2008-09-26/ISS-2008-09-26.xml</datafile>
 </lecture>

 <lecture>
       <id>ISS-2008-10-03</id>
       <categ_id>1</categ_id>
       <datafile>data/iss/ISS-2008-10-03/ISS-2008-10-03.xml</datafile>
 </lecture>
   ...
</data>
```

Kód 3.2: Spôsob rozdelenia prednášok

Tento súbor obsahuje id prednášky, ktoré je dané jedinečným reťazcom skladajúcim sa z názvu predmetu a času kedy bola prednáška vytvorená. Ďalej je uvedené číslo kategórie a názov XML súboru, kde sa nachádzajú presnejšie dáta k prednáške. Tento súbor sa takisto nebude používať, opäť sa prednášky budú vytvárať z administračného rozhrania.

<prednáška>.XML

Súbor obsahuje dáta a odkazy na ostatné súbory, ktoré sa viažu na prednášku z *lecture_list.xml.*

```
<?xml version="1.0" encoding="utf-8"?>
<data>

 <title>2. přednáška, 3. 10. 2008</title>
 <description></description>

 <datetime>
       <recorded>3. 10. 2008, 10:00 - 12:59</recorded>
       <added>28.7.2009 11:02</added>
 </datetime>
 ...
 <files>
       <wordlist src="ISS-2008-10-03_wordlist.txt" />
       <subtitles src="ISS-2008-10-03_subtitles.tt" />
       ...
 </files>

</data>
```

Kód 3.3: Náčrt obsahu súboru s prednáškou

Vyskytujú sa tu textové dáta typu názov, popis, čas vytvorenia, a pod. a cesty k ďalším súborom, ktoré sa k prednáške viažu. Dáta budú parsované a následne vložené do databázy. Ďalej k nim budú pridané ostatné údaje z administračného rozhrania.

3.1.4 Slajdy a dokumenty

<prednáška>_slides.xml zahŕňa 2 entity a to zoznam dokumentov a zoznam slidov pre prednášku.

