VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOMATICKÉ STAHOVÁNÍ DAT Z INTERNETU PRO TRÉNOVÁNÍ ROZPOZNÁVAČŮ

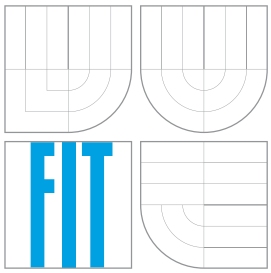
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAN JEŘÁBEK

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOMATICKÉ STAHOVÁNÍ DAT Z INTERNETU PRO TRÉNOVÁNÍ ROZPOZNÁVAČŮ

AUTOMATIC ACQUISITION OF DATA FROM THE INTERNET

FOR TRAINING OF RECOGNIZERS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAN JEŘÁBEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. IGOR SZŐKE, Ph.D.
SUPERVISOR

BRNO 2012

Abstrakt

Tato práce se zabývá návrhem a tvorbou aplikace pro usnadnění stahování audio dat z internetu pro trénování rozpoznávačů. Zabývá se výběrem vhodného zdroje dat a rozborem možností vyhledávání a stahování z vybraného média. Na těchto základech pak rozebírá výběr vhodých nástrojů a knihoven. Součástí je i rozbor implementace, zaměřený na popis klíčových části aplikace. Na závěr jsou shrnuty dosažené výsledky a návrhy na další rozšíření.

Abstract

This paper describes the design and creation of an application to facilitate downloading of audio data from internet for trainign recognizers. Focuses on selection of optimal data source and analyzes options for searching and downloading from selected medium. On this base appropriate selection of tools and libraries is described. Description of the implementation follows where key parts of application are described. In the conclusion are summarised the achieved results and proposal for futher development.

Klíčová slova

Stahování z YouTube, Audiominer, Python, CherryPy, Mako, SQLite, HTML, Javascript, FFmpeg.

Keywords

Download from YouTube, Audiominer, Python, CherryPy, Mako, SQLite, HTML, Javascript, FFmpeg.

Citace

Jan Jeřábek: Automatické stahování dat z internetu pro trénování rozpoznávačů, bakalářská práce, Brno, FIT VUT v Brně, 2012

Automatické stahování dat z internetu pro trénování rozpoznávačů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Igora Szökeho, Ph.D., a že jsem uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Jeřábek
15. května 2012

Poděkování

Rád bych poděkoval panu Ing. Igoru Szökeovi, Ph.D., vedoucímu mé bakalářské práce, za důvěru, vedení a podnětné rady. Dále své rodině, která mě po celou dobu studia podporovala.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Mluvená řeč je jedním z nejstarších způsobů komunikace mezi lidmi. Jedná se také o nejpřirozenější a nejčastěji používanou formu komunikace. Není proto divu, že při neustále se zvyšujících možnostech výpočetní techniky je snaha, aby i počítač mohl být dialogem zapojen do mezilidské komunikace.

Automatickému rozpoznávání řeči počítačem se věnují různé vědecké aktivity již 60 let. V roce 1952, kdy sestavili v Bellových laboratořích systém pro rozpoznávání izolovaných slov (číslovek) pro jednoho mluvčího, započala éra klasifikátorů pracujících na principu porovnávání vzorů. U těchto klasifikátorů je slovo zpracováváno jako celek a je porovnáváno s databází vzorů, přičemž je klasifikováno do té třídy slov, jejímuž vzorovému obrazu se nejvíce podobá. Tyto metody byly v následujících dvou dekádách dále rozšiřovány o nové poznatky a procedury. Se vzrůstajícím výkonem počítačů se rovněž zvětšovaly databáze vzorů používané při klasifikaci.

Na začátku 80. let se do popředí dostává nová skupina klasifikátorů, založená na statistických metodách. Tyto metody modelují slova, ale i celé promluvy pomocí tzv. skrytých Markovových modelů. Nejčastěji jsou vytvářeny skryté Markovovy modely na úrovni subslovních jednotek (foném, slabika, ...) a výsledná promluva je modelována jejich zřetězením, ovšem lze modelovat i jednotlivá slova jako celek pomocí jednoho modelu. Vývoj byl zpočátku zaměřen na systémy pro rozpoznávání celých slov, později v průběhu 90. let se výzkum přesouvá k systémům schopným zpracovat souvislou řeč (nejčastěji čtený text) nezávisle na řečníkovi. Současný výzkum se zaměřuje na další milník u systémů zpracování řeči a tím je přirozená řeč.[11][10]

Všechny výše zmíněné metody mají jeden společný faktor a tím je požadavek na vstupní soubor již klasifikovaných zvukových dat. První skupina klasifikátorů je využívala jako referenční databázi (nejčasteji izolovaných slov), proti které mohla porovnávat nově získaná data (slova). Statistické metody pak tuto kolekci dat využívají pro natrénování pravděpodobnostního modelu, který se následně používá při analýze nových dat.

Klasifikace řeči není jedinou úlohou, na kterou lze aplikovat statistické metody. Skryté Markovovy modely se využívají v dalších úlohách zpracování řeči, například v aplikacích pro identifikaci mluvčího, identifikace jazyka, zjištění emocionálního stavu řečníka a pod. Z výše uvedeného je jasné, že při tvorbě aplikací zpracovávajících řečový signál je poptávka po vstupních datech, vhodných pro konkrétní úlohu, na kterých je možno provést trénování modelu. Pro jednoduché úlohy, jako je rozpoznávání číslovek, je možne vstupní data vyrobit s pomocí mikrofonu a počítače. U složitých systémů, které jsou zaměřeny na klasifikaci přirozené řečí, je ovšem potřeba velké množství dat. Jednou z možností je zakoupení již klasifikovaných dat, ovšem pro malé týmy nebo jedince se zájmem o řečové technologie může

byt toto řešení finančně nedostupné. Jako druhé řešení, které je levné a relativně rychlé, se nabízí využití moderních informačních kanálů, jako jsou rádio, televize a internet. V tomto procesu je ovšem nutno data nejdříve nahrát a poté zpracovat.

Tato práce se zaměřuje na poslední možnost, a to využití internetu při získávání zvukových dat, které by bylo následně možné použít při trénování klasifikátorů řeči, mluvčího a dalších aplikací. Cílem této práce je návrh a implementace této aplikace.

Detailní specifikaci zadání, jeho rozboru a následné analýze požadavků se věnuje kapitola 2. Na základě této analýzy je v kapitole 3 proveden návrh jednotlivých částí aplikace. Kapitola 4 se zaměřuje na výběr nástrojů pro implementaci aplikace. Dále se zaměřuje na popis teorie, jejíž znalost byla nezbytná k úspěšnému dokončení aplikace.

Kapitola 5 se zaměřuje na popis vybraných částí aplikace a jejich implementace. Výsledky a návrhy na vylepšení jsou popsány kapitole 6. Závěr práce v kapitole 7 shrnuje celkové hodnocení práce, dosažené výsledky a její potenciální přínos.

Kapitola 2

Analýza požadavků a jejich specifikace

Základem etapy vývoje každého programu bývá specifikace požadavků na výsledný produkt a jejich následná analýza. V softwarovém inženýrství se tento krok označuje obecně jako analýza požadavků a lze ji rozdělit na tři typy inženýrství[2]:

Získávání požadavků, nejčastěji komunikací se zákazníky a budoucími uživateli. Cílem je dokument o představě, v němž je v hlavních rysech popsáno, co bude nový program dělat. Smyslem tohoto dokumentu je zachycení cílů, jež jsou z pohledu zainteresovaných osob nejpodstatnější.

Analýza požadavků za účelem jejich sjednocení. Každý uživatel má jiné názory a jiné nápady, tudíž je potřeba je sjednotit, aby nedocházelo k jejich konfliktům.

Zapis požadavků vhodnou formou. Obvykle se používá zapis formou modelu požadavků, nebo vytvořením modelu případu užití.

Jako zadavatel, návrhář a programátor v jedné osobmě mám analýzu požadavků zjednodušenu, jelikož nemůže dojít ke kolizi požadavků.

2.1 Zdroj dat

Jak vyplývá z názvu této práce, měla by aplikace využívat internetových zdrojů pro automatické získávání dat, konkrétně audia. Internet je ovšem obrovské médium s nepřeberným množstvím služeb. Je tedy namístě specifikovat jednu základní službu, kterou bude tato aplikace využívat. Další služby budou rozebrány v kapitole 5 jako potenciální rozšíření.

Jako hlavní zdoj pro získávání dat lze využít služby, které poskytují audiovizuální obsah. Tyto služby lze rozdělit na ty, které vysílají v reálném čase, např. televizní a rádiové vysílání, a na ty, které poskytují obsah uložený na serveru. Mezi druhé jmenované lze zařadit služby pro sdílení uživatelských videí (tzv. videohosting), jako jsou YouTube, Vimeo, Metacafe a další. Pokud tyto dva zdroje porovnáme z hlediska využitelnosti, zjistíme, že je mnohem výhodnější použít některou ze služeb pro sdílení vídeí, protože díky tomu, že jsou videa nahrána na serveru, lze:

- Zjistit dopředu velikost dat a délku záznamu. Dále lze přehrát celý jejich obsah a tím určit, zda obsahuje vhodná data v dostatečném množství. U obsahu vysílaného v re-

álném čase (TV) nemáme obvykle možnosti zjistit dopředu délku ani typ vysílaných dat.

- Stahovat maximální možnou rychlostí. U internetové televize probíhá přenos v reálném čase, tudíž pro získání hodiny dat by bylo potřeba hodinu nahrávat. U videohostingu je limitujícím faktorem pouze přenosová rychlost linky a vytížení serveru.

Pro využití v tomto projektu byla jako základní služba zvolen videohostingový server YouTube. Tento server umožňuje svým uživatelům nahrávat, sledovat a sdílet videa a patří mezi největší internetové videohostingové servery. Množství dat na YouTube se nedá odhadnout. Dle statistik pro tisk je každou hodinu nahráno uživateli přibližně 60 hodin videa, což je potenciálně i 60 hodin nových audio dat. Jako další výhody tohoto serveru lze uvést propracované aplikační rozhraní (API) pro vyhledávání videí a možnost vytvářet k uživatelským videím titulky. Pomocí titulků lze následně určit místa, kdy se v nahrávce mluví a kdy ne.

2.2 Specifikace zadání

Poté, co byl zvolen výhodný zdroj dat, nic nebrání specifikovat zadání na výslednou aplikaci. Zadání je psáno obecnou formou z pohledu cílového uživatele a zaměřuje se na funkce, které by měl program zvládat.

Cílem tohoto projektu je vytvořit program, který bude usnadňovat stahování dat ze serveru YouTube a jejich následnou správu v databázi. Představa je taková, že uživatel pouze zadá, kolik hodin audio dat požaduje, a program provede automatické vyhledání, stažení a uložení dat do databáze. Jako další možnost bude k dispozici manuální výběr dat, kdy uživatel provede vyhledání na základě klíčového slova a sám posoudí, o která data má zájem. Zvolená data označí a program pak je opět automaticky zpracuje. Pokud jsou k dispozici k vybranému videu i titulky, bude jejich stažení provedeno automaticky.