```
<?xml version="1.0" encoding="utf-8"?>
<data>

  <links>
    <a href="1-org_motiv.pdf">Program a organisace kursu</a>
    <a href="02-zakl_sig.pdf">Základní pojmy o signálech</a>
  </links>
  <slides>
    ...
    <slide start="493" src="slide04.png" thumbnail="slide04.png"
       caption="1 GB Ethernet - zdroj 3COM" />
  </slides>
<data>
```

Kód 3.4: Náčrt obsahu súboru so slajdami

Tento súbor bude parsovaný a dáta uložené do databázy.

3.1.5 Transcript

Obsahom tohto súbora sú frázy, ktorým sú pridelené časové razítka začiatku. Súbor má tvar *<prednáška>_transcript.xml* a bude použitý na napĺňanie databáze.

3.2 Analýza ostatných súborov

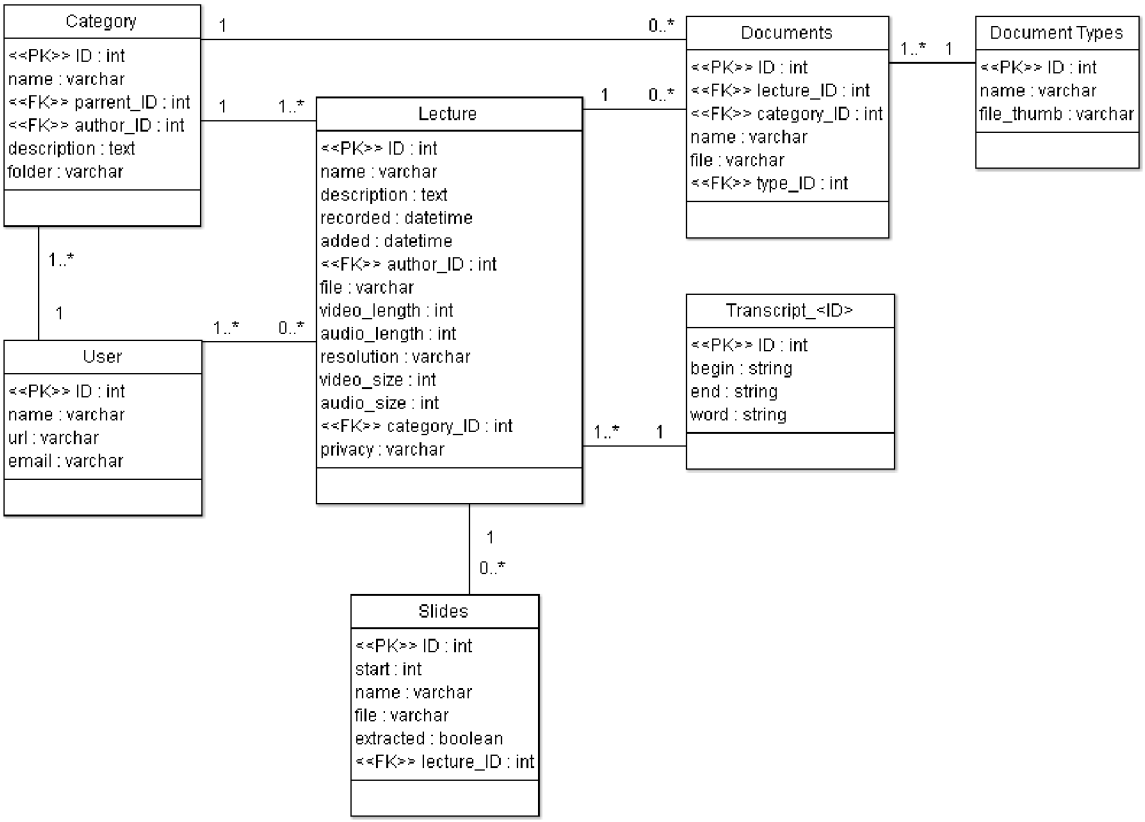
V tejto kapitole ukážeme ostatné súbory, ktoré súvisia s prednáškou.

<prednáška>.flv – obsahuje video vo formáte flash
<prednáška>.jpg – náhľad na prednášku
<prednáška>.mp3 – zvuková stopa prednášky
<prednáška>.xml – XML súbor prednášky
<prednáška>_large.jpg – veľký náhľad na prednášku
<prednáška>_slides.XML – XML súbor pre slajdy a dokumenty k prednáške
<prednáška>_subtitles.tt – titulky k prednške
<prednáška>_transcript.* - rôzne súbory viažuce sa na prepis prednášky
<prednáška>_wordlist.txt -zoznam slov a ich časový výskyt

Ide najmä o multimediálne dáta viažuce sa k prednáške, prípadne o textové, v ktorých sa nachádza prepis reči získaný z automatického rozpoznávača skupiny Speech@FIT.

3.3 Návrh štruktúry databázy

Návrh (Obr. 3.1) bol vytvorený z analýzy pomocou modelovacieho softvéru ArgoUML. Databáza sa skladá zo *7 + (k)* tabuliek. Hodnota *k* určuje počet prednášok v databáze, je to prepis reči z prednášky.



Obr. 3.1: Návrh schémy databáze

U každej tabuľky sa nachádza primárny kľúč (<<PK>>) ktorý je jedinečný, a má vlastnosť autoincrement, čo jeho jedinečnosť pri vkladaní údajov do databázy zaručuje.

Category

Táto tabuľka obsahuje jej názov, popis, koreňový adresár. Cudzí kľúč author_ID prezentuje autora, prípadne autorov kategórie. Kategórie sa môžu zanorovať, čo je zabezpečené cudzím kľúčom parrent_ID.

User

Tu sa vyskytujú základné údaje o užívateľovi ako sú meno, web adresa a email. Intuitívny názov „autor“ bol zavrhnutý, keďže sa počíta do budúcna s úpravou tejto tabuľky na možnosť pracovania s právami a zabezpečovanie autentifikácie užívateľov.

Documents

Ide o dátový sklad dokumentov, ktoré sa môžu viazať ako na prednášku, tak na kategóriu. Prvky tabuľky sa viažu na tabuľku **DocumentTypes**, v ktorej sú uložené typy dokumentov (názov a cesta k náhľadu ikony). Každý dokument má v tabuľke uložené meno a cestu k súboru.

Lectures

Ide o základnú tabuľku obsahujúcu údaje o prednáške. Prednáška sa viaže na kategóriu (nemôže existovať nezaradená prednáška), ďalším viazaným prvkom je autor/autori. Inak obsahuje informácie o audio a video stope, čas záznamu a pridania do systému. Poslednou položkou je *privacy*, ktorá je rezervovaná do budúcna, keďže sa počíta s možnosťou iných vlastností prednášky ako je súkromie apod. Privacy je reprezentovaná reťazcom `0` a `1`, ktoré pre každý „bit“ reťazca bude určovať true/false príznak. (Príklad: `110`: prvá „`1`“: povolenie prezerania, druhá „`1`“: povolené komentáre, tretia „`0`“: prístup len registrovaným užívateľom, a pod.)

Slides

Ku každej prednáške sa môžu viazať slajdy. O nich je v zázname uložený čas, kedy sa slajd v prednáške vyskytne, názov, cesta k súboru a príznak *extracted,* ktorý určuje či je slajd extrahovaný z videa alebo z dokumentu.

Transcript_<ID>

Transcript_<ID> znázorňuje množinu tabuliek prepisov reči. Je v nej uložené slovo z rozpoznanej prednášky, začiatok a koniec. Možnosť vytvorenia viacerých tabuliek pre „jednu“ entitu bola zvolená kvôli zložitosti systému. Ak by boli reťazce uložené v jednej tabuľke, dotazy na ňu by boli na výpočet veľmi náročné.

Z tejto tabuľky budú generované všetky súbory, ktoré sú potrebné na správny chod prehliadača vzhľadom na časť prepisu reči.

Dáta by mohli ostať v XML, ale v rámci aplikácie sa plánuje možnosť úpravy transkriptu. Úpravy transkriptu už boli v návrhu uvažované. K súčasným tabuľkám vznikne jedna tabuľka, nazvime ju *transcriptHistory*. Tá by obsahovala okrem *<<PK>>ID* upravený reťazec, ID originálu vety (väzba na id z *Transcript<ID>*), a hodnotu *index*. *Index* by znázorňoval o koľkú zmenu ide, čiže by bolo možné nahliadať do histórie úprav (podobný model ako wikipédia).

Väzby medzi tabuľkami databázy nie sú presnejšie popisované, pretože sú z obrázku (Obr. 5) návrhu zrejmé.

4 Analýza a návrh vytvárania korektúr prepisu reči

Táto kapitola sa venuje analýze požiadaviek a návrhu spôsobu možnosti úpravy prepisu reči, ktorý je rozpoznávaný automatizovaným rozpoznávačom.

Budú tu rozobrané viaceré návrhy, ktoré boli zamietnuté, ale najmä návrh, ktorý je použitý v samotnej aplikácii. Popísané budú algoritmy a postupy, ktoré sa použijú pri vytváraní upravených segmentov z prepisu reči.

4.1 Korektúra prepisu reči

Automatický rozpoznávač nepracuje na 100% korektne. Ide o nástroj, ktorý vie rozpoznávať reč na základe slovníka a vlastných znalostí. Preto je potrebné tieto znalosti prehlbovať, naučiť rozpoznávač nové slová, výrazy a pod.

Tento problém sa snaží riešiť práve modul korekcie rozpoznávania reči. Užívatelia budú môcť počas prednesu sledovať prepis reči prednášajúceho. Ak je nesprávny, je možné ho upraviť. Následne bude možné na základe rozpoznaného originálu s kombináciou úprav vytvoriť korektný prepis reči. Správnosť prepisu zaručí hlasovanie. Ak je prepis správny, môže slúžiť ako vstup pre rozpoznávač, ktorý sa týmto trénuje a učí nové výrazy. Toto je jeden z hlavných cieľov, prečo bola korektúra prepisu reči vytvorená.

4.2 Analýza problému

Je treba vybrať a implementovať spôsob korekcie prepisu reči tak, aby bol čo najefektívnejší, najjednoduchší, intuitívny a najmä, aby upravené texty boli správne. Je snaha docieliť, aby užívatelia mali vôľu o vytváranie korektúr, a aby tie boli správne.

Nutnosťou je, aby sa užívateľ dostal ku korektúre čo najrýchlejšie a intuitívnym spôsobom – teda bez zložitých preklikov na sekcie, s tým aby pre úspešný prepis užívateľ vykonal čo najmenej krokov.

Je nutné zvoliť správne metódy pri vytváraní nového segmentu, pretože ide o zložitú formu prevodu reťazca na iný, kde sa nemôže prepísať jednoducho jeden za druhý, ale je treba nájsť medzi nimi podobnosť a následne vytvoriť časovanie upraveného segmentu na základe primárneho segmentu.

Výslednú úpravu nemožno automaticky považovať za správnu, preto je treba navrhnúť metódu akou docieliť, aby zmenené segmenty boli správne. Tento problém sa nedá vyriešiť automaticky, preto bude vytvorený prvok, ktorý bude zaisťovať manuálne ohodnocovanie úprav.

4.2.1 Analýza vytvárania upravených segmentov

Po vykonaní užívateľskej časti úpravu prepisu segmentu budú na vstupe tohoto problému 2 reťazce: nazvime ich *starý* a *nový* pre lepšiu orientáciu.

O starom sú k dispozícii 2 informácie, a to:

- Štartovací čas každého slova + koncový čas posledného slova v reťazci
- Samotný text reťazca

Nový reťazec ale nie je načasovaný a dá sa zistiť teda len jeho textová podoba. Časové razítka pre každé slová budú musieť byť zistené alebo vygenerované.

Ako prvý krok je podstatné nájsť rozdiely medzi starým a novým segmentom. Rovnaké časti by sa mali zarovnať pod seba a časti, ktoré sú odlišné je treba upraviť do potrebnej podoby (viac podkapitola 4.4).

Na schéme 4.1 je možné si všimnúť dvoch rozličných reťazcov, ktorých sémantika je podobná. Takáto problematika sa bude spájať aj s porovnávaním reťazcov pri editácii segmentov. Isté slová budú musieť byť mazané, iné doplňané. Ostatné budú rovnaké, a práve z nich bude možné zistiť časové stopy a prípadne generovať nové stopy pre nové slová.

Reťazec 1:	Ahoj	Peter,	ideš	dnes	do	školy		?
Reťazec 2:	Ahoj	Peter,		dnes	do	školy	prídeš	?

Schéma 4.1: príklad zarovnania 2 podobných reťazcov

Na základe schémy 4.1 je možné demonštrovať vytváranie opravených segmentov. Predpokladajme že *Reťazec 1* je pôvodný reťazec a chceme ho opraviť na základe nového reťazca *Reťazec 2*. Zelenou farbou sú vyznačené tie slová, ktoré sú v oboch reťazcoch rovnaké a červenou rozdielne. Čiže pri generovaní opraveného segmentu sa z *Reťazca 1* odstráni slovo *„ideš“* a z *Reťazca 2* sa vyberie slovo *„prídeš“* a pridá sa do opraveného segmentu.

Ďalšou otázkou je časovanie slov. Opravené segmenty sa vytvoria zo starého reťazca a jeho časovými razítkami skombinovaním s novým reťazcom (viac podkapitola 4.4).

4.3 Návrh riešenia úpravy segmentov

Táto podkapitola je zameraná na klientskú časť, konkrétne užívateľské rozhranie vytvárania prepisov v rozpoznanom texte.

Bolo vymyslených viacero návrhov. Každý z nich bol ohodnotený malým prieskumom medzi potencionálnymi užívateľmi. Návrhy sa snažili čo najlepším spôsobom aplikovať celú problematiku od nájdenia danej časti prepisu, cez samotný prepis až po záverečné vyhodnotenie.

4.3.1 Výber korigovanej časti

Výber konkrétneho segmentu textu, ktorý bude upravovaný musí byť jednoznačný. Je treba zabezpečiť jednoduchý pohyb po celom texte a treba ponúknuť užívateľovi čo najergonómnejšiu možnosť úpravy.

Vlastná sekcia

Ide o myšlienku vytvoriť pre korektúru vlastné rozhranie. K danej sekcii by sa užívateľ dostal preklikom z hlavnej stránky, kde by si potom vybral konkrétnu prednášku alebo zo stránky prednášky, ktorú aktuálne prehliada.

Výhody:

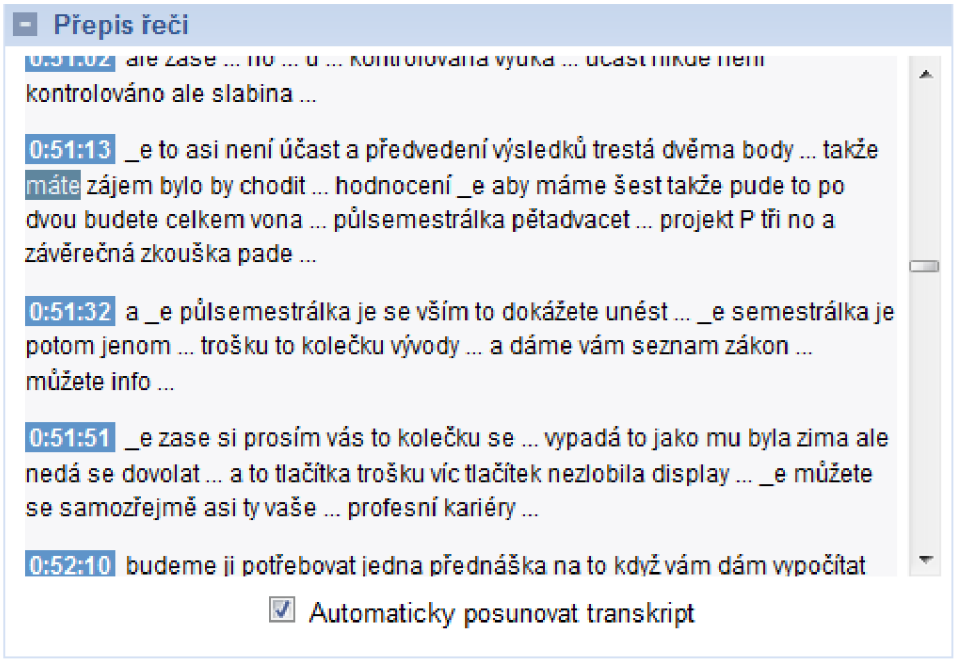
- možnosť vytvorenia viacerých prvkov kvôli väčšiemu priestoru na novej stránke
- možnosť viac naviesť užívateľa k správnej úprave segmentov

Nevýhody:

- zložitejšia orientácia – ďalšia stránka, na ktorú sa treba „prekliknúť“
- rýchlosť práce – pozmeniť text je časovo menej náročné

Korektúra v prepise reči

Ďalšou preberanou možnosťou bolo zakomponovať výber do už vytvoreného prepisu reči (Obr. 4.1), kde užívateľ, ak má spustený automatický posun transkriptu vidí v texte, čo prednášajúci práve rozpráva.



Obr. 4.1: Prepis reči s automatickým posunom

Výhody:

- všetko v jednom – poslucháč vyberie segmet počas pozorovania prednášky
- nie je nutnosť ďalších prvkov na stránke, je pre užívateľa jednoduchšia

Nevýhody:

- intuitívnosť prostredia – pre nových užívateľov môže byť zložitejšie
- nemožnosť vytvorenia ďalších blokov s nápovedami, ktoré by poslucháča usmernili pri výbere daného bloku

Výber segmentu z celého prepisu môže byť realizovaný rôznymi udalosťami (event v Javascript-e):

- **onClick** (kliknutie) – ten je už ale obsadený a pri kliku na segment sa prednáška posúva na danú časť prednášky
- **onDoubleClick** (dvojklik) – tento event má obmedzenú podporu v rozličných prehliadačoch a na jeho simuláciu neboli nájdené dostatočné podklady
- **onRightClick** (klik pravým tlačidlom) – obmedzená funkčnosť v niektorých prehliadačoch, simulácia je náročná, a podklady nedostatočné
- **onMouseDown + onMouseUp** (podržanie kliknutia) – tieto eventy sú podporované vo všetkých bežných prehliadačoch. Medzi stlačením a uvoľnením tlačítka sa nastaví časový limit na oddelenie tohto eventu od jednoduchého kliknutia. Teda pôjde o podržanie ľavého tlačidla.

Nakoniec bol zvolený spôsob výberu v prepise reči. Zvolený segment sa vyberie pomocou JavaScript udalostí onMouseDown + onMouseUp. Táto metóda bola po konzultáciách s potencionálnymi užívateľmi vybraná a v aplikácii použitá.

4.3.2 Úprava segmentu

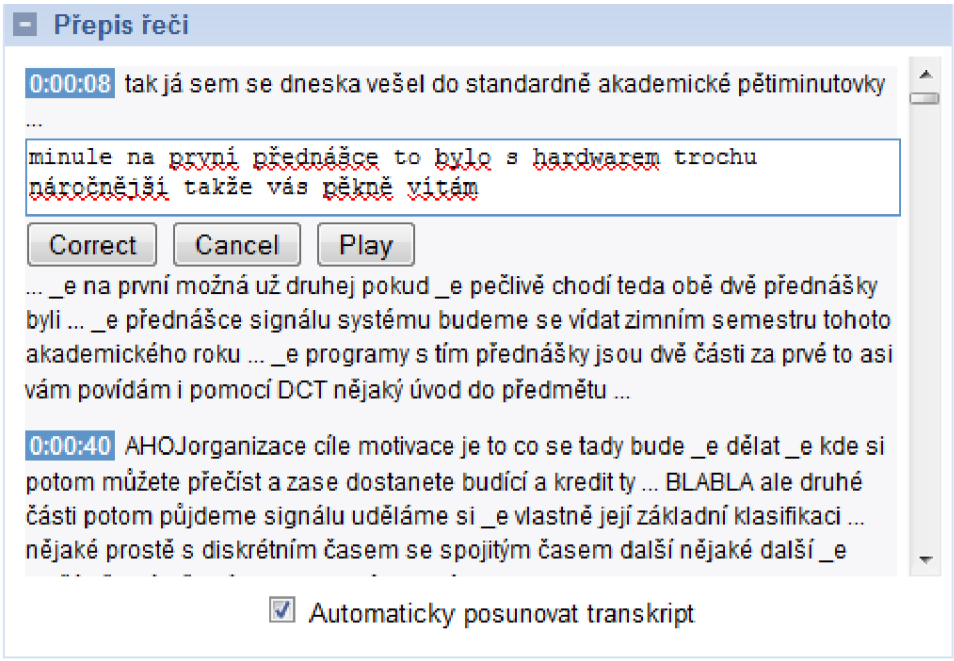
Predpokladajme, že máme vybraný konkrétny segment. Teraz je potreba ponúknuť poslucháčovi zrozumiteľnú a jednoduchú možnosť úpravy danej časti. Výsledný text bude treba vpísať do textového poľa a potvrdiť. Týmto sa ďalej odošle na úpravu, ale to už je záležitosť serverovej časti procesu.

Treba uvažovať nad prvkami, ktorými sa docieli zrozumiteľnej komunikácie medzi užívateľom a prvkami rozhrania. Boli zvolené tri aktívne prvky pre tento krok, ktoré budú reprezentované tlačidlami:

- Úprava
- Návrat späť (bez zmeny)
- Znovu vypočuť segment od prednášajúceho

Celá forma editačného políčka musí byť minimalistická, aby nepôsobilo rušivo vzhľadom na stránku prednášky. Políčko bude vložené do prepisu reči (Obr. 4.2), a tým nahradí pôvodný segment v tomto prvku. V ňom sa vyskytne editačný prvok *textarea* z HTML, ktorého obsahom bude pôvodný text (Obr 4.2).

Tlačidlom Corrrect (Úprava) sa bude realizovať prepis segmentu. Cancel (Späť) vráti prepis reči do pôvodného stavu. Play posunie video na časť začiatku segmentu, aby si užívateľ pri prepise mohol spustiť časť znova.