Server YouTube je video hostingový server, ale tato aplikace je zaměřena na získávání audio dat, takže další požadavek na program je schopnost extrahovat audio z videa. Extrahované audio je enkodované[1] nějakým kodekem, ovšem pro práci se nejčastěji používají data dekódovaná, takže by bylo vhodné, aby program uměl i převod mezi nějčasteji používanými audio formáty. Poté, co jsou data uložena do databáze, je potřeba je nějak zpřístupnit pro práci. Uživatel si vybere, se kterými daty chce pracovat. Tato data se pak přesunou z databáze do uživatelem definované složky. Poslední požadavek na aplikaci je jednoduchý modul pro postprocessing, který umožní uživateli provádět libovolný soubor operací nad vybranými daty. Toho lze dosáhnout kombinací uživatelských skriptů, ve kterých si uživatel může naprogramovat libovolné operace a jejich následné propojení programem do cílového skriptu, který bude aplikován na vybraná data z databáze.

Výsledná aplikace by jako celek měla být snadno přenositelná mezi hlavními operačními systémy. Pro usnadnění instalace by bylo vhodné minimalizovat externí vazby na knihovy a aplikace třetích stran. Na závěr by uložená data měla být přístupná i po znovuspuštení programu.

[1] encoding - uložení sekvence dat do specializovaného formátu za účelem zvýšení efektivity přenosu nebo uložení informace.

2.3 Existující řešení

S rozvojem internetu v posledních 10 letech stoupá množství aplikací dostupných zdarma, přičemž často můžeme najít podobné aplikace zaměřené na řešení stejného problému. Jako logický krok je tedy vhodné zkusit vyhledat a porovnat existující aplikace. Toto porovnání se zameřuje na open-source řešení.

Jediným podobným projektem, který se zabývá získáváním dat pro trénování klasifikátoru řeči, je projekt VoxForge. Tento projekt je zaměřen na získávání audio nahrávek s přepisem pro využití v open-source nástrojích a projektech zaměřených na rozpoznávání řeči a její převod na text (Speech-to-Text)[13]. K získávání dat se používá java applet, který zobrazuje text pro uživatele. Uživatel pomocí mikrofonu nahraje čtený diktát tohoto textu a odešle jej zpět na server. Celý systém je otevřený a kdokoliv se může zapojit.

Jako potenciální nevýhodu tohoto projektu lze označit jeho specializaci na čtenou řeč. Na aplikace, jako je rozpoznávání emocí, hledání mluvčího nebo přepis přirození řeči, nejsou přístupná data příliš vhodná. Omezením je jednak čtená promluva, dále je většina řečových dat namluvena mužem a počet řečníků není příliš veliký.

2.4 Zápis požadavků

Jelikož je zadání psáno obecně, je vhodné je převést do strukturované formy, která bude zachycovat přesně požadavky kladené na výsledný systém. Požadavky lze rozdělit na tzv. funkční požadavky, které určují, co by měl výsledný systém dělat, a nefunkční požadavky, které specifikují, za jakých omezení musí systém pracovat. Funkční požadavky jsou dále seskupeny do logických celků podle částí programu, které definují.

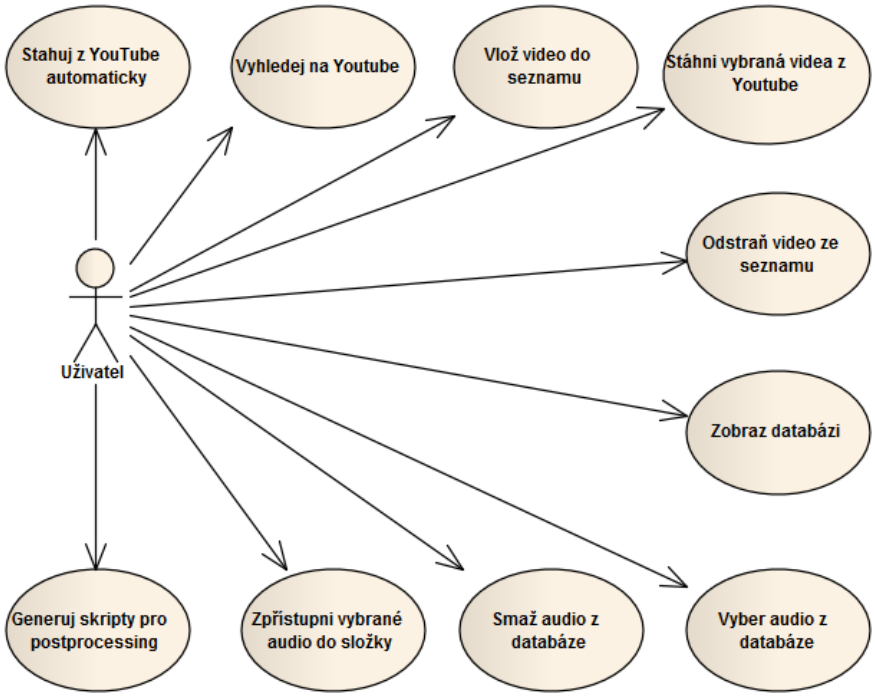
Funkční požadavky :

- Jádro aplikace a databáze dat
  1. Program bude obsahovat databázi audio dat.
  2. Program bude umožňovat jednoduchou správu zvukových souborů v databázi.
  3. Program bude spravovat databázi pomocných souborů (titulky).
  4. Program bude hlídat duplicity na základě video id z YouTube.
- Stahování z YouTube
  1. Program bude vyhledávat na YouTube vhodná videa.
  2. Program stáhne uživatelem zadané množství dat.
  3. Program stáhne uživatelem vybraná videa.
  4. Program stáhne k videu titulky, budou-li k dispozici.
  5. Program po dokončení extrahuje audio z video souboru a to uloží do databáze.
- Zpřístupnění dat z databáze a postprocessing
  1. Program bude umožňovat extrahování audio dat z databáze do pracovního formátu a vybrané pracovní složky.
  2. Program bude umožňovat generování skriptů, které propojí uživatelem již vytvořené skripty.

Nefunkční požadavky :

1. Program bude snadno přenositelný mezi různými operačními systémy.
2. Program by měl být co nejvíce nezávislý na externích knihovnách a programech.
3. Databáze programu bude perzistentní.

Jako další forma dokumentování požadavků se používá model případu užití. Jedná se o grafickou prezentaci, která zobrazuje účastníky systému (uživatele) a činnosti, které mohou v rámci systému vykonávat[4]. Aktuální návrh počítá s jedním typem uživatele. Podpora více typů uživatelů a jejich možná spolupráce v rámci programu může být zajímavým rozšířením této aplikace.



Obrázek 2.1: Model případů užití.

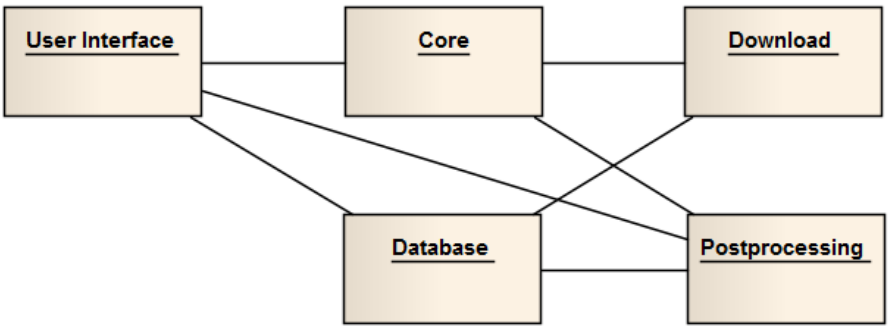
Kapitola 3

Návrh aplikace

Návrh aplikace logicky navazuje na analýzu popsanou v předchozí kapitole. Cílem této kapitoly je navrhnout strukturu aplikace a popsat konkrétní technická řešení, ze kterých se bude vycházet při implementaci.

3.1 Základní moduly aplikace

Pro snadnější implementaci a následnou údržbu kódu je vhodné rozdělit aplikaci na samostatné logické celky, které budou mezi sebou spolupracovat. Jako výchozí základ lze využít rozdělení dle specifikace funkčních požadavků z kapitoly 2.4 a dále je rozšířit. Výsledný návrh je na obrázku 3.1.



Obrázek 3.1: Základní části programu a jejich vazby.

Komunikace mezi moduly bude probíhat dle uvedeného schématu. Uživatelské rozhraní zachytí požadavek od uživatele. V závislosti na čase, který akce potřebuje ke svému vykonaní, je buď vytvořena nová úloha (například stahování), nebo je obsloužena ihned (vyhledavání v databázi, generování skriptů). Typy akcí, které může uživatelské rozhraní zachytit:

- Vytvoření nové úlohy. Uživatel vytvořil požadavek na stahování nebo zpřístupnění dat. Vstup od uživatele je ověřen, a pokud je validní, je vytvořena nová úloha. Tato úloha je předána jádru (modul core), které s ní bude dále pracovat. V případě chybného vstupu je uživateli zobrazeno chybové hlášení.
- Opredace nad databází, konkrétně vyhledávání v databázi nebo mazání prvků z databáze. Pomocí modulu Database je vykonána požadovaná operace a výsledek, pokud

existuje, je zobrazen uživateli. V případě chyby je uživateli zobrazeno chybové hlášení.

- Vygenerování skriptů pro post-processing. Generování skriptů je jednoduchá a rychlá záležitost, proto není potřeba vytvářet novou úlohu.
- Změna aktuálně zobrazovaných informací - uživatel se přepnul do jiné části uživatelského rozhraní. Pokud je potřeba, lze získat informace o aktuálním stavu úlohy z modulu jádra nebo z databáze.

3.2 Jádro

Hlavním úkolem jádra aplikace (modul Core) bude správa a spouštení čekajících úloh. Správa úloh bude řešena následovně. Aktivní bude v daný okmažik pouze jedna úloha. Nově vytvořené úlohy budou zařazeny na konec fronty úloh. Po dokončení aktuální úlohy vezme jádro první úlohu z fronty, označí ji jako aktivní a poté ji spustí. Hlavní třída jádra bude implementována jako vlákno aplikace, které poběží po celou dobu běhu aplikace. Na požádání ukončí hlavní třída nejprve aktivní úlohu a poté i samu sebe.

3.3 Stahování dat

Modul Download zapouzdřuje kompletní proces získávání dat ze serveru YouTube. Jeho činnost lze v krocích popsat jako:

1. Vyhledání videa pomocí aplikačního rozhraní YouTube
2. Stažení videa ze serveru
3. Extrakce audio stopy z video souboru
4. Uložení do databáze
5. Smazání dočasných souborů

Je zřejmé, že kroky 2 a 3 budou časově náročné, proto by bylo vhodné tuto úlohu paralelizovat pro urychlení stahování. Jako nejvhodnější se jeví zapouzdřit kroky 2 - 5 do samostatného pracovního vlákna.

Po vytvoření a spustění hlavního vlákna bude vytvořeno N pracovních podvláken, které budou realizovat samotný proces popsaný body 2 - 5. Pracovní vlákna budou při vytvoření a při čekání na nové stahování uložena v poli neaktivních vláken. Po dobu práce budou pracující vlákna uložena v poli aktivních vláken. Hlavní vlákno bude pracovním vláknům předávat informace o videích, které chce stahnout, a po dokončení jejich procesu analyzuje výsledek stahovaní.

Vzhledem k tomu, že režie při vytváření vláken je poměrně veliká, budou pracovní vlákna existovat, dokud nebude staženo požadované množství dat. Pokud rodičovské vlákno usoudí, že má již dost stáhnutých dat, bude pracovní vlákna ukončovat. Za předpokladu, že máme požadované množství dat a že nejsou aktivní žádná pracovní vlákna, bude úloha považována za dokončenou a vlákno se ukončí. Popis, jak by měla vypadat obsluha pracovních vláken, je zapsán v pseudokódu v bloku 3.1.

Blok kódu 3.1: Popis obsluhy pracovních vláken v pseudokódu

```
dokud není staženo dost dat a pole aktivních vláken není prázdné:
    pro každé vlákno v poli aktivních vláken:
        pokud vlákno dokončilo stahování:
            odeber vlákno z pole aktivních vláken
            zpracuj výsledek stahování
            přidej vlákno do pole neaktivních vláken

    pro každé vlákno v poli neaktivních vláken:
        vyjmi vlákno z pole neaktivních vláken
        pokud není staženo dostatek dat:
            vyhledej nové video
            přiřaď vláknu nově vyhledané video
            vlož vlákno do pole aktivních vláken
            spusť stahování
        pokud je staženo dostatek dat:
            ukonči vlákno
```

3.4 Databáze

Aby byla práce s audio soubory staženými z Internetu efektivní, je nutné uchovávat persistentně některé informace o souboru v databázi. O audio souboru lze obecně zjistit a uložit poměrně velké množství informací, ovšem ne všechny potřebujeme k plnohodnotnému fungovaní aplikace a k nasledné práci s daty. Jako uplně základní byly vybrány tyto položky:

id - primární klíč - unikátní identifikátor audio souboru v celé databázi. Server YouTube katagolizuje svá videa pomocí unikátního textového řetězce. Tento řetězec by mohl být použit jako unikátní klíč položky, ovšem v budoucnu by to mohlo způsobit komplikace při rozšiřování aplikace o další stahovací služby. Proto je jako primární klíč zvolena unikátní číselná hodnota.

video_id - unikátní identifikátor videa. Jedná se o textový řetězec, který identifikuje video na zdrojovém serveru. Tato hodnota bude použita jako jméno uloženého audio souboru a při generování složkové struktury na pevném disku. Dále tato položka může být použita jako cizí klíč do tabulky titulků.

title - popisek originálního souboru na serveru.

duration - délka zvukového záznamu v sekundách.

container - kodek, kterým je audio soubor enkodován. Pro zjednodušení je uložena přípona souboru, která by měla reflektovat použitý kodek.

bitrate - jednotka přenosové rychlosti v bitech za sekundu. Na základě přenosové rychlosti lze určit, jak velké množství informací nahrávka obsahuje (čím větší bitrate, tím více informací).

sample_rate - vzorkovací frekvence zvukové nahrávky v Hz.

channels - počet zvukových kanálů.

Stejně je potřeba definovat položky tabulky pro soubor s titulky.

id - primární klíč, jedná se o unikátní hodnotu v rámci celé tabulky.

video_id - klíč audio souboru, ke kterému tento soubor patří.

filename - název souboru.

Aby nedošlo k zbytečnému plýtvání místem na pevném disku, budou soubory do databáze úkládány ve stejném formátu, v jakém budou extrahovány z videa. Toto pravidlo také uspoří procesorový čas, protože extrahované audio nebude potřeba konvertovat do jiného audio formátu.

3.5 Post-processing

Modul pro post-processing bude zapouzdřovat dvě hlavní činnosti:

Zpřístupnění dat dat pro uživatele, aby s nimi mohl pracovat. Audiodata jsou v databázi uložena ve stejném formátu, v jakém byla vyextrahována z videa. Programy pro trénování skrytých Markovových modelů (např. HTK nebo Julius) ovšem preferují data dekodováná. Dále by nebylo vhodné, aby uživatel pracoval přímo nad daty v databázi, jelikož by mohlo dojít k narušení její konzistence (např. omylem smazaným souborem). Proto budou data v databázi dekódována do uživatelem zadané složky, kde s nimi bude moci následně pracovat. Jelikož dekódování audia lze považovat za výpočetně náročný proces, je při požadavku na zpřístupnění dat vytvořena nová úloha, která je předána jádru aplikace.

Generování skriptů pro pokročilé uživatele. Je možné, že pokročilý uživatel bude mít již vytvořenou nějakou skupinu skriptů pro práci se zvukovými daty. Řekněme, že má vytvořeny skripty s názvy extract_audio, filter, train a evaulate a že je chce aplikovat v uvedeném pořadí na soubor vybraných dat z databáze. Úlohou generátoru pak bude vytvořit dva skripty. První skript s názvem process bude generovat spouštění jednotlivých uživatelských skriptů. Pro minimalizaci chyb se předpokládá, že následující skript dostane jako vstup výstup skriptu předcházejícího. První skript v pořadí dostane na vstup první parametr předaný skriptu process. Výstup tohoto souboru pro daný příklad je uveden v bloku 3.2.

Blok kódu 3.2: Příklad vygenerovaného skriptu process.sh pro linux

```
#!/bin/bash
out=$(sh extract_audio.sh $1)
out=$(sh filter.sh $out)
out=$(sh train.sh $out)
out=$(sh evaulate.sh $out)
```

Skript process spolu s vybranými skripty uživatele budou zkopírovány do složky scripts. Druhý skript s názvem run bude volat vygenerovaný skript process a jako parametr mu předá absolutní cestu k souboru v databázi. Příklad skiptu run je uveden v bloku 3.3.

Blok kódu 3.3: Příklad vygenerovaného skriptu run.sh pro linux

```
#!/bin/bash
export WDIR=/home/output/test
cd /home/output/test/scripts
sh process.sh /home/audiolib/9didzzv1ppixee/9didZv1PixE.ogg
sh process.sh /home/audiolib/xctbhgghhn82s/xctbhGHn82s.ogg
sh process.sh /home/audiolib/h5x6evvbbtt_oee/h5x6eVBT_oE.ogg
sh process.sh /home/audiolib/kk59ttmrs9yydd0/K59Tmrs9YD0.ogg
```

Všechny vygenerované skripty budou uloženy do složky, kterou si zadá uživatel. Součástí hlavního skriptu bude export globální proměnné WDIR, která bude obsahovat absolutní cestu k pracovní složce, ve které se skript run nachází. Jelikož je generování kódu nenáročná operace, není potřeba vytvářet novou úlohu a generování bude provedeno okamžitě. Vygenerovaný skript nebude spouštěn programem, ale uživatel bude spouštět hlavní skript run manuálně. Takovýto postup byl zvolen proto, že skript run může být spouštěn jiným programem, případně je možnost, že jej uživatel bude chtít spustit na jiném, výkonnějším počítači nebo skupině počítačů.

Při generování skriptů bude platit předpoklad, že uživatel ví, co dělá. Aby nedošlo k narušení dat v databázi, bude vhodné, aby první skript z řetězce prováděl rozbalení nebo zkopírování dat do pracovní složky. Jelikož práce s audioformáty je složitá a náročná činnost, bude nejvhodnější využít některý z open-source projektů pro praci s audio a video formáty, jako je MEncoder nebo FFmpeg. Tyto programy pak budou volány automaticky při převodu nebo extrakce audia.

3.6 Uživatelské rozhraní

Na závěr je třeba specifikovat modul, který bude obsahovat potřebné třídy a funkce pro generování a správu uživatelského rozhraní. Hlavním úkolem uživatelského rozhraní je umožnit uživateli aplikaci ovládat a pracovat s daty, které poskytuje. Hlavními požadavky, které jsou v dnešní době kladené na uživatelské rozhraní, jsou přehlednost a jednoduchost. Pro tuto aplikaci se přepodkládá použití grafického uživatelského rozhraní.

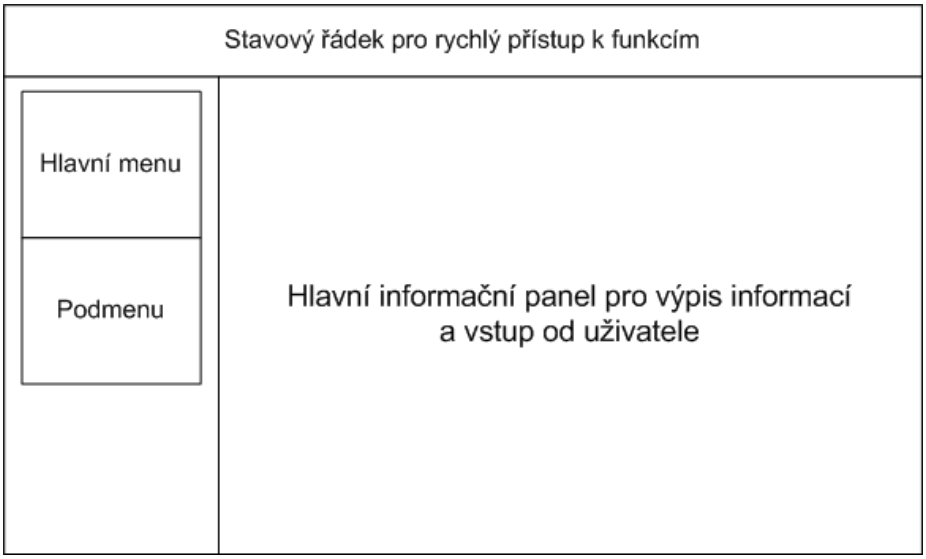
Uživatelské rozhraní aplikace bude rozděleno na 3 hlavní bloky, tak jako na obrázku 3.2:

Hlavní informační panel bude zobrazovat hlavní informace (v závislosti na zvolené sekci programu) jako je aktuální stav úloh nebo výpis z databáze. Dále bude použit pro získávání vstupu od uživatele pomocí vhodně generovaných formulářů.

Menu panel bude obsahovat hlavní a vedlejší menu pro přepínání mezi jednotlivými sekcemi programu. Hlavní menu bude zobrazeno vždy ve stejné formě, vedlejší menu se bude generovat dle aktuálních potřeb zvolené sekce.

Stavový řádek bude umožnovat rychlý přístup k funkcím, případně výpis stavových informací.