Obr 4.2: Prepis reči so zobrazeným editačným poľom

4.4 Návrh ohodnocovania upravených segmentov

Novovytvorený segment nie je možné považovať za korektný. Nesprávnosť môže vzniknúť z rôznych dôvodov. Bežné bude prepočutie sa opravujúceho užívateľa, prípadne môžu nastať chyby ako vytvorenie prepisu k nesprávnemu segmentu a pod. Nemôžme ale vylúčiť ani neprajníkov, ktorí by úmyselne sa snažili poškodiť alebo znehodnotiť prednášku vpisovaním nezmyselných, urážlivých alebo vulgárnych slov.

Potvrdzovanie správcom

Jednou z možností je nechať potvrdzovať upravené segmenty nejakým správcom. Správca by mohol byť jednotlivec alebo aj skupina aktérov, ktorí by boli vybraní ako dôveryhodní. Tento spôsob je natoľko spoľahlivý ako je spoľahlivý sám správca, čiže správnym výberom aktérov by bola korektnosť úprav zaručená.

Nevýhodou takéhoto systému je pravdepodobná nemožnosť nájdenia správnych užívateľov na túto pozíciu. Správca by sa musel venovať svojej práci takmer neustále, aby bola zachovaná aktuálnosť údajov.

Hodnotenie užívateľmi

Ďalšou možnosťou, ktorá je nakoniec vybraná ako realizovaná je hodnotenie každého segmentu užívateľmi. Všetci poslucháči a aj administrátori môžu hlasovať za každý vybraný segment, a tým mu udeľovať skóre. Toto skóre má buď kladné alebo záporné hodnoty. Každý opravujúci má zároveň priradenú konštantnú hodnotu zvanú „karma“, ktorá sa mení podľa jeho správania sa v systéme.

Týmto sa práca spája s ďalšou prácou [13], kde je užívateľom karma upravovaná, a to pri komentovaní a hodnotení komentárov prednášok.

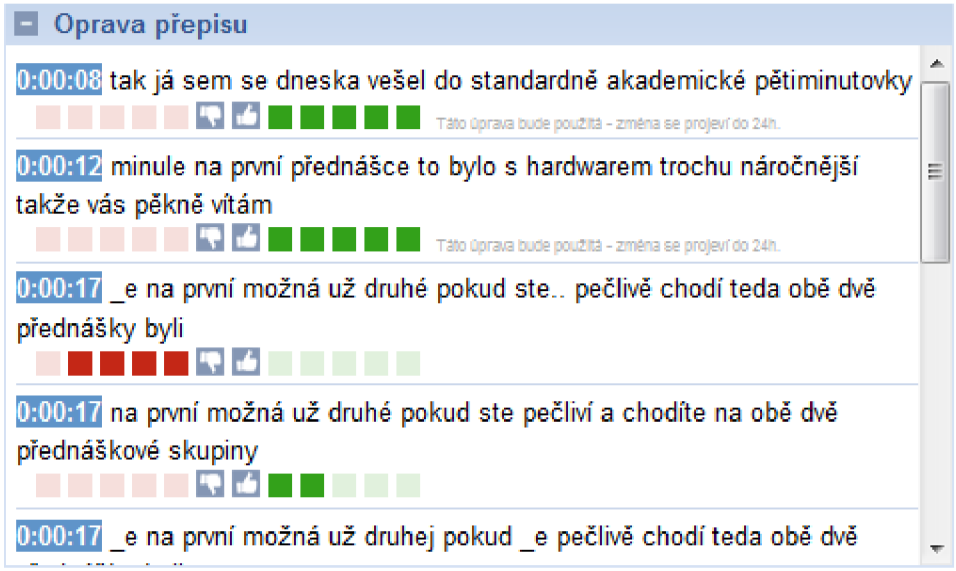
Pokiaľ dosiahne skóre natoľko vysokých hodnôt, aby bol použitý, tak na základe toho bude segment vytvorený v novom prepise reči. Naopak, ak dosiahne určitú hodnotu mínusového skóre, zo systému bude zmazaný.

Na každý segment z prepisu reči môže byť vytvorená jeho úprava. Akonáhle dosiahne určitého skóre, tak sa segment zmení a v ponuke hodnotenia zostávajú nepoužité návrhy.

4.4.1 Hodnotenie opravy prepisov

Pre tento účel bol do stránky prednášky vložený prvok na hodnotenie opráv prepisov. Pri návrhu sa dbalo na minimalizáciu a jednoduchosť, aby prvok zbytočne nepôsobil rušivo. Hodnotenie by malo byť realizované pomocou AJAX-u na strane serveru, aby poslucháč nemusel zbytočne zastavovať a znovu-načítavať prednášku.

Každý užívateľ môže jeden segment hodnotiť najviac jeden krát. Prideľujú sa buď kladné alebo záporné body kliknutím na ikonku „palec hore“ resp. „palec dole“.



Obr. 4.3: Panel s hodnotením prepisov reči.

4.5 Návrh vytvorenia nového segmentu

Táto podkapitola, naopak od predchádzajúcej, sa zaoberá vytváraním prepisu rozpoznaného textu na strane servera. Čiže nepôjde o grafické prvky, ale nastolí problematiku postupu pri transformácii jedného reťazca na druhý.

Podkapitola bude sprevádzaná príkladom, ktorý bude demonštrovať vytvorenie nového segmentu na základe 2 reťazcov – pôvodného a náhradného, ktorý bol vytvorený poslucháčom. Tieto časti sú z reálnej aplikácie. Reťazec A (pôvodný), ktorý bol vygenerovaný rozpoznávačom a reťazec

B, ktorý bol zachytený z neautomatizovaného počutia. Reťazce boli príliš dlhé na demonštrovanie, tak boli skrátené.

Celý segment:	„tak já sem se tak aby šel do standardné akademické pětiminutovky“
Reťazec A:	„tak já sem se tak aby šel do“
Reťazec B:	„tak já sem se dneska vešel do“

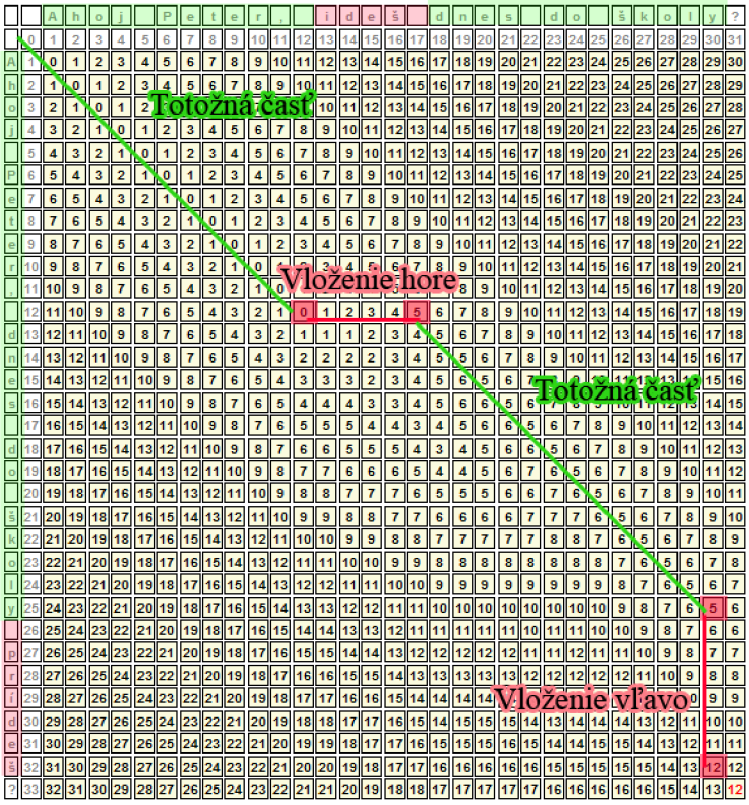
4.5.1 Nájdenie rozdielu reťazcov – levenshtein algoritmus

Na hľadanie rozdielu medzi dvomi reťazcami sa využije algoritmus Levenshtein distance (podkapitola 2.4.1). Výstupom je počet zmien potrebných na transformáciu jedného reťazca na druhý, tá ale v práci nie tak dôležitá ako postup ako sa k tejto hodnote dostaneme.

Vytvorí sa matica, ktorá reprezentuje hodnoty výpočtu. Na maticu sa pozeráme smerom od ľavého horného rohu po pravý dolný. Zisťujeme cestu medzi týmito dvomi hraničnými bodmi (Obr. 4.4). Môžu nastať 4 situácie:

- medzi dvomi susednými hodnotami nie je zmena
  - v diagonále - dva porovnávané znaky sú rovnaké
- medzi dvomi susednými hodnotami je zmena
  - v diagonále – dva porovnávané znaky sú rozdielne
  - zľava doprava – vloženie znaku do horného reťazca
  - zhora dole – vloženie znaku do ľavého reťazca

Výstupom tejto časti je matica, ktorá je naplnená hodnotami použitými pri výpočte algoritmu Levenshtein distance.



Obr. 4.4: Ukážka postupu výpočtu Levenshtein algoritmu s načrtnutou výslednou cestou

Príklad - levenshtein

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
13	12	11	10	9	8	7	6	5	4	3	2	1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
14	13	12	11	10	9	8	7	6	5	4	3	2	1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15	14	13	12	11	10	9	8	7	6	5	4	3	2	2	2	3	4	5	5	6	7	8	9	10	11	12	13	14
16	15	14	13	12	11	10	9	8	7	6	5	4	3	3	3	3	4	4	5	6	7	8	9	10	11	12	13	14
17	16	15	14	13	12	11	10	9	8	7	6	5	4	4	4	4	4	5	5	5	6	7	8	9	10	11	12	13
18	17	16	15	14	13	12	11	10	9	8	7	6	5	5	5	5	5	5	5	6	6	7	8	9	10	11	12	13
19	18	17	16	15	14	13	12	11	10	9	8	7	6	6	6	6	6	6	6	6	7	7	8	9	10	11	12	13
20	19	18	17	16	15	14	13	12	11	10	9	8	7	7	7	7	7	7	7	7	7	8	8	9	10	11	12	13
21	20	19	18	17	16	15	14	13	12	11	10	9	8	8	8	8	8	8	8	7	8	8	9	9	10	10	11	12
22	21	20	19	18	17	16	15	14	13	12	11	10	9	9	9	9	8	9	9	8	8	9	8	9	10	11	11	12
23	22	21	20	19	18	17	16	15	14	13	12	11	10	10	10	9	9	9	10	9	9	8	9	8	9	10	11	12
24	23	22	21	20	19	18	17	16	15	14	13	12	11	11	11	10	10	10	10	10	10	9	9	9	8	9	10	11
25	24	23	22	21	20	19	18	17	16	15	14	13	12	12	12	11	11	11	11	10	11	10	10	10	9	8	9	10
26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	13	12	12	12	12	11	11	11	11	11	10	9	8	9
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	13	13	13	13	13	12	12	12	12	12	11	10	9	8

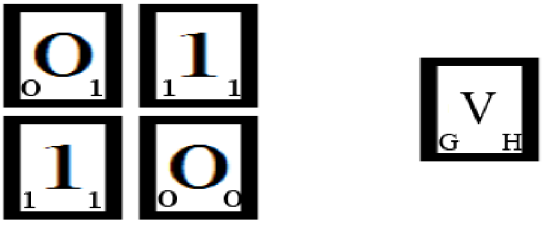
Obr 4.5: Výstup z levenshtein algoritmu

4.5.2 Hľadanie najefektívnejšej trasy – A*

Vyhľadanie najefektívnejšej trasy z jedného bodu do druhého výborne rieši algoritmus A*, preto bude využitý aj v problematike nájdenia trasy medzi ľavým horným a pravým spodným rohom výstupnej matice z algoritmu Levenshtein distance.

A* algoritmus bude musieť byť mierne upravený, pretože v skutočnosti sa nehľadá najkratšia trasa na čo je primárne určený, ale najefektívnejšia. Úprava spočíva v zmene použitej heureisitky. Vzdialenosť medzi aktuálnym prvkom a počiatkom sa neprepočítava metricky, ale ako rozdiel hodnôt týchto dvoch prvkov (Obr. 4.6).