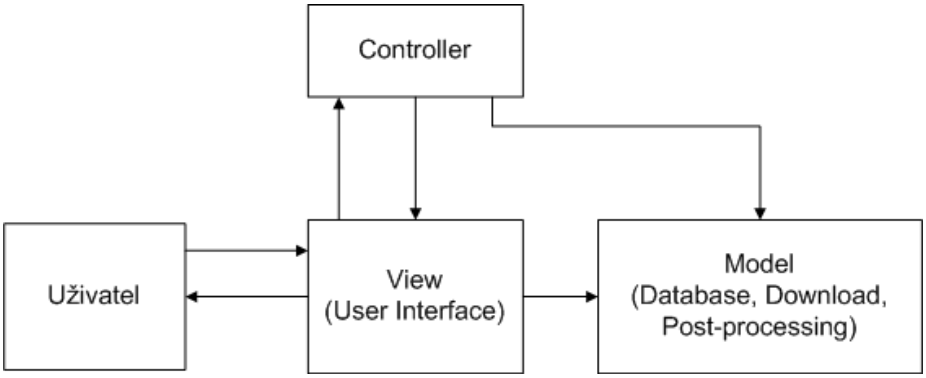
Hlavní třída rozhraní by měla běžet nezávisle na ostatních částech aplikace, ideálně ve vlastním vlákně. Velice populární při návrhu aplikací s uživatelským rozhraním je architektura Model-View-Controler (dále jen MVC). Jak z názvu vyplýva, architektura MVC dělí aplikaci na 3 logické části[5]:



Obrázek 3.2: Návrh uživatelského rozhraní aplikace.

- **Model** - reprezentuje data, se kterými aplikace pracuje.
- **View** - zobrazuje uživatelské rozhraní.
- **Controller** - má na starosti aplikační logiku programu.

Tyto části je pak možné samostatně upravovat, přičemž dopad změn na ostatní časti je výrazně snížen. Obrázek 3.1 zobrazuje aplikaci MVC architektury na návrh modulů z kapitoly 3.1 (obrázek 3.1). V závorkách jsou uvedeny konkrétní části aplikace, spadající pod daný logický blok MVC.



Obrázek 3.3: Aplikace MVC architektury na základní části aplikace.

Kapitola 4

Teorie a použité technologie

Při vývoji aplikace hrají také velkou roli vhodně zvolené nástroje a knihovny. Jejich volba se opírá o funkční a nefunční požadavky definované v kapitole 2.

4.1 Implementační prostředí

Volba vhodného programovacího jazyka je nejdůležitějším rozhodnutím při vytváření aplikace. Vzhledem k požadavku na snadnou přenositelnost se jako ideální volba pro tento projekt jeví některý z rodiny interpretovaných jazyků. Mezi nejpoužívanější v této kategorii patří programovací jazyky Ruby, Perl, Python a Java. Díky tomu, že je kód programu vykonáván v interpretu, a ne přímo operačním systémem, je možné dobře napsaný kód přenášet mezi těmito systémy pouhým zkopírováním aplikace. Z výše uvedených jazyků byl vybrán programovací jazyk Python, a to z následujících důvodů:

- Je k dispozici pro všechny hlavní operační systémy: Windows, Linux, Unix, Mac OS a další.
- Každá distribuce Pythonu obsahuje standardní knihovnu, která disponuje několika desítkami modulů a knihoven, které poskytují široké spektrum nástrojů.[8]
- Jedná se o vysokoúrovňový objektově-orientovaný programovací jazyk. Nezaměřuje se jen na procedurální a objektové orientované principy programování, ale přebírá i další koncepty, jako jsou generátorová notace, principy funkcionálního programování a další.
- Díky široké základně uživatelů existuje pro Python mnoho dalších knihoven a rozšíření, které lze využít buď jako samostatné knihovny, nebo nainstalovat do standardní knihovny jazyka Python.

V době tvorby této práce [1] byly k dispozici dvě hlavní verze jazyka Python, a to 2.7 a 3.2. První verze je označena jako stabilní a je z hlediska vývoje uzavřená. Jediné změny, které jsou prováděny v této verzi, jsou opravy chyb. Druhá verze byla zveřejněna v únoru 2009 jako nová vývojová větev tohoto jazyka. Jelikož nová verze přinesla změny v syntaxi jazyka i způsobu, jakým interpret pracuje s objekty, jsou obě verze navzájem nekompatibilní.

Jako standardní platforma pro vývoj se v současné době stále používá Python verze 2. Je to z toho důvodu, že poskytuje stabilní a odladěné prostředí pro vývoj aplikací. Z toho důvodu byla pro implementaci zvolena aktuální dostupná revize jazyka - Python 2.7.3.

[1] 2011/2012

4.2 Databázový systém

Databáze se dá popsat jako kolekce informací, které jsou vhodně organizovány tak, aby se s nimi dalo co nejjednodušeji manipulovat. Program, který obstarává tuto manipulaci, se nazývá databázový systém (nebo také systém řízení báze dat - SŘBD).

Požadavky na databázový systém u této aplikace nejsou nikterak velké. Jedná se pouze o ukládání informací o audio souboru a o souboru s titulky. Při výběru datábazového systému byly zváženy tyto aspekty:

Podpora API pro Python - existuje API rozhraní pro vybranou databázi v daném programovacím jazyce?

Instalace a údržba - jak složitá je instalace a počáteční nastavení databáze?

Výkon - poskytuje databáze dostatečný výkon pro zvolenou aplikaci?

Funkcionalita - poskytuje databázový systém vhodné funkce pro zvolenou aplikaci?

Podpora databázových systémů pro Python je velice bohatá. Kromě obecného databázového rozhraní ODBC[2] existují knihovny s podporou pro konkrétní databázový systém. Z relačních databází lze jmenovat například MySQL, PostgreSQL, SQL Server, SQLite, z ne-relačních například MongDB, nebo BerkleyDB.

Z podporovaných databázových systémů je třeba vybrat ten nejvhodnější. Jelikož se do databáze budou ukládat pouze základní data o audio soboru a množství takto zpracovávaných dat nebude nikterak veliké, lze požadavky na výkon a funkcionalitu zanedbat. Hlavním měřítkem nakonec zůstává jednoduchost instalace a údržby databáze. Z podporovaných databázových sytémů byl nakonec vybrán relační databázový systém SQLite. Jedná se o open-source aplikaci, jejíž nespornou výhodou je, že se vkládá do aplikace jako její součást. To znamená, že pro svůj běh nespouští nový proces, ale že se spouští spolu s aplikací. Navíc tento databázový systém není potřeba samostatně instalovat, jelikož je vestavěn do standardní knihovny jazyka Python[1].

4.3 Uživatelské rozhraní

Pro vývoj grafického uživatelského rozhraní (dále jen GUI) se obvykle využívá grafický subsystém operačního systému. Tento přístup je ovšem pro tuto aplikaci nevhodný, jelikož každá rodina operačních systémů využívá vlastní grafické knihovny. Tyto knihovny jsou navzájem nekompatibilní, a tudíž by při jejich použití byla výsledná aplikace nepřenositelná.

Jako další možnost lze využít grafických knihoven, které vytvářejí nadstavbu nad grafickým subsystémem operačního systému a díky tomu je aplikace přenositelná mezi platformami, které daná grafická knihovna podporuje. Mezi nejpoužívanější knihovny se řadí Qt, wxWidgets a GTK+. Všechny tyto knihovny existují také ve verzi pro Python, konkrétně PyQt, PyGTK a wxPython. Nevýhodou těchto knihoven je to, že nejsou součástí standardní knihovny jazyka Python a pro použití by musely být doinstalovány na cílovou platformu před instalací aplikace.

V této práci byla pro vytvoření GUI zvolena třetí, ne příliš používaná možnost a tou je vytvoření webového grafického rozhraní. Součástí běžící aplikace bude webový server, který

[2] ODBC - Open Database Connectivity, standardizované programové rozhraní pro přístup k databázovým systémům

bude na základě vnitřního stavu aplikace generovat rozhraní aplikace ve formě WWW stránek, dostupných přes webový prohlížeč. Takovýto přístup má několik výhod. Webový server nepotřebuje spolupracovat s operačním systémem na takové úrovni jako grafická knihovna, tudíž vhodně vybraná knihovna může být zakomponována přímo do aplikace bez nutnosti instalace. Jelikož se webový obsah distribuuje přes počítačovou síť, bude možné aplikaci používat vzdáleně z jiného počítače. Při vhodně vyřešené synchronizaci dat, bude moci aplikaci používat i více uživatelů zároveň.

Na druhou stranu program a GUI pracují obvykle asynchronně pomocí zasílání zpráv. To znamená, že při interakci s GUI je vytvořena zpráva, která je následně zaslána jádru programu. Analogicky v případě nutnosti (například při vzniku chyby) může jádro vytvořit zprávu a zaslat ji GUI, které ji zobrazí uživateli. Tento koncept není zatím u webové aplikace možný. Reakce ze strany uživatele bude zaslána programu, ten na ni zareaguje a předá zpět informace, které budou zobrazeny uživateli. Ovšem program samotný nemá k dispozici prostředky, aby mohl uživatelské rozhraní informovat o změnách, které nastaly. Proto bude nutné implementovat programovou smyčku, která se bude periodicky ptát jádra aplikace na změny.

Pro tvorbu webového rozhraní budeme potřebovat stejné nástroje jako při tvorbě klasického webového obsahu. Použité nástroje lze rozdělit do dvou kategorií podle toho, jestli pracují na straně klienta (webový prohlížeč), nebo na straně serveru (aplikace).

Klientská část aplikace bude postavena na klasické kombinaci technologií pro tvorbu webového obsahu, kterou je HTML, CSS a JavaScript.

HTML je značkovací jazyk pro strukturování a prezentování webového obsahu vytvořený v roce 1990. O 7 let pozdeji byl ve verzi 4.0 standardizován organizací W3C. V roce 1999 byl standard aktualizován na verzi 4.01, která je dnes oficiální standardizovanou verzí[7]. V roce 2008 začaly práce na nové verzi jazyka, označované jako HTML 5. I když je tato verze stále ve fázi vývoje, nejnovější verze webových prohlížečů již implementují základní části jazyka a dovolují tak jeho použití při tvorbě webového obsahu. Pro použití v projektu byla zvolena verze 5. I když ještě není plně specifikována, nejdůležitější části jazyka jsou již podporovány.

CSS je deklarativní jazyk pro definici vzhledu dokumentů, zapsaných pomocí značkovacích jazyků. Značkovací jazyky pouze specifikují strukturu dokumentu, ale neobsahují žadnou informaci o tom, jak by se daný dokument měl vykreslit. O specifikaci vzhledu se stará právě jazyk CSS, který definuje pravidla aplikovaná na jednotlivé elementy značkovacího jazyka.

JavaScript je objektově-orientovaný skriptovací jazyk vyvinutý firmou Netscape v roce 1995. Jeho původní úloha byla validace vstupu HTML formulářu, ovšem časem se z něj stal robustní víceúčelový nástroj pro interakci s webovým obsahem a prohlížečem.

Jelikož definování vzhledu aplikace/webu je práce spíše pro designéra a autor se považuje hlavně za programátora, je v projektu využita CSS knihovna Bootstrap od Twitteru[3]. Jedná se o css knihovnu, která vytváří přívětivé uživatelské rozhraní a podporuje všechny moderní prohlížeče.

Na straně serveru je potřeba vybrat vhodný webový server, který bude distribuovat webové rozhraní připojeným klientům. Jelikož lze předpokládát, že se data v aplikaci budou

[3] http://twitter.github.com/bootstrap/

dynamicky měnit, bude vhodné vybrat i šablonovací systém pro dynamické generování HTML kódu.

Pro Python existuje velké množství webových serverů. Pro tuto aplikaci byla vybrána knihovna CherryPy. CherryPy je Python knihovna, která pomocí jednoduchého rozhraní zpřístupňuje HTTP protokol vývojářům. Mezi hlavní přednosti této knihovny patří[7]:

- Jednoduchost - jeden z hlavních cílu při vývoji CherryPy je udržovat knihovnu co nejjednodušší na použití a konfiguraci.
- Samostatnost - CherryPy nepotřebuje žádné další knihovny pro svůj běh, protože využívá pouze moduly ze standardní knihovny Pythonu.
- Rozšiřitelnost - struktura knihovny umožňuje snadné připojení dalších nástrojů, pokud jsou potřeba.

Pro usnadnění generování HTML kódu existují tzv. šablonovací systémy. Šablonovací systém obvykle definuje syntaxi jednoduchého šablonovacího jazyka, který lze vkládát do HTML kódu. HTML kód je pak před odesláním klientovi zpracován, vložený jazyk je vyhodnocen a tento výsledek je odeslán klientovi.

Jako šablonovací systém, byla pro projekt zvolena Python knihovna Mako. Tato knihovna využívá vlastní sadu skriptovacích značek pro generování HTML kódu. Výsledná šablona je pro zvýšení rychlosti převedena na modul Pythonu. Blok 4.1 ukazuje použití knihovny Mako v HTML kódu pro vygenerování seznamu o třech položkách.

Blok kódu 4.1: Ukazká skriptování pomocí Mako

```
<%
item_list = ["one", "two", "three"]
%>
<div>
        %if item_list:
                <ul>
                % for value in item_list:
                        <li>Item have value: ${value}</li>
                % endfor
                </ul>
        %endif
</div>
```

4.4 Vyhledávání pomocí Youtube API

Server YouTube poskytuje velice propracované aplikační rozhraní (Application Interface, dale jen API) pro integraci uživatelských služeb na další webové stránky, aplikace nebo zařízení. Aplikační rozhraní je rozděleno na dvě části.

Player API dovoluje kontrolovat, jak budou vypadat videa vložená na cizí webovou stránku. Toto rozhraní není v této práci použito.

Data API umožňuje programu nebo webové službě provádět většinu operací dostupných na webových stránkách YouTube, jako je vyhledávání videí, uživatelská správa videí, přístup ke komentářům, správa uživatelského profilu a další. Pro tuto aplikaci je důležité právě vyhledávání videí.

4.4.1 Vytvoření dotazu

Data API funguje na principu dotaz - dopověď. Dotaz na server je formulován jako GET požadavek HTTP protokolu na následující URL adresu

`https://gdata.youtube.com/feeds/api/videos`

K této adrese se následně připojuje seznam parametrů s informacemi, které budou použity při vyhledávání. Parametr se zapisuje ve tvaru `parametr=hodnota`, přičemž jednotlivé parametry jsou od sebe odděleny pomocí znaku `&`. Výsledný řetězec je k URL adrese připojen pomocí znaku `?`. Následující seznam parametrů není úplný a zaměřuje se pouze na parametry použité v projektu.[převzato z [15]]

v - specifikuje verzi API, kterou by měl YouTube server použít, aby korektně zpracoval požadavek. Současná verze je verze 2. Použitím tohoto parametru se zpřístupňují nová rozšíření, která nejsou dostupná v předchozí verzi (například titulky jsou dostupné pouze ve verzi 2). Pokud je tento parametr vynechán, požadavek se bude zpracovávat rozhraním verze 1.

start-index - určuje index prvního výsledku, který by měl být zahrnut do odpovědi, a pracuje ve spolupráci s parametrem **max-result**. Pokud tedy chceme zobrazit výsledky 26-50, nastavíme hodnoty parametru **start-index** na 26 a **max-result** na 25.

max-result - udává maximalní počet výsledků, které by měly být vráceny na dotaz. Defaultní hodnota tohoto parametru je 25, maximum 50.

q - tento parametr specifikuje klíčové slovo nebo skupinu slov, na základě kterých se bude vyhledávat. Pro vyhledávání lze použít i logické spojky NOT (-) a OR (—).

time - tento parametr omezuje rozsah hledání na videa, která byla nahrána během zvoleného časového úseku. Povolené hodnoty jsou `today`, `this_week`, `this_mothn` a `all_time`, které analogicky omezují vyhledávání jen na videa stará maximalně 1 den, 7 dní, 1 měsíc a neomezeně. Výchozí hodnota je `all_time`, tedy bez omezení.

orderby - parametr specifikuje, jakým způsobem budou výsledky v odpovědi seřazeny. V závislosti na hodnotě parametru jsou výsledky řazeny dle:

- `relevance` - relevance.
- `published` - data vydání od nejnovějších po nejstarší
- `viewCount` - sestupně podle počtu zhlédnutí
- `rating` - sestupně dle hodnocení uživatelů

category - umožňuje vyhledat videa, patřící do nějaké kategorické skupiny nebo označená určitým klíčovým slovem. Pro usnadnění je v projektu uživateli nabídnut seznam kategorií založený na souboru keywords.cat[4], přičemž byly vybrány pouze ty hodnoty, které nejsou označené jako deprecated (zastaralé). Seznam kategorií, používaných serverem YouTube ze souboru keywords.cat: `Autos`, `Comedy`, `Education`, `Entertainment`, `Film`, `Games`, `Howto`, `Music`, `News`, `Nonprofit`, `People`, `Animals`, `Tech`, `Sports`, `Travel`. Tento seznam je pouze doporučený, jako vstup může být použito libovolné slovo.

[4]Soubor keywords.cat je dostupný online na adrese http://gdata.youtube.com/schemas/2007/keywords.cat

duration - tento parametr filtruje výsledky podle jejich délky. Jako validní hodnoty parametru lze použít `short` pro videa s délkou do 4 minut, `medium` pro videa s délkou 4 - 20 minut a `long` pro videa delší než 20 minut.

caption - umožňuje omezit vyhledávání pouze na videa, která mají nebo nemají titulky. Pro vyžádání videí pouze s titulky je potřeba nastavit tento parametr na hodnotu `true`. Z důvodu zpětné kompatibility je ovšem nutné připojit k URL požadavku pouze název parametru bez specifikované hodnoty. Například níže uvedený příklad vrátí seznam videí s titulky, které odpovídají termínu podcast:

```
https://gdata.youtube.com/feeds/api/videos?q=podcast&caption&v=2
```

Pro vyhledávání videí bez titulků je potřeba nastavit hodnotu parametru jako `false`. Zde žádné omezení není, je vyžadován jak parametr, tak jeho hodnota.

4.4.2 Zpracování odpovědi

Odpověď ze serveru je defaultně ve formátu Atom. Atom je standardizovaný XML formát určený k publikování obsahu pomocí webových kanálů. Jako další možnosti umožňuje Data API generovat odpověď ve formátech rss, json, json-in-script, and jsonc. Atom je podmnožinou jazyka XML, a tudíž pro něj platí stejná pravidla jako pro XML[3]. Odpověď z YouTube ve v formátu Atom je zobrazena v bloku 4.2. Pro zjednodušení byly vynechány prvky, které nejsou z pohledu tohoto projektu důležité.

Blok kódu 4.2: Část odpovědi z YouTube ve formátu Atom

```
<?xml version='1.0' encoding='UTF-8'?>
<feed xmlns='http://www.w3.org/2005/Atom'>

  <link rel='previous' type='application/atom+xml' href='...'/>
  <link rel='next' type='application/atom+xml' href='...'/>

  <entry gd:etag='W/&quot;CEAAQn47eCp7I2A9WhVVF0s.&quot;'>
    <id>tag:youtube.com,2008:video:Mx3kKVo2NPM</id>
    <title>...</title>
    <media:group>
      <yt:duration seconds='4557'/>
      <yt:videoid>Mx3kKVo2NPM</yt:videoid>
    </media:group>
  </entry>
</feed>
```

Formát Atom definuje pevnou strukturu dokumentu, jehož obsah musí být mezi tagy (značkami) **feed ... /feed**. Na začátku dokumentu je hlavička, která definuje informace platné pro celý dokument. Při zpracovávání odpovědí z YouTube jsou v hlavičce zajímavé pouze tagy **link**. Tyto tagy s atributy rel="previous"a rel="next"obsahují URL adresu na následující respektive předchozí stránku výsledků, generovanou přímo serverem. YouTube doporučuje tyto odkazy používat při procházení výsledků místo manuálního definování URL parametrů **start-index** a **max-result** v dotazu.

Za hlavičkou následuje seznam odpovědí. Každá položka odpovědi je zapoudřena do **entry ... /entry**. Pro potřeby projektu jsou z každé položky načteny informace o délce videa (**yt:duration**), unikátní identifikátor videa video_id (**yt:videoid**) a názvu videa (**title**). Zbytek informací není pro potřeby tohoto projektu důležitý.

4.5 Stahování videí z YouTube

Na základě unikátního identifikátoru videa, `video_id`, lze získat potřebné informace pro stáhnutí videa. Stahování, narozdíl od vyhledávání, ovšem není ze strany YouTube podporováno. Následující podkapitola shrnuje poznatky potřebné k pochopení principu stahovaní videí z YouTube. Pro stáhnutí videa potřebujeme získat přímou URL adresu na konkrétní videosoubor. Toho docílíme správně formulovaným dotazem na adresu

```
http://www.youtube.com/get_video_info
```

Tato adresa je používaná externími přehrávači YouTube pro získání informací o videu, nutných pro přehrávání. Na stránkách YouTube jsou informace, které vrací tato adresa, již vloženy v HTML kódu stránky během jejího generování. Odtud si je může načíst přehrávač videa. Pokud ale vkládáme video z YouTube na cizí stránky, nejsou tyto informace přibaleny a přehrávač si je musí zjistit sám právě z této adresy.