Hodnoty v prvkoch sú 1 a 0. *G* reprezentuje už absolvovanú vzdialenosť od počiatočného bodu (v počiatočnom = 0), *H* je prepočítaná vzdialenosť k cieľu a V je hodnota prvku matice. G je vypočítané na základe predchádzajúceho prvku a rozdielu hodnôt prvkov a H sa vypočíta ako rozdiel *x* a *y* indexu prvku daného a *x* a *y* indexu cieľového prvku.



Obr. 4.6: Heureistika v algoritme A*

Výstupom bude trasa daná prvkami, ktoré obsahujú index matice a hodnotou pre daný prvok. Týmto sa teda našla najkratšia trasa a vieme teda určiť akým spôsobom je jeden reťazec transformovaný na druhý.

Príklad - A*

Obr 4.7: Výstup z A* algoritmu

4.5.3 Zmena trasy na transformačný vektor

V tomto kroku je k dispozícii trasa, ktorú je nutné zmeniť na transformačný vektor. Ten sa skladá zo znakov:

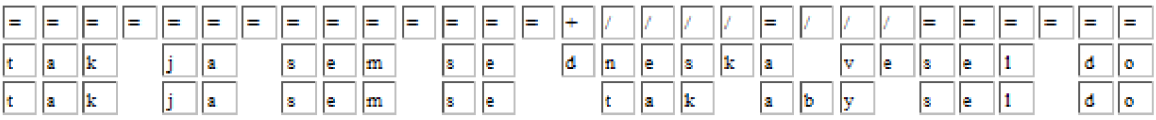
- = - pre rovnaký znak
- / - pre zámenu znakov
- - - pre vypustenie znaku
- + - pre vloženie znaku

Aby bolo toto docielené, treba prechádzať trasu z algoritmu A* postupne od konca k začiatku. Podľa spôsobu vytvárania matice levenshtein algoritmu sa spätne zisťuje, aký krok bol použitý. Podobne môžu nastať 4 situácie:

- medzi dvomi susednými hodnotami krokov z trasy nie je zmena
  - v diagonále - dva porovnávané znaky sú rovnaké – do vektoru sa vloží „=“
- medzi dvomi susednými hodnotami krokov z trasy je zmena
  - v diagonále – dva porovnávané znaky sú rozdielne – do vektoru sa vloží „/“
  - zľava doprava – vloženie znaku do horného reťazca – do vektoru sa vloží „+“
  - zhora dole – vloženie znaku do ľavého reťazca – do vektoru sa vloží „-“

Vstupom je teda matica z A* a výstup je transformačný vektor, ktorý bude následne upravený. V obrázku 4.8 u príkladu je transformačný vektor na prvom riadku, ale pre ilustráciu je doplnený oboma reťazcami, ktoré boli určené na vytvorenie opraveného segmentu. Tieto reťazce boli pre lepšie znázornenie upravené podľa vektora.

Príklad – transformačný vektor



Obr 4.8: Výstup z vytvárania transformačného vektoru

4.5.4 Vytváranie výstupného reťazca

Postupným prechádzaním reťazcov pôvodného (*bad*) a nového (*good*) s kombináciou s priechodom transformačného vektora sa postupne vytvára výsledný reťazec (*result*).

Z tranformačného vektoru sledujeme postupne príznaky:

- = - do *result* pripíšeme znak z *good*
- / - do *result* pripíšeme znak z *good*
- + - do *result* pripíšeme znak z *good*
- - - do *result* nepripisujeme nič

Týmto získame presný tvar výsledného reťazca, tento krok sa bude realizovať spolu s pridaním časovania, keďže pôjde o podobný prístup.

Pri tvorbe ale zaznamenávame, či sa v *bad* vyskytuje medzera – oddelovač slov. Ak áno, tú zachytíme a u výstupu bude segment rozdelený na podvety. Detailnejší postup je popísaný v implementácií (kapitola 5.3.4).

Príklad – Vytváranie výstupného reťazca

1. podveta	2. podveta	3. podveta	4. podveta	5. podveta	6. podveta
tak	ja	sem	se	dneska vesel	do

Tabuľka 4.1: výstup z vytvárania výstupného reťazca

4.5.5 Pridanie časovania

Ako bolo spomínané, časovanie sa vytvára počas predošlého kroku, ale pre lepšiu demonštráciu boli tieto časti logicky oddelené.

V tejto fáze, prepisu segmentu sú k dispozícii 2 reťazce a trasnformačný vektor. Teraz sa segment finálne vytvorí, s tým že bude správne načasované každé slovo.

Najprv sa z pôvodného reťazca odstránia nepotrebné záznamy, ktoré majú transformačný znak „-“, pretože nebudú potrebné, keďže sa pravdepodobne detekovali rozpoznávačom nesprávne.

Ďalej sa vytvorí nový segment a to takým spôsobom, že:

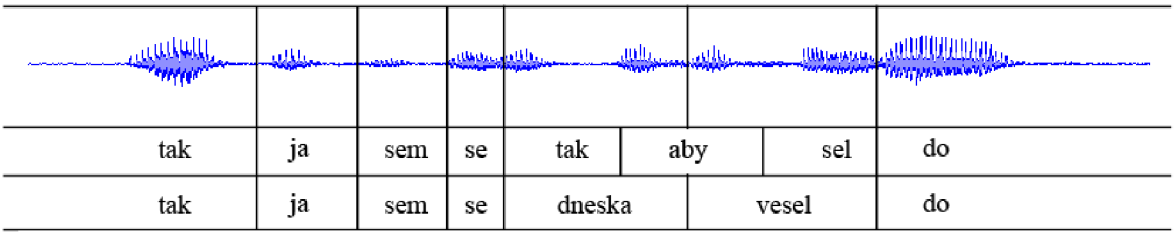
Ak sa v transformačnom vektore nachádzajú znaky „=“ tak sa do segmentu zapíše aktuálny podreťazec s originálnym časovaním slov s tým, že časovanie sa posúva na základe výskytu medzier, pretože slová sú delené vždy medzerami. Ak sa v tranformačnom vektore nachádzajú znaky „+“, do

segmentu sa zapíše nový podreťazec s tým, že časovanie pre každé slovo sa aritmeticky dopočíta podľa počtu písmen v novej časti segmentu. Dopočítanie je realizované na základe predošlých resp. následujúcich častí segmentov, ktoré už načasované sú.

Príklad – Pridanie časovania

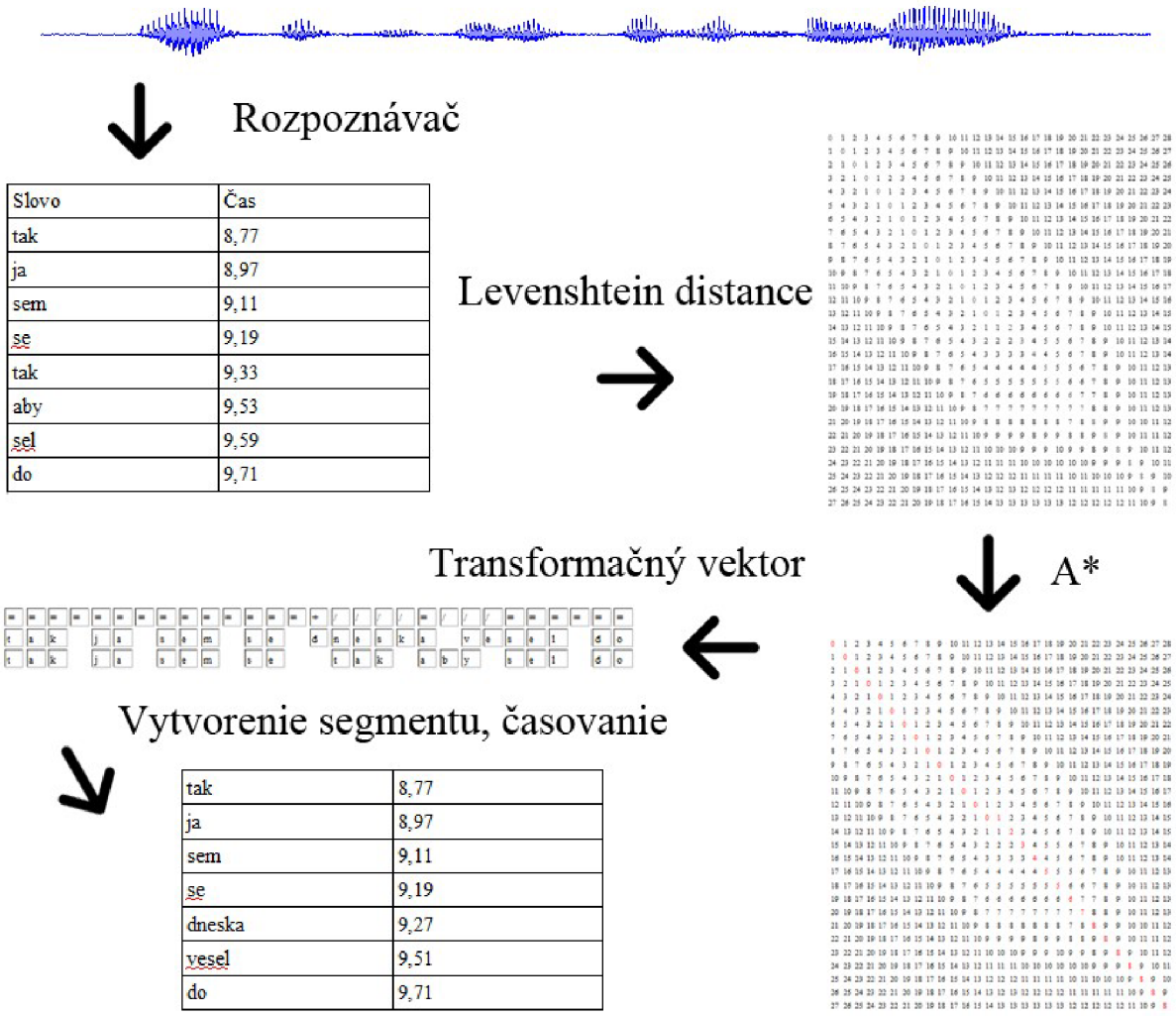
Pôvodné časovanie		**Nové časovanie**	
Slovo	Čas	Slovo	Čas
tak	8,77	tak	8,77
ja	8,97	ja	8,97
sem	9,11	sem	9,11
se	9,19	se	9,19
tak	9,33	dneska	9,27
aby	9,53	vesel	9,51
sel	9,59	do	9,71
do	9,71		

Tabuľka 4.2: výstup z pridávania časovania



Obr 4.9: Postup rozoznania reči + následná oprava

Na analýzu zvukového signálu bol použitý softvér Audacity [15].

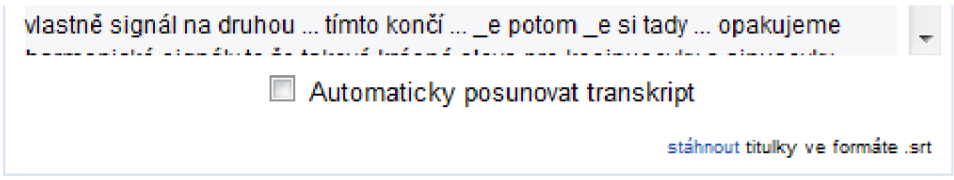


Obr 4.10: Postup vytvorenia korektného segmentu zo zvukového formátu

4.6 Návrh vytvorenia titulkov

Súčasťou aplikácie je možnosť stiahnutia titulkov. Titulky bude možné stiahnuť vo formáte SubRip, teda pôjde o súbor s príponou .srt. Ide o najbežnejšie používaný formát, ktorý nemá problém prehrať ktorýkoľvek z dnešných prehliadačov videa.

Možnosť stiahnuť titulky bude umiestnená v paneli „prepis reči“ pomocou odkazu (Obr. 4.9).



Obr 4.11: Prepis reči s možnosťou stiahnutia titulkov

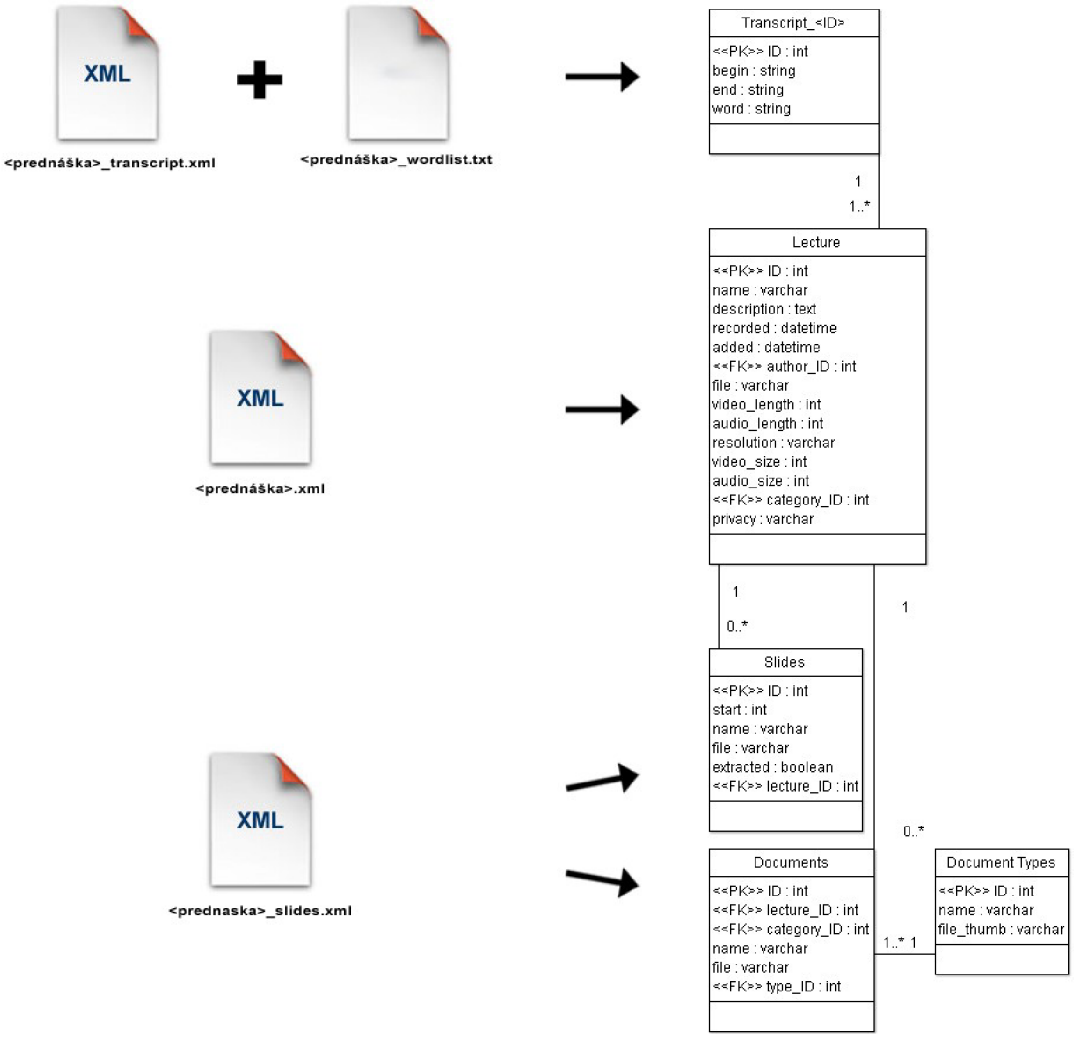
5 Implementácia

Táto kapitola sa zaoberá implementáciou prevodu prehliadača prednášok na MySQL databázu a vytvárania opráv transkriptov. Popíšu sa postupy, ktoré napĺňajú databázu dátami zo súborov a prevodu súčasnej implementácie na implementáciu využívajúcu databázový server. Ďalej sa spomenú spôsoby, ktoré boli použité pri vývoji nahradzovania segmentov, ako sú dané algoritmy, rozdelenie problémov na podproblémy a pod.

Projekt bol vyvíjaný a testovaný s použitím softvérového balíku WAMP 2.0i.

5.1 Prevod obsahu súborov do MySQL databáze

Prevod bol realizovaný využitím parsera XML Parser [8]. Vytvorila sa množina funkcií v PHP, ktoré parsujú súbory a dáta ukladajú do databáze.



Obr. 5.1: Migrácia dát zo súborov do databázy

5.1.1 Browser parser

Browser parser je skupina funkcií písaná v PHP, umožňujúca vytvárať entity z prednáškových súborov v databáze. Môžme ju logicky rozdeliť na 2 časti:

Správa prednášky

V tejto časti sú implementované funkcie na vytváranie prednášok zo súborov *<prednáška>.xml* a *<prednáška>_slides.xml*.

```
$xmlL = "prednaska.xml";
$xmlS =  "slides.xml";

if(($lID = parse_XML1($xmlL, 1))!=0){ // uspešné vytvorenie prednášky

  if(($id = parse_slides($xmlS,$lID))!=1)
    echo "nepodarilo sa vytvoriť slajdy";

}
else echo "nepodarilo sa vytvoriť prednášku";
```

Kód 5.1: Vytvorenie prednášky so slajdami

Ukážka (Kód 5.1) znázorňuje vytvorenie prednášky zo súboru *prednaska.xml*, a slajdov zo súboru *slides.xml*. Obe funkcie vrátia hodnotu `0` pri neúspešnom vytvorení. Inak funkcia *parse_XML1()* vracia *ID* práve vytvorenej prednášky, ktoré je použité vo funkcii *parse_slides()*.

Správa transkriptu

Ide o funkcie, ktoré vytvárajú databázové tabuľky s transkriptami a naopak vytvárajú súbory s transktiptami z databázy.

Na vytvorenie tabuľky je nutné mať k dispozícii dva súbory:

- <prednáška>_subtitles.tt
- <prednáška>_wordlist.txt

Kombináciou týchto dvoch súborov sa vytvorí tabuľka v MySQL databáze. Jeden súbor nešlo použiť, pretože je potrebné ukladať do databázy slová s ich časom začiatku/konca. Zo súboru *wordlist* sa vyberajú slová a počiatočné časy a z *subtitles* sa vyberajú koncové časy.

Teda na vytvorenie slúži následujúci kód:

```
$lecture_file = "prednaska.xml";
$slides_file =  "slides.xml";

if(($result = parse_subtitles($wordlist,$subtitles))==0)
 echo "nepodarilo sa vytvoriť transkript";
```

Kód 5.2: Vytvorenie transkriptu

Vytváranie súborov s transkriptami z databázy je proces zložený z funkcií. Každá dielčia funkcia vytvára samostatný súbor, je teda možné súbory vytvárať aj jednotlivo. Pre vytvorenie všetkých súborov je možné použiť funkciu *create_all_files()* (Kód 5.3) s jediným parametrom *lecture*, ktorý reprezentuje identifikáciu prednášky.

```
if(($result = create_all_files($lecture))==0)
 echo "nepodarilo sa vytvoriť súbory";
```

Kód 5.3: Vytvorenie súborov

5.2 Prevod prehliadača prednášok na MySQL

Po preštudovaní zdrojového kódu prehliadača prednášok od Ing. Jozefa Žižku bola analyzovaná architektúra projektu. Projekt je napísaný v jazyku PHP na generovanie HTML súborov s použitím JavaScript-u a CSS. Boli využité objektovo-orientované vlastnosti PHP.

Sú tu použité 4 hlavné triedy na generovanie obsahu stránok.

- Page
- CategoryPage
- LecturePage
- AudioSearchPage

A 3 na získavanie dát

- Lecture
- LectureList
- CategoryList

Na prevod bolo zasahované takmer do každého súboru aplikácie. Spočiatku bola vytvorená trieda MySQL, ktorá zabezpečuje pripojenie na MySQL databázu. Tá obsahuje pripojenie, získanie stavu pripojenia a generovanie chybových hlášok pre pripojenie

V rámci tried na generovanie obsahu stránok bolo potrebné zmeniť najmä prvky tak, aby boli správne naviazané na triedy na získavanie dát. Samozrejme bolo pridaných niekoľko metód kvôli zmene štruktúry tried na získavanie dát.

Triedy na získavanie dát boli upravené tak, aby dáta neboli načítavané z XML súborov, ale z databázy MySQL. Tu bolo vytvorených viacero metód na previazanie kategórií s prednáškami a naopak. Na následujúcom príklade (Kód 5.4a a 5.4b) si ukážeme jeden z jednoduchších prevodov z XML čítania dát na MySQL. Ide o získanie kategórie na základe zadaného ID prednášky z triedy LectureList.

```
public static function getLectureCategory($id)
{
  for ($u = 0; $u < count(self::$lectureList); $u++) {
    if ( strcmp($id, self::$lectureList->lecture[$u]->id) == 0 )
      return (string)self::$lectureList->lecture[$u]->categ_id;
  }
  return null;
}
```

Kód 5.4a: Získanie id kategórie na základe id prednášky s použitím XML

U tohto príkladu je možné si všimnúť aj rozdielnu zložitosť. Pri použití XML sa musí prejsť celým XML súborom s prednáškami (*lecture_list.xml*) získať každé *id* prednášky. Ak sa našla zhoda, vrátilo sa *id* kategórie k nej prislúchajúce.

Naopak pri použití MySQL sa vykoná jeden vyhľadávací dotaz a výsledkom je *id* kategórie.