4.5.1 Vytvoření dotazu

Pro správnou funkci je potřeba uvést následující parametry:

ps - činnost tohoto parametu zůstává neznámá. Doporučuje se použít s hodnotou `default`.

eurl - url adresa stránky, kde je vložen YouTube přehrávač. Hodnota může být nedefinována, ale parametr se doporučuje připojit.

gl - tento parametr není na stránkách YouTube dokumentován. Stejný parametr ale používá Google Search API[5], kde specifikuje cílový region pro vyhledávání. Jelikož Google vlastní YouTube, nabízí se předpoklad, že pro své služby standardizoval rozhraní a parametry, takže tento parametr funguje stejně jako u Google Search API[6].

hl - v Data API parametr hl specifikuje preferovaný hlavní jazyk filmu. Předpokládá se, že zde má stejné využití.

video_id - unikátní identifikační řetězec videa na YouTube. Při přehrávání videa jej můžeme najít v adrese stránky jako hodnotu parametru **v**.

el - tento parametr a jeho validní hodnoty byly získány reverzním inženýrstvím flashového přehrávače YouTube[9][7]. Není přesně zdokumentováno, k čemu přepínač slouží. Experimenty provedené na náhodném vzorku 100 videí zjistily, že různá hodnota tohoto parametru vracela vždy stejné výsledky. Vzhledem k množství videí na YouTube je ale více než pravděpodobné, že tento parametr bude mít pro nějaký konkrétní vzorek

[5] https://developers.google.com/custom-search/docs/xml_results

[6] Kódy oblastí jsou dostupné na adrese https://developers.google.com/custom-search/docs/xml_results?hl=en#countryCodes

[7] jako zdroj citace je uveden celý diskuzní server, jelikož konkrétní stránka byla v průběhu psaní práce ze serveru smazána.

videí své využití. Jako validní lze použít hodnoty **embedded**, **detailpage** a **vevo**. Parametr lze i vynechat, ale je doporučeno jako výchozí používat hodnotu **embedded**.

4.5.2 Zpracování odpovědi

Jako odpověď na výše uvedený URL dotaz zašle server řetězec ve formátu, v jakém jsou formátovány parametry URL (tzv. query string), tedy dvojic parametr=hodnota oddělených znakem **&**. Jedná se o soubor všech informací, které přehrávač potřebuje pro správné načtení a přehrání videa. Následující seznam popisuje vybrané parametry důležité pro stahování:

has_cc - Hodnota tohoto parametru je `true` nebo `false` v závislosti na tom, zda k vybranému videu existují titulky nebo ne.

ttsurl - Hodnota tohoto parametru je URL adresa defaultního souboru s titulky pro zvolené video. Tento parametr je ve výsledku zahrnut, pouze pokud má parametr **has_cc** hodnotu `true`.

token - Tento parametr se původně využíval při stahování videí přes url adresu

```
http://www.youtube.com/get_video?id=video_id&token=token
```

avšak tato adresa již nějakou dobu nefunguje a byla nahrazena následujícím parametrem **url_encoded_fmt_stream_map**. Pravděpodobně z důvodu zpětné kompatibility je parametr token stále součástí generovaných informací. Pokud tento parametr ve výsledku chybí, není zvolené video aktuálně dostupné, a tudíž nemá cenu zpracovávat zbytek dotazu.

url_encoded_fmt_stream_map - obsahuje seznam řetězců ve stejném formátu jako tělo odpovědi, ve které se nachází. Každá položka seznamu obsahuje informace o jednom videu, které je dostupné pro přehrávání (a tedy i stažení) a je identifikováno těmito parametry:

url - obsahuje přímou URL adresu na konkrétní video soubor.

type - specifikuje MIME typ video souboru, případně další informace o použitém kodeku.

fallback_host - URL adresa cache serveru.

quality - textová hodnota specifikující kvalitu videa, může nabývat hodnot `small`, `medium` nebo `large`.

itag - unikátní identifikátor typu videa. Odpovídá hodnotě **fmt value** v tabulce na obrázku 4.1. Tento parametr se využívá při volbě nejvhodnějšího videa pro stáhnutí.

4.5.3 Výběr nejvhodnějšího videa

Server YouTube poskytuje přístup ke stejnému videu v různé kvalitě. Kvalita video i audio záznamu je závislá na rozlišení videa a zvoleném kontejneru. Tyto informace shrnuje tabulka na obrázku 4.1. Z této tabulky je patrné, že kvalita audia se zvyšuje s kvalitou videa. Jako ideální bude stahovat videa s rozlišením 720p ve formátu MP4 nebo WebM. Preferovaným

formátem byl vybrán kontejner WebM, jelikož tento kontejner i jeho audiokodek Vorgis Ogg je open-source, a proto by s jeho zpracováním neměly být žádné problémy. Videa s vyšším rozlišením se kvůli jejich velikosti nevyplatí stahovat. Pokud není rozlišení 720p dostupné, vybere se nejbližší nižší rozlišení tak, aby vždy bylo stahováno video s co možná nejlepší zvukovou stopou.

fmt value		5	6	34	35	18	22	37	38	83	82	85	84	43	44	45	46	100	101	46	102	13	17
Default container		FLV				MP4								WebM								3GP	
Video	Encoding	Sorenson H.263		MPEG-4 AVC (H.264)										VP8								MPEG-4 Visual	
	Profile	–		Main		Baseline	High			3D				–				3D				–	
	Resolution progressive	224p	270p	360p	480p	360p	720p	1080p	2304p	240p	360p	520p	720p	360p	480p	720p	1080p	360p	480p	540p	720p	–	
	Resolution VGA	WQVGA	HVGA	nHD	FWVGA	nHD	WXGA	WUXGA	HXGA	–				nHD	FWVGA	WXGA	WUXGA	–				–	
	Max width (pixels)	400	480	640	854	640	1280	1920	4096	854	640	1920	1280	640	854	1280	1920	640	854	1920	1280	176	
	Max height (pixels)	240	270	360	480	360	720	1080	3072	240	360	520	720	360	480	720	1080	360	480	540	720	144	
	Bitrate (Mbit/s)	0.25	0.8	0.5	0.8–1	0.5	2–2.9	3–4.3	3.5–5	0.5		2-2.9		0.5	1	2	–	–				0.5	2
Audio	Encoding	MP3		AAC										Vorbis								AAC	
	Channels	1–2		2 (stereo)																		1	
	Sampling rate (Hz)	22050		44100																		22050	
	Bitrate (kbit/s)	64		128		96	152			96		152		128		192		128	192			–	

Obrázek 4.1: Tabulka závislosti kvality videa a audia z YouTube na použitém kontejneru [převzato z [14]]

4.6 Post-processing

Na závěr kapitoly je potřeba vybrat vhodné nástroje pro práci s audiem. Pro snadné ovládání pomocí Pythonu je nutné, aby zvolený program uměl pracovat z příkazové řádky bez grafického rozhraní a byl multiplatformní. Tyto požadavky splňují následující dvě aplikace:

FFmpeg je open-source multimediální framework pro nahrávání, konverzi a streamování audio a video souborů. Pro svou práci využívá knihovnu kodeků libavcodec. [převzato z [6]]

MEncoder - je nástroj pro převod audio a video souborů, šířený s distribucí multimediálního přehrávače MPlayer. Hlavní zaměření tohoto programu je konverze videa, ovšem práce s audiem je také podporována [převzato z [12]].

Pro práci s audio a video soubory byl vybrán balík nástrojů FFmpeg, konkrétně dvě aplikace, a to analyzátor multimediálních souborů FFprobe a program pro konverzi audio a video formátu FFmpeg.

Kapitola 5

Implementace

Tato kapitola se zabývá vybranými implementačními detaily různých částí aplikace. Celkově se implementace snažila vycházet z návrhu popsaného v kapitole 3.

5.1 Vliv GIL na výkon

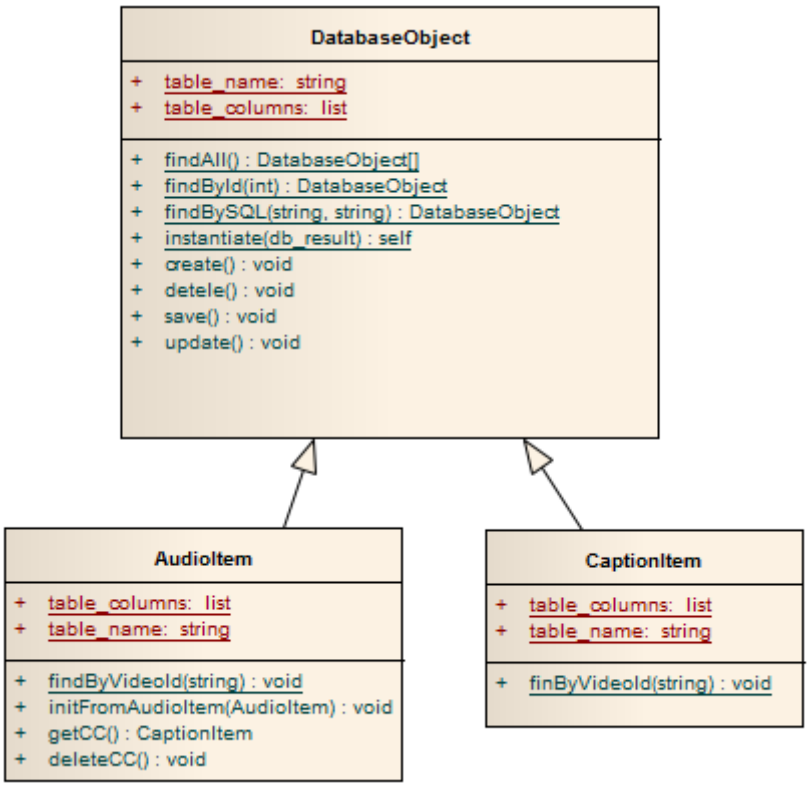
Implementace jádra byla provedena dle doporučení z kapitoly 3.2. Po spuštění se vytvoří nové vlákno, které běží po celou dobu aplikace a zpracovává uživatelem generované úlohy. Úloha je implementována také jako vlákno, a pokud se jedná o úlohu, která stahuje data, je v rámci této úlohy vytvořeno dalších N vláken (dle nastavení), které obstarávají souběžný proces stahování. V nejhorším případě může být jen pro modul jádra aktivních 2+(N) vláken, která by měla pracovat souběžně. Tak tomu ve skutečnosti není. Vlákna v Pythonu jsou sice vytvořena jako plnohodnotná vlákna operačního systému, ale v daný okamžik je aktivní pouze jedno vlákno. Za tuto činnost může globální zámek interpretu (Global Interpret Lock, dále jen GIL). Tento zámek zabraňuje, aby vytvořená vlákna vykonávala svůj bytecode souběžně. Tento zámek je ale nezbytný, protože správa paměti Python interpretu nepodporuje souběžný přístup.

Je tedy namístě položit otázku, zda toto omezení ve výsledku nedegraduje výkon aplikace místo jejího urychlení. Pokud analyzujeme vlákna z hlediska jejich činnosti, zjistíme, že převážnou část své výpočetní doby vlákno čeká na dokončení vstupně/výstupní (dále jen I/O - Input/Output) operace. Při stahování dat čeká, až jsou data načtena z internetu do paměti, při extrakci nebo konverzi audia vlákno čeká na dokončení podprocesu, který vykonává vlastní proces konverze/extrakce. Pokud vlákno čeká na vstup/výstup, na nějakou dobu se deaktivuje a uvolní GIL, který si může zabrat vlákno jiné. Díky tomuto mechanizmu lze efektivně využít čekací doby pro obsluhu více vláken, i když se v danou chvíli vykonává pouze jedno vlákno. Z toho lze usoudit, že využití vláken v modulu jádra nebude způsobovat degradaci výkonu.