```
public static function getLectureCategory($id)
{
 if($id){
    $result = mysql_query("SELECT category_ID FROM `lecture` WHERE
                           id='".$id."' LIMIT 1");
    $row = mysql_fetch_array($result);
      return $row[0];
 }
 return null;
}
```

Kód 5.4b: Získanie id kategórie na základe id prednášky s použitím MySQL

Použitie MySQL sa vidí ako lepšie aj pri zložitejšom vyhľadávaní. Príkladom je vyhľadávanie na základe dátumu. Predstavme si, že chceme nájsť prednášky pridané po určitom čase *X*.

U XML by bolo treba načítať opäť všetky prednášky, následne ich zoradiť podla času. Po zoradení by sa našla hraničná prednáška vzhľadom na čas *X*. Nakoniec všetky prednášky, ktoré mali čas väčší ako *X*, by boli výsledkom.

U MySQL je situácia jednoduchšia. Opäť celé vyhľadávanie riadi jeden dotaz. Dotaz je v tvare „vyber všetky prednášky z tabuľky prednášky, kde čas je väčší ako *X*“. Ide teda o prirodzenejšie a výpočtovo jednoduchšie vyhľadávanie.

Ako nevýhoda použitia MySQL na prehliadač bol zaznamenaný zložitejší prístup k dátam. V XML sa dáta načítali pomocou Simple XML do objektu a práca s nimi bola intuitívnejšia a jednoduchšia.

Pri prevode z XML implementácie do MySQL nedošlo k zmene vlastností prednášok, či zmene komponentov. Bolo pridaných pár vlastností, ako je zobrazenie popisu prednášok alebo kategórií, zobrazenie dokumentov z dátového skladu spolu s ikonkou dokumentu a podobne.

5.3 Úprava transkriptov

Úprava transkriptov je vytvorená podľa návrhu, ktorý je detailne popísaný v kapitole 4. Boli použité najmä skriptovací jazyk PHP, výstup je prezentovaný s využitím jazyka JavaScript a značkovacieho jazyka HTML s použitím kaskádových štýlov CSS.

5.3.1 Grafické užívateľské rozhranie

Stránka webového prehliadača je ladená v jednoduchom štýle, preto bola snaha tento štýl zachovať a nevytvárať zbytočne zložité prvky rozhrania. Zostala farebnosť, ktorá má modrastú farbu.

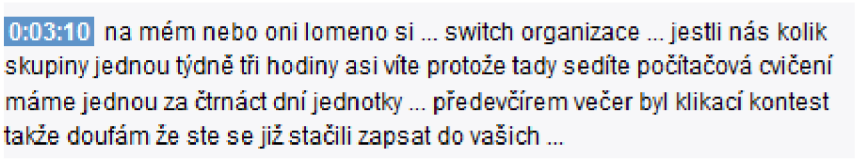
Celá práca v aplikácii je orientovaná najmä na prácu s myšou, čiže tento smer nebol narušený inými prvkami. Okrem prepisovania textu, ktoré je realizované na klávesnici aplikačná časť tejto práce naďalej ovláda myšou.

Obrázky, ktoré boli použité sú vo formátoch png. Tento formát sa často používa na webových aplikáciach, pretože pri nich klient zo serverom komunikuje rýchlejšie, kvôli ich malej veľkosti súborov.

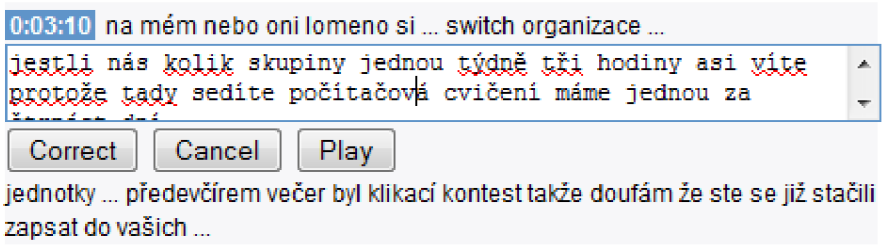
5.3.2 Výber a prepis segmentu

Pri výbere a prepisovaní segmentu sa väčšina krokov deje na klientovi, takže z implementačného hľadiska bol využívaný najmä JavaScript, HTML, CSS.

Výber segmentu sa vytvoril pomocou kliknutia a podržania ľavého tlačidla myši, teda pomocou dvoch funkcií *onMouseDown* a *onMouseUp* volaných na elementoch z panelu prepisu reči..
Problémom bolo, že pri udalosti *onClick* bola pridelená elementom už funkcionalita posúvania sa v rámci prednášky. Bolo treba tieto udalosti oddeliť. Potrebná doba podržania ľavého tlačidla bola nastavená na 1 sekundu, čo sa javí ako postačujúce, a zároveň nie príliš dlhé. Po detekcii sa zavolá funkcia editSegment(segment, čas). Tá v paneli prepisovania reči transformuje element obsahujúci segment na HTML formulár.



Obr. 5.2a: Blok segmentov v prepise reči

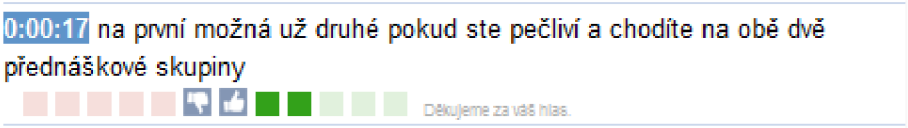


Obr. 5.2b: Blok segmentov v prepise reči s formulárom

Ak sa potvrdí editácia (tlačítko „Correct“), je volaný AJAX – ový súbor *updateTranscript.php*, ktorému sú odoslané hodnoty formulára metódou GET. Ten už daný segment pridelí do systému. Vložením do databázy je vytvorený element hodnotenia pridaný do panelu hodnotení, aby poslucháč videl prejavenú.

5.3.3 Hodnotenie segmentu

Zobrazenie panelu hodnotenia transkripcií vyvolá nová metóda v triede LecturePage. Z databázy sa vyberú segmenty, ktoré majú skóre dané minimom a maximom nachádzajúceho sa v nastaveniach prehliadača prednášok a vložia sa ako jednotlivé elementy do panelu hodnotenia (Obr 5.3).



Obr. 5.3: Element hodnotenia s jedným segmentom

Element sa skladá z 2 častí. Čas a text segmentu, na ktorý keď užívateľ klikne, tak video je posunuté na oblasť, kde daný text prednášajúci rozpráva. Druhá časť predstavuje hodnotenie segmentu. Nachádza sa tu „progress bar“, ktorý určuje mieru správnosti, či nesprávnosti úpravy, získaný zo skóre. Kliknutím na „palec hore“/“palec dole“ je volaný AJAX-ový súbor *updateScore.php*, ktorý zaručí pridanie/odobratie bodu v skóre a aktualizáciu elementu daného segmentom. Bod je popritom násobený konštantou zvaná karma, ktorou každý užívateľ disponuje. Momentálne je nastavený na hodnotu `1`, ale doimplementovanie je ponechané na neskoršie pokračovanie a rozšírenia práce, keďže iná práca, ktorá výpočty karmy vytvára, ešte nie je dokončená.

5.3.4 Implementácia úpravy segmentov

Implementácia je rozdelená na 5 časti podľa návrhu v kapitole 4.5 s tým, že niektoré kroky sú zdvojené: A* s vytvorením transformačného vektora a generovanie výsledného reťazca so správnym načasovaním. Implementácia bola pridaná do modulu Browser parser.

Levenshtein algoritmus

Matica sa vytvára v podobe dvojrozmerného poľa podľa daných pravidiel. Najskôr sa naplnia hodnoty horného riadku a ľavého stĺpca hodnotami *1* .. *n*, kde *n* je dĺžka reťazca. Podľa týchto hodnôt a porovnaní znakov sa doplnia ďalšie prvky matice.

A* a vytvorenie transformačného vektora

Pre tento algoritmus boli vytvorené nové triedy zabezpečujúce správne nájdenie najkratšej trasy. Implementácia bola realizovaná už dávnejšie, ešte ako súčasť iného projektu. Detailnejší popis je možné nájsť v referenčnej literatúre [12].

Výsledkom je transformačný vektor, ktorý je implementovaný v podobe jednorozmerného poľa znakov.

Vytvorenie výsledného reťazca so správnym načasovaním

Majme na vstupe 2 reťazce, ktoré nazveme *bad* (pôvodný), kde máme k dispozícii aj časy kedy bolo slovo povedané a *good* (nový), transformačný vektor a vytvoríme prázdny nový reťazec *result* reprezentovaný asocitatívnym poľom, ktorého index je čas. Pre bad a good si vytvoríme index, ktorý bude použitý pri priechode počas celého algoritmu.

Zistí sa čas prvého slova z *bad*, potom sa prechádza transformačný vektor znak po znaku:

- ak sa v ňom vyskytuje „=“ alebo „/“, tak sa do *result* vloží aktuálny znak z *good* a zvýšia sa vnútorné indexy oboch vstupných reťazcov

- ak sa vyskytuje „+“, vloží sa znak z *good* do *result* a zvýši sa jeho index, index *bad* ostáva nezmenený
- ak sa vyskytuje „-“, zvýši sa index *bad* a tým sa simuluje „vymazanie“ zlého znaku

Týmto dostaneme *result* ako výsledný správny reťazec časovanie prebieha popri spomenutom priechode nasledovné:

- ak je „=“ a zároveň na aktuálnej pozícii (index) v *good* medzera, získame nasledujúci čas z reťazca *bad*, a ďalšie znaky budeme vkladať do *result* s indexom tohoto času (asociatívne pole).
- ak je „/“, skontroluje sa, či sa na aktuálnej pozícii v *bad* nenachádza medzera, ak áno, posunie sa index *bad* (ale nezapisuje sa hodnota kľúču, pretože sa tam môžu vyskytovať viaceré medzery).
- ak je „+“, tak sa nedeje s časovaním nič.
- ak je „-“, tak sa kontroluje *bad* reťazec na výskyt medzery, ak sa tam nachádza, posunie sa index *bad*.

5.3.5 Vytvorenie nových segmentov

Pre ušetrenie výpočtových zdrojov sa tranformácia starého segmentu na nový implementovala ako funkcia, ktorá bude volaná pomocou softvérového démonu cron. Ten v určitých časových okamihoch (najlepšie v noci) spustí skript na vytvorenie nových segmentov pre všetky prednášky v databáze pre ktoré, bude nájdený prepis s adekvátnym skóre.

5.3.6 Vytvorenie titulkov

Titulky sú vytvárané podobne ako pri vytváraní textových súborov popísaných v podkapitole 5.1.1. Generované sú na základe syntaxe danej v špecifikácii formátu.

6 Testovanie a výsledky

Záverečná kapitola sa venuje testovaniu aplikačnej časti vytvorenej v rámci tejto diplomovej práce. Boli použité jednoduché, ale aj náročné-záťažové testy na generujúce väčšieho počtu prvkov. Ďalším testom bol prieskum verejnej mienky, kde je hodnotená aplikácia medzi potencionálnymi užívateľmi.

6.1 Testy vytvárania nových prednášok

Jednoduché testy

Vytvorenie jednej prednášky spolu so slajdami.
Script executed in 0.0085 seconds.
Vytvorenie jednej tabuľky v databáze zo súborov s transkriptom.
Script executed in 3.3675 seconds.

Stredne náročné testy

Vytvorenie 100 prednášok spolu so slajdami.
Script executed in 0.5628 seconds.
Vytvorenie piatich tabuliek s transkriptami.
Script executed in 23.9983 seconds.

Náročné testy

Vytvorenie 1000 prednášok spolu so slajdami.
Script executed in 5.6921 seconds.
Vytvorenie 10 tabuliek s transkriptami.
Script executed in 50.6806 seconds.

Testy vytvárania tabuliek s prednáškami a slajdami ukázali, že ide o pomerne rýchly proces, keď pri vytváraní 100 prednášok trval výpočet cca 0.5 sekundy.

Naopak, pri vytváraní transkriptov je doba výpočtu väčšia, je to z dôvodu veľkosti zdrojových dát. Do databázy je pre jednu priemerne dlhú prednášku vkladaných asi 15000 riadkov slov.

Testy prebiehali na HP Pavilion dv6 Notebook PC so špecifikáciou:
CPU: Intel(R) Core(TM)2 Duo CPU T6600 @ 2.20GHz 2.20GHz
RAM: 4,00GB
OS: Windows 7 Home Premium (64-bitový operačný systém)
Web Server: WAMP

6.2 Testy vytvárania nových transkriptov

Išlo o testy, kde sa vytvorili súbory potrebné pre chod prehliadača z hľadiska prepisov reči. Teda ide o všetky textové súbory. Pridaný je novovytvorený súbor *<prednáška>.srt*, ktorý reprezentuje titulky k prednáške vo formáte SubRip.

Rozdielnosť testov je daná počtom úspešne upravených segmentov. Budeme si všímať najmä zmeny medzi časmi výpočtu pre rozličné počty úprav.

Vytvorenie súborov bez použitia úprav

Segmentov celkovo: 2551
Z toho upravených: 0
Celkový čas: 1.2943 s

Vytvorenie súborov s jedným upraveným segmentom

Segmentov celkovo: 2551
Z toho upravených: 1
Celkový čas: 1.6179 s