5.2 Databáze

Každá tabulka v databázi má v programu definovánu třídu, jejíž instance reprezentuje jeden řádek tabulky. Hodnoty načtené z řádku tabulky jsou uloženy jako atributy objektu. Vzhledem k tomu, že Python je dynamický jazyk, lze k atributům třídy přistupovat pomocí názvu daného atributu, uloženého v textovém řetězci. Takto můžeme pomocí pole řetězců definovat sloupce tabulky a zároveň použít názvy jednotlivých sloupců pro čtení nebo zápis

do atributu stejného jména. Definujme tedy třídní proměnnou, která bude obsahovat názvy sloupců tabulky a pomocí těchto nazvů se bude zároveň přistupovat k atributům třídy. Pokud nyní přesuneme název tabulky do proměnné třídy, můžeme definovat obecný databázový objekt se základní množinou operací nad databází. Pro vytvoření nového objektu, který bude pracovat s vybranou tabulkou v databázi, pak stačí zdědit obecný databázový objekt a přepsat pole s názvy sloupců a atribut s názem tabulky. Stejného návrhu je využito v projektu, kde je implementována bázová třída DatabaseObject, která obsahuje základní metody pro práci s databází. Tuto třídu pak dědí třídy AudioItem a CaptionItem, které specifikují své chování dalšími metodami. Diagram tříd objektů pracujících s databází tak, jak jsou implementovány v projektu, je znázorněn na obrázku 5.1



Obrázek 5.1: Diagram tříd znázorňující hirearchii databázových objektů

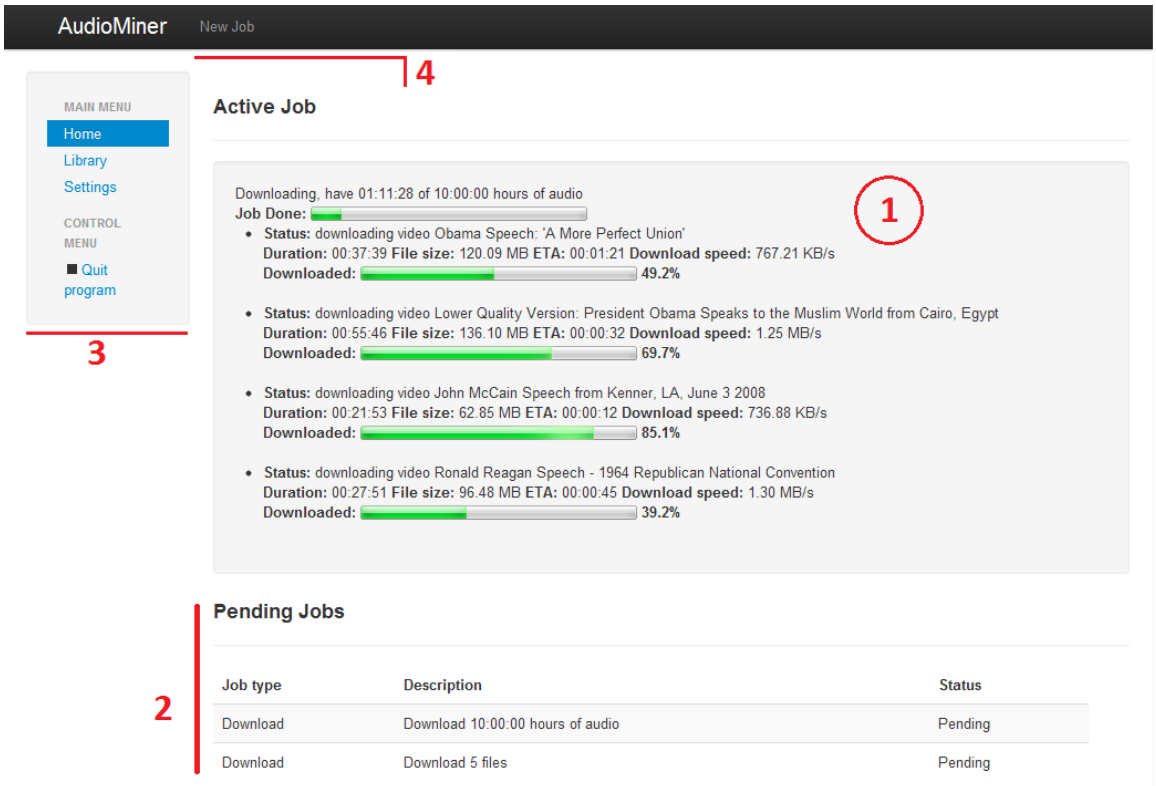
5.3 Uživatelské rozhraní

Uživatelské rozhraní bylo implementováno na základě návrhu obrázku 3.2 z kapitoly 4.3. Výsledný grafický návrh je na obrázku 5.2, na kterém je zobrazena hlavní stránka aplikace. Tato stránka zobrazuje informace o právě prováděném úkolu (1) a dále ve formě tabulky vypisuje úkoly, které čekají na zpracování (2). Hlavní stránka aplikace je přístupná přes **Home** položku v hlavním menu (3). Další položky v hlavním menu jsou **Library** a **Settings**, které odkazují na odpovídající stránky zobrazující výstup z knihovny a nastavení aplikace.

Knihovna slouží pro správu stažených audiosouborů. Soubory je možno prohlížet a v případě potřeby vymazat. Stránka s nastavením umožňuje uživateli definovat základní nastavení aplikace, jako je IP adresa a port, na kterých bude distribuováno grafické rozhraní,

možnosti stahování a úprava složek s nástroji.

Odkaz na vytvoření nové úlohy (4) je přístupný v horním stavovém řádku aplikace. Po kliknutí je automaticky načtena stránka pro vytvoření nové úlohy. Mezi tvorbou úloh lze přepínat pomocí vygenerovaného "Job menu", které se nachází pod hlavním menu aplikace. Kompletní ovládání aplikace je zdokumentováno v příloze C.



Obrázek 5.2: Implementovaný vzhled uživatelského rozhraní aplikace

Kapitola 6

Zhodnocení výsledků

Pokud porovnám výslednou aplikaci s požadavky definovanými v zadání, mohu říci, že aplikace splňuje zadání. Obsahuje modul pro správu stažených dat i modul pro stahování nových dat z Internetu. Součástí aplikace je i modul pro post-processing, který dává uživateli možnost definovat vlastní posloupnost operací nad vybranými daty z databáze, pomocí předem připravených skriptů.

Ve výsledku je aplikace použitelná pro stahování dat z YouTube a jejich správu.

6.1 Možná rozšíření

Během tvorby projektu bylo zjištěno, že použitý návrh není jediný možný přístup k řešní problému. Následující seznam je založen těchto poznatcích a předkládá nápady na možné budoucí rozšíření programu.

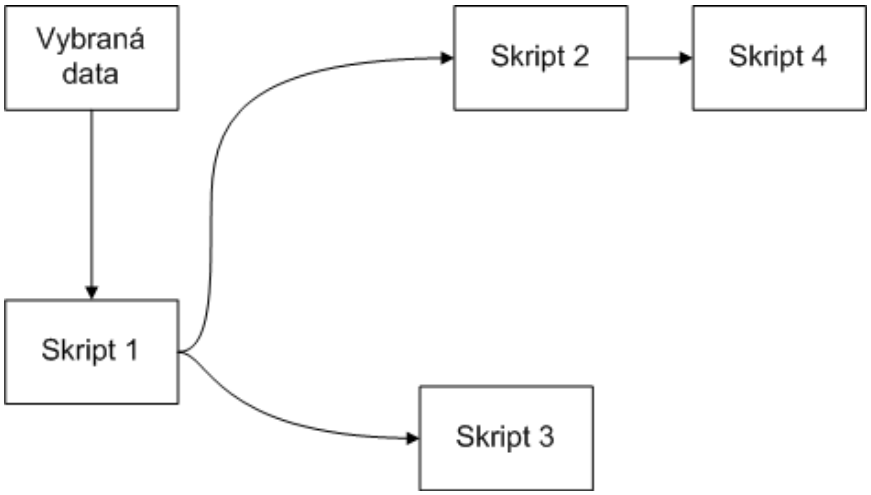
Perzistentní vytváření úloh - v současné verzi aplikace jsou úlohy vytvářeny odpočátku jako vlákna. Tento přístup má ovšem tu nevýhodu, že po vypnutí aplikace jsou ztraceny všechny nedokončené úlohy z fronty úloh. Vhodnější by bylo, kdyby se s úlohou pracovalo jako s objektem databáze, který obsahuje informace o požadované úloze. Tímto by byla zaručena perzistence objektu i po vypnutí aplikace. Vlákno by se generovalo až v době, kdy přijde úloha na řadu, a to z informací uložených v databázovém objektu úlohy.

Modulární návrh - jako další rozšíření aplikace by bylo vhodné rozdělit aplikaci na moduly, které by bylo možno připojovat a odpojovat za běhu aplikace. Specifikací obecného rozhraní pro přístup k jádru aplikace a databázi by se usnadnil vývoj libovolných rozšiřujících modulů.

Websockety - Jak je popsáno v kapitole 4.3, hlavním nedostatkem webového rozhraní je nutnost periodicky ověřovat nové informace. Řešením tohoto problému by mohlo být využití technologie WebSocket pro předávání zpráv mezi klientem a serverem. Technologie WebSocket zpřístupňuje obousměrnou komunikaci mezi klientem a serverem přes TCP spojení. Použitím této technologie by odpadlo periodické dotazování - při výskytu události by jádro informovalo uživatelské rozhraní a naopak. Tato technologie je aktuálně ve fázi testování a ne všechny prohlížeče ji plně podporují.

Post-processing - aktuální modul pro generování skriptů je velice jednoduchý, proto by nebylo špatné jej rozšířit. Jako zajímavý koncept se jeví použití podobného rozhraní,

které se využívá u 3D grafických programů pro tvorbu a specifikaci materiálů (V programu Blender3D se tato funkce nazýva Blender Composite Nodes). Každy materiál nebo modifikátor je reprezentován malým oknem a tato okna lze mezi sebou propojovat čárami. Analogicky k tomuto projektu by byly skripty prezentovány jako objekty na ploše a jejich propojením by vznikl orientovaný graf, na základě kterého by bylo možno generovat mnohem pokročilejší strukturu skriptů. Jednoduchý návrh tohoto rozhraní pro generování skriptů je na obrázku 6.1



Obrázek 6.1: Návrh nového rozhraní pro generování skriptů.

Kapitola 7

Závěr

Cílem této práce bylo vytvořit aplikaci, která bude automaticky stahovat zvuková data z internetu. Tyto zvukové soubory by byly následně využity v procesu výzkumu a tvorby rozpoznávačů řeči a podobných aplikací. Jako výchozí zdroj, odkud budou data stahována, byl vybrán server YouTube.

Práce by nemohla být dokončena bez detailní specifikace a analýzy zadání, která je popsána v kapitole 2. Na základě této analýzy byly v kapitole 3 navrženy a popsány jednotlivé části výsledné aplikace se zaměřením na specifikaci funkčnosti jednotlivých částí aplikace. Obě kapitoly se snažily využívat znalostí o Unifikováném vývoji aplikací.