Vytvorenie súborov s 30 upravenými segmentami

Segmentov celkovo: 2551
Z toho upravených: 29
Celkový čas: 8.7269 s

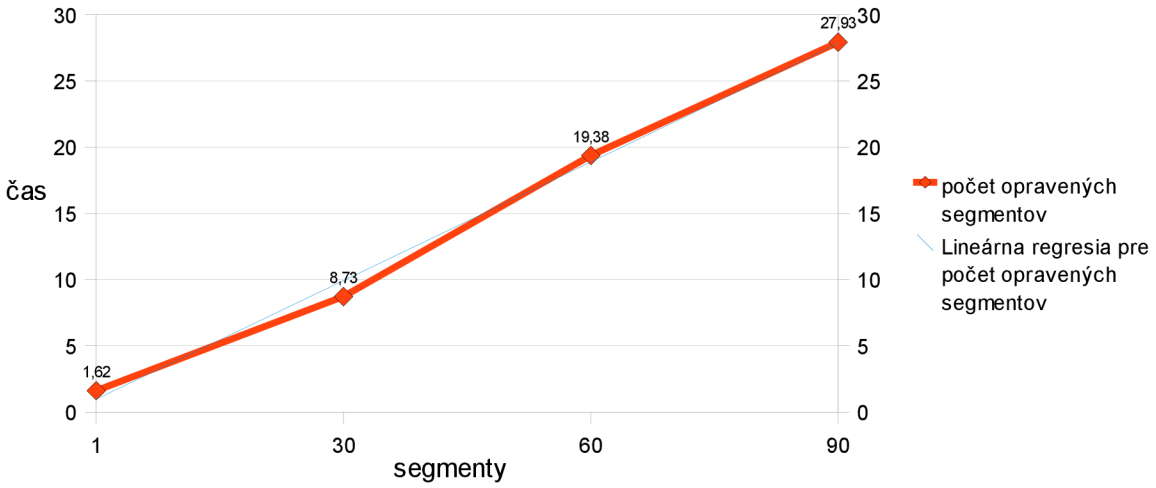
Vytvorenie súborov so 60 upravenými segmentami

Segmentov celkovo: 2551
Z toho upravených: 60
Celkový čas: 19.3832 s

Vytvorenie súborov s 90 upravenými segmentami

Segmentov celkovo: 2551
Z toho upravených: 90
Celkový čas: 27,.9277 s

Môžme si všimnúť lineárnu závislosť u testov s rozličnými počtami opravovaných segmentov. Vytvorenie jednej skupiny súborov bez použitia úprav trvá približne 1,3s. U ostatných testov po odpočítaní tejto hodnoty z celkovej doby trvania výpočtu získame, že vytvorenie jedného nového segmentu trvá približne 0,3s.



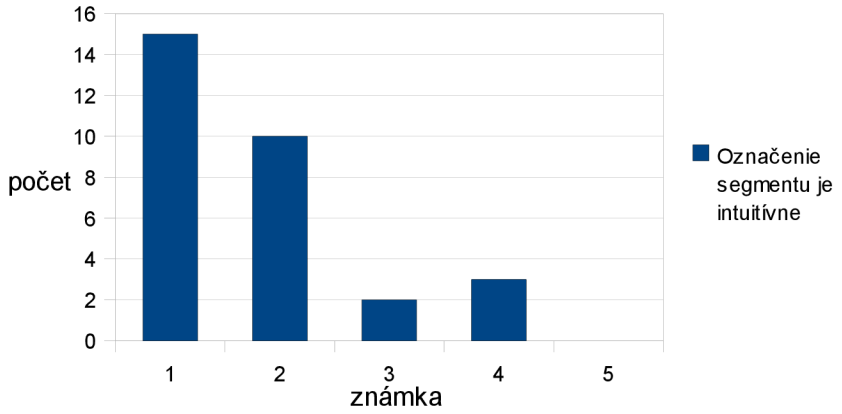
Graf 6.1: Závislosť doby výpočtu od počtu opravovaných segmentov

Testy prebiehali na Acer Aspire 5930G Notebook PC so špecifikáciou:
CPU: Intel(R) Core(TM)2 DuoCPU P7350 @ 2.00GHz 2.00GHz
RAM: 3,00GB
OS: Windows 7 Professional (64-bitový operačný systém)
Web Server: WAMP

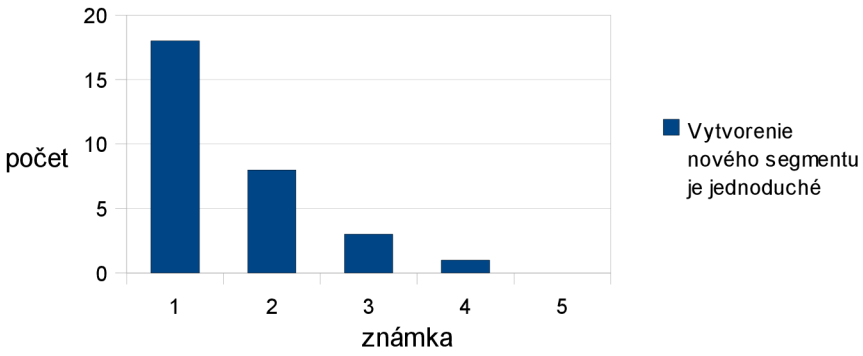
6.3 Prieskum verejnej mienky

Prieskum verejnej mienky bol realizovaný formou rozoslania elektronických formulárov. Zameraný bol na použiteľnosť úpravy transkriptov, keďže zisťovať názor verejnej mienky u prevodu na MySQL je bezpredmetné.

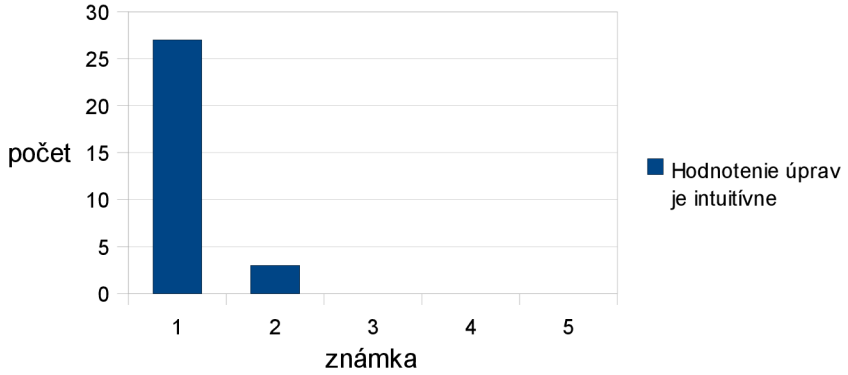
Vo formulári sa vyskytovalo 5 jednoduchých tvrdení, ktoré mali užívatelia ohodnotiť známkou 1 až 5 (1 najlepší, 5 najhorší) podľa ich názoru. Výstupy sú znázornené v grafoch 6.2 až 6.6.



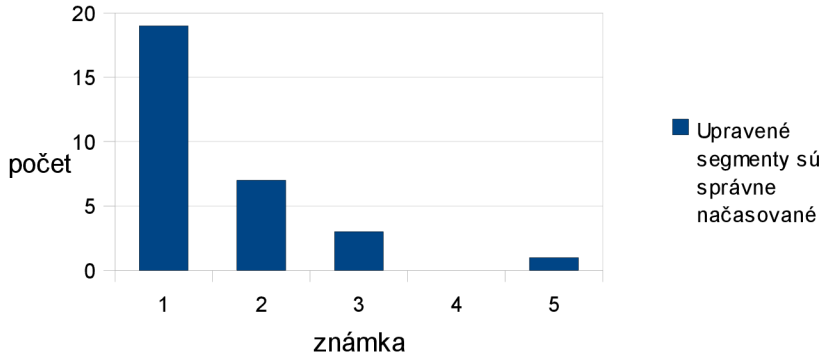
Graf 6.2: Tvrdenie: Označenie segmentu je intuitívne



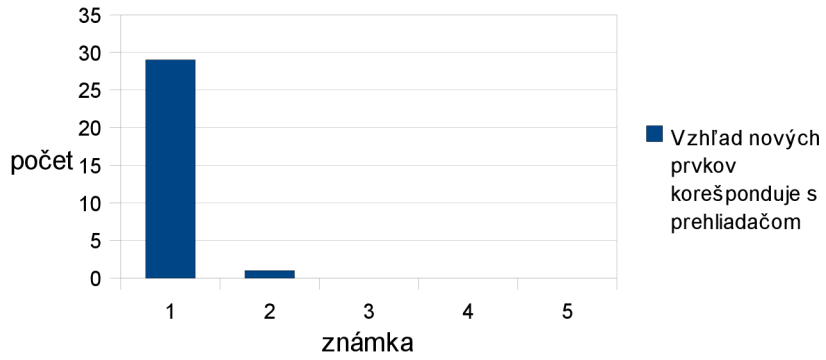
Graf 6.3: Tvrdenie: Vytvorenie nového segmentu je jednoduché



Graf 6.4: Tvrdenie: Hodnotenie úprav je intuitívne



Graf 6.5: Tvrdenie: Upravené segmenty sú správne načasované



Graf 6.6: Tvrdenie: Upravené segmenty sú správne načasované

Dotazník vyplnilo 30 potenciálnych používateľov systému (študenti). Každý z nich odpovedal na všetky otázky z dotazníka.

6.4 Zhodnotenie

V tejto časti budú zhodnotené výsledky testov a prieskumu verejnej mienky. Išlo o objektívne hodnotenia – testy boli realizované za rovnakých podmienok a dotazník vyplnili nezaujatý študenti. Preto môžeme výsledky považovať za smerodatné.

6.4.1 Testy vytvárania nových prednášok

Výsledky z vytvárania nových prednášok dopadli dobre. Predpokladám, že prednášky budú zväčša vkladané jednotlivo, prípadne po skupinách o maximálnom počte 13 (priemerný počet na jednen predmet). Hodnota, ktorá bude predstavovať dobu vloženia prednášky by nemala presiahnuť dobu 1 minúty, keďže pôjde o server, ktorý má vyššiu výpočtovú silu, ako počítač, na ktorom boli testy realizované. Pokiaľ sa budú prednášky vkladať do systému v noci tento úkon nebude pre užívateľov citeľný.

6.4.2 Testy vytvárania nových transkriptov

Úprava transkriptov bude vytváraná užívateľmi počas používania systému. Každý upravený segment musí byť „odsúhlasený“ verejným hlasovaním momentálne nastaveným na skóre s hodnotou 20 a vyššie. Preto sa nepredpokladá, že každý deň bude menených viac ako 100 segmentov. Vytvorenie nových textových súborov s použitím 90 úprav trvá približne pol minúty. Táto hodnota je taktiež postačujúca. Zmeny transkriptov sa budú vykonávať keď bude server najmenej zaťažený – v noci.

6.4.3 Prieskum verejnej mienky

Prieskum verejnej mienky bol zameraný hlavne na grafickú časť aplikácie, ale jedna otázka bola formovaná na kvalitu výstupu z úpravy korektúr. Prieskum dopadol veľmi dobre, dotazovaní boli s aplikáciou spokojný, čo dokazujú aj udelené známky tvrdeniam. Jediné tvrdenie, ktorého hodnotenie sa mierne líšilo bolo: tvrdenie: „Označenie segmentu je intuitívne“. To je spôsobené formou výberu daného segmentu, pretože užívatelia nie sú zvyknutý používať „podržanie ľavého tlačidla myši“. Pri vložení nápovedy, ktorá bude do prehliadača, predpokladám, doplnená sa tento potencionálny nedostatok odstráni.

7 Záver

Výsledkom práce je webová aplikácia zahrňujúca viacero multimediálnych prvkov slúžiaca na výukové účely. Ide o komplexný systém, ktorý bol prevedený z nie príliš šťastného riešenia použitia XML súborov ako úložisko dát na robustnejší systém s použitím databáze MySQL.

Bolo riešených viacero problémov z oblasti návrhu databázového modelu, parsovania dokumentov a prevodu fungujúceho systému na prácu s inou štruktúrou dát.

Doplnená bola zaujímavá časť, ktorou je úprava prepisov reči. Ide o opravovanie rozpoznaných viet, ktoré poslucháč vyhodnotí ako nesprávne a navrhne ich zmenu.

Za vlastný prínos považujem z prvotného hľadiska prácu na spoločnom projekte väčších rozmerov. Teší ma, že som členom týmu, ktorý vytvára systém s takým vysokým potenciálom ako je webový prehliadač prednášok. Dobrou skúsenosťou je analýza rozsiahleho zdrojového kódu, ktorý implementuje už funkčnú časť aplikácie.

V dnešnej dobe je internet zaplnený jednoduchými aplikáciami a rôznymi systémami, ktoré sú väčšine ľudí známe. Chýbajú aplikácie s konkrétnym odbornejším zameraním a aj tak dostupným pre širokú verejnosť.

Ako plus vidím, že výstupy aplikácie budú nápomocné skupine Speech@FIT pri vývoji ich rozpoznávača reči. S opravenými transkriptami sa naučí novým slovám, a tým obohatí svoj už dosť objemný slovník.

Hodnotenie popísané v podkapitole 6.4 naznačuje, že aplikácia je použiteľná v bežnej praxi, či už z prívetivosti užívateľského prostredia alebo z výpočtovej doby pri generovaní dát potrebných pre chod prehliadača prednášok.

Práca vystupuje z diplomovej práce, ale aj neskoršieho úsilia Ing. Jozefa Žižku. Ten vytvoril základnú funkčnosť a moduly uvedené v podkapitole 2.2. Aplikačná časť bola kvalitne implementovaná, a riadne zdokumentovaná.

Na webovom prehliadači, pracuje nielen spomínaný Ing. Jozef Žižka, ale aj ďalší dvaja členovia: Radim Glajc a Antotnín Kalvoda - študenti bakalárskeho programu. Obaja projekt realizujú pod vedením Ing. Igora Szökeho v rámci Bakalárskej práce. Ich úlohou je pridanie ďalšej funkcionality, ako sú časované komentáre k prednáške, možnosť zrýchlenia/spomalenia prednášky a administračná sekcia. Všetky práce sú uvedené v literatúre.

Výsledkom je správne fungujúca aplikácia, ktorá má veľa možností na rozširovanie a napredovanie. Z tohto dôvodu verím, že vývoj aplikácie bude naďalej pokračovať. Vývoj by sa mohol zamerať týmito smermi:

- zaindexovanie zmien transkriptov
- doimplementovanie karmy
- vytvorenie administračného rozhrania na aplikovanie funkcií z tejto práce pre manuálne úpravy prednášok
- návrh a vývoj nových komponent
- rozširovanie aplikácie do viacerých oblastí
- pridanie vlastností informačného systému.

Postupy pri tvorbe boli zvolené na základe skúseností nadobudnutých z doterajšej praxe s realizovaním projektov na báze informačných systémov a spoločných konzultácií. Som rád, že pomocou tejto práce sa znalosti z oblasti vývoja softvéru, ale aj z oblasti algoritmov prehĺbili.

Literatúra

[1] ŽIŽKA, Jozef. *Webový prohlížeč přednášek*. Brno, 2009. 44 s. Diplomová práce. FIT VUT Brno.

[2] Video. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 25. 6. 2006, last modified on 9. 3. 2010 [cit. 2010-01-20]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Video>.

[3] MySQL. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 21.11.2004, last modified on 13. 5. 2010 [cit. 2010-01-20]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/MySQL>.

[4] FALLANSBEE J.: *Streaming media* UK: Focal press, 2006. s. 10-16 ISBN 978-0-240-80863-5

[5] Unified Modeling Language. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 14. 2. 2005, last modified on 25.3.2010 [cit. 2010-01-19]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Unified_Modeling_Language>.

[6] *ArgoUML User Manual* [online]. 2000 [cit. 2010-01-10]. Dostupné z WWW: <http://argouml-stats.tigris.org/documentation/manual-0.30/>.

[7] *PHP: XML Manipulation - Manual* [online]. 21.5.2010 [cit. 2010-01-22]. Dostupné z WWW: <http://www.php.net/manual/en/refs.xml.php>.

[8] *PHP: XML Parser - Manual* [online]. 21.5.2010 [cit. 2010-01-22]. Dostupné z WWW: <http://www.php.net/manual/en/book.xml.php>.

[9] *Úvod | Skupina zpracování řeči* [online]. 2009 [cit. 2010-01-20]. Dostupné z WWW: <http://speech.fit.vutbr.cz/cs>.

[10] Levenshtein distance. In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 18.12.2003, last modified on 2.4.2005 [cit. 2010-05-06]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Levenshtein_distance>.

[11] Apostolico A, Galil Z..: *Streaming media* Oxford University Press, Inc., 1997. s. 123 ISBN 0-19-511367-5

[12] JANOVIČ, Jakub. *Efektivní logistika dopravy vozidel*. Brno, 2008. 36 s. Bakalářská práce. FIT VUT Brno.

[13] GLAJC, Radim . *Webový prohlížeč audio/video záznamů přednášek: přidání funkcionality*. Brno, 2010. Bakalářská práce. FIT VUT Brno.

[14] BAJER, L.: *Algoritmy pro pathfinding* [online], MFF UK, 2006, [cit. 2010-05-08] Dostupný z WWW: <http://ksvi.mff.cuni.cz/~brom/hagents/H-likeAgents4_Bajer060410.pdf>

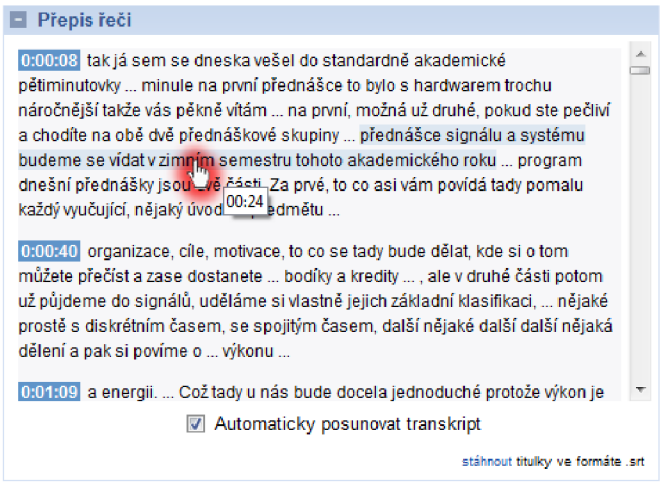
[15] *Audacity* [online]. 2010 [cit. 2010-05-22]. Dostupné z WWW: <http://audacity.sourceforge.net/>.

[16] KALVODA, Antonín . *Webový prohlížeč audio/video záznamů přednášek: přidání funkcionality*. Brno, 2010. Bakalářská práce. FIT VUT Brno.

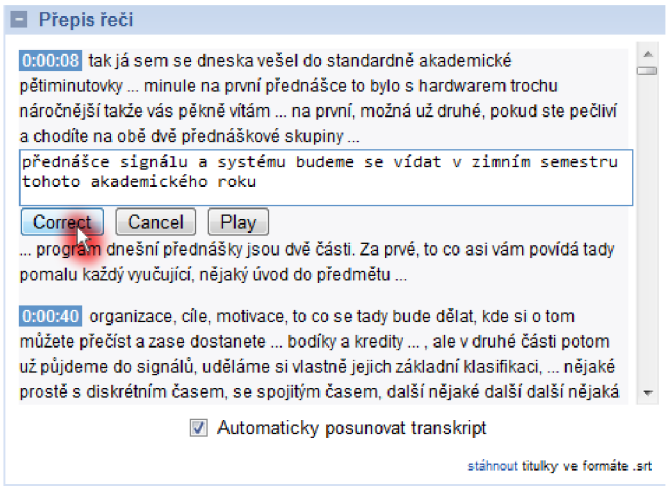
Zoznam príloh

Príloha 1. CD (obsahuje prehliadač prednášok, Browser parser – v zložke parser a túto technickú správu vo formáte PDF)
Príloha 2. Obrazový manuál pre vkladanie nového segmentu

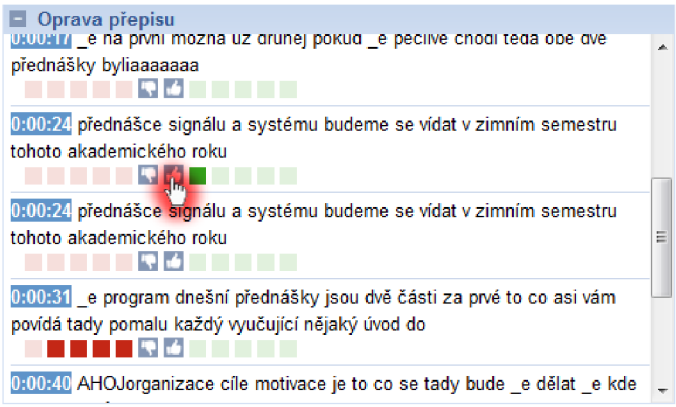
Príloha 2 – Obrazový manuál pre vkladanie nového segmentu a jeho hodnotenie



Obr. 1: Výber segmentu podržaním ľavého tlačidla myši na konkrétnom segmente



Obr. 2: Upravenie vybratého segmentu a potvrdenie úpravy



Obr. 3: Hodnotenie upraveného segmentu „kladným hlasom“