Jako poslední krok před samotnou implementací bylo potřeba vybrat sadu nástrojů a knihoven a prostudovat možnosti vyhledávání a stahování z YouTube. Získané poznatky jsou publikovány v kapitole 4.

Vybrané části implementace jako konečný vzhled uživatelského rozhraní nebo návrh objektů databáze popisuje kapitola 5. Celkové zhodnocení aplikace je prezentováno v kapitole 6 spolu s návrhy na budoucí rozšíření aplikace, jako je modifikace aktuálního monolitického jádra programu na jádro modulární nebo využití WebSocketů pro asynchronní předávání zpráv mezi uživatelským rozhraním a programem.

Lze říci, že všechny zadané cíle se podařilo splnit. Aplikace plní svůj účel a i přes určité nedostatky ji lze efektivně použít pro získávání zvukových dat z Internetu.

Literatura

[1] Anders, M.: *Python 3 Web Development Beginner's Guide*. Packt Publishing, 2011, ISBN 978-1-849-513-74-6.

[2] Arlow, J.; Neustadt, I.: *UML2 a unifikovaný proces vývoje aplikací*. Comuter Press a.s., 2008, ISBN 978-8-025-115-03-9.

[3] AtomEnabled Alliance: Atom Syndication Format - Introduction. [online], [cit. 2012-05-14].
URL http://www.atomenabled.org/developers/syndication/

[4] Baumann, H.; Baumann, P.; Grssle, P.: *UML 2.0 In Action*. Packt Publishing Ltd., 2005, ISBN 978-1-904811-84-8.

[5] Bernard, B: Úvod do architektury MVC. [online], [cit. 2012-14-05].
URL http://www.zdrojak.cz/clanky/uvod-do-architektury-mvc/

[6] ffmpeg.org: About VoxForge. [online], [cit. 2012-05-14].
URL http://ffmpeg.org/about.html

[7] Hellegouarch, S.: *CherryPy Essentials*. Packt Publishing Ltd., 2007, ISBN 978-1-904811-84-8.

[8] Hellmann, D.: *The Python Standarat Library by Example*. Addison-Wesley, 2011, ISBN 978-0-321-767-34-9.

[9] http://stackoverflow.com/: Stack Overflow. [online], [cit. 2012-05-14].
URL http://stackoverflow.com/

[10] O'Shaughnessy, D.: Invited paper: Automatic speech recognition: History, methods and challenges. *Pattern Recognition*, ročník 41, č. 10, 2008: s. 2965 – 2979, ISSN 0031-3203.
URL http://www.sciencedirect.com/science/article/pii/S0031320308001799

[11] Psutka, J.; Müller, L.; Matoušek, J.; aj.: *Mluvíme s počítačem česky*. Prague: Academia, 2006, ISBN 80-200-1309-1, 752 s.

[12] The MPlayer Project: MPlayer. [online], [cit. 2012-14-05].
URL http://www.mplayerhq.hu/

[13] voxforge.org: About VoxForge. [online], [cit. 2012-05-14].
URL http://voxforge.org/home/about

[14] Wikipedia.org: YouTube. [online], [cit. 2012-14-05].
URL http://en.wikipedia.org/wiki/YouTube

[15] youtube.com: YouTube API - Developer's Guide. [online], [cit. 2012-05-14].
URL http://code.google.com/apis/youtube/getting_started.html

Příloha A

Obsah CD

Přiložené CD obsahuje: zdrojové kódy aplikace, popis instalace, text práce ve formátu tex a pdf. Detailní popis v souboru OBSAH.txt v kořenovém adrešáři CD.

Příloha B

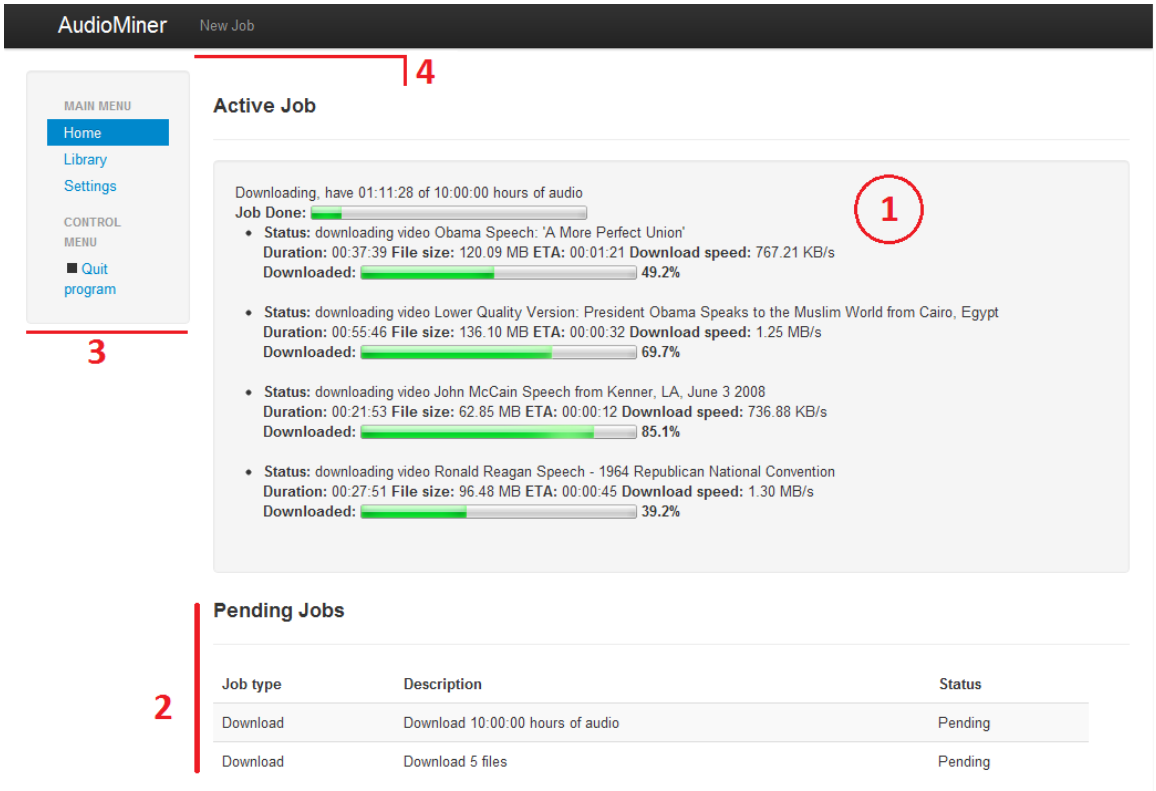
Instalace aplikace

- Aplikace pro svůj běh vyžaduje standardní instalaci Pythonu verze 2.7.* Pro nainstalování stačí zkopírovat aplikaci z CD do vybrané složky na pevném disku.
- Součástí aplikace jsou i 32-bitové binární soubory FFmpeg a FFprobe pro Windows i Linux. Na Linuxu ale může vzniknout problém se sdílenými knihovnami. Pokud nastane při používání aplikace nějaký problém při extrakci nebo konverzi audia, je vhodné doinstalovat FFmpeg na cílový systém a nastavit cestu k jeho pracovní složce buď v souboru audiominer.ini, nebo přes webové rozhraní v sekci Settings.
- Aplikace se spouští příkazem `python audiominer.py` z příkazové řádky ve složce programu.
- Pro přístup k webovému rozhraní zadejte do prohlížeče adresu 127.0.0.1:8090
- Aplikace nefunguje na serveru merlin.fit.vutbr.cz kvůli chybějící knihovně pro SQLite.

Příloha C

Manuál k programu

Tento manuál popisuje ovládání programu audiominer. Základní obrazovka je zobrazena na obrázku C.1



Obrázek C.1: Hlavní okno aplikace

C.1 Vytvoření nové úlohy

Po kliknutí na **New Job** ve stavovém panelu aplikace [obrázek C.1 (4)] se načte stránka pro vytvoření nové úlohy. Uživatel si může vybrat mezi těmito typy úloh:

- **Automatic download** - nabízí vytvoření úlohy, která bude sama stahovat a zpracovávat data z YouTube. Stahování bude ukončeno po stažení odpovídajícího množství

dat. Tím může být buď celková délka stažených dat v hodinách, nebo celková délka řeči na stažených nahrávkách v hodinách. Pokud je zvoleno stahování na základě promluvy, jsou automaticky vyhledávána jen ta videa, u kterých jsou titulky. Stahování může být upřesněno vyplněním formuláře Search settings. Informace z tohoto formuláře jsou použity při vyhledávání videí na YouTube.

- **Custom download** - vytváří rovněž stahovací úlohu, ovšem data pro stahování si musí uživatel vybrat sám. Toho docílí vyplněním formuláře **Search settings** a odesláním na server pomocí tlačítka Search. Prvních 25 výsledků je zobrazeno v tabulce pod formulářem. Kliknutím na ikonu + přidá uživatel vybrané video do fronty ke stáhnutí. Pro zjednodušení budeme tuto frontu nazývat jednoduše košík. Pokud je video v košíku, je místo + zobrazen znak **-**. Ten odebere video z košíku. Pro stáhnutí vybraných videí stačí následně kliknout na **Download**. Pokud je video již stáhnuto a uloženo v databázi, není interakce s tímto videem umožněna.

- **Deploy audio** vytvoří úlohu, která vybrané audiosoubory z databáze rozbalí do uživatelem definované složky. Pro výběr položek je potřeba kliknout na tlačítko **Add audio files**. V načteném okně jsou v tabulce zobrazeny všechny položky v databázi. Vybírání funguje stejně jako u **Custom download**, tzn. kliknutím na + je audio přidáno do košíku, kliknutím na **-** je z košíku odebere. Po vybrání souborů z databáze, lze kliknutím znovu na **Deploy Audio** v Job menu pokračovat dokončením úlohy. Vybrané soubory jsou zobrazeny pod formulářem. Nakonec je potřeba vyplnit název složky, kam se videa rozbalí, a vybrat, jak se bude zacházet se stereo zvukem - zda bude každý kanál uložen do samostatného souboru, nebo budou kanály sloučeny, nebo bude mít nový soubor stejné nastavené kanálů jako originální soubor v databázi.

- **Deploy scripts** - vygeneruje postprocessing skripty na základě výběru uživatele. Rozhraní je zobrazeno na obrázku C.2. Před vygenerováním skriptů je potřeba vybrat z databáze audiosoubory, které budou předány na vstup prvního skriptu v řetězci. Výběr audio dat se provádí stejně jako u **Deploy audio**. Po selekci dat z databáze si uživatel vybere, které skripty bude chtít aplikovat na vybraná data. Dostupné skripty jsou v levém seznamu. Pomocí **Add skript** jsou vybrané skripty přidány do pravého seznamu. Tlačítko **Remove skript** odstraní vybrané skripty z pravého seznamu. Skripty v pravém seznamu jsou použity ve vygenerovaných skriptech, a to ve stejném pořadí, v jakém jsou uloženy v seznamu. Pro dokončení operace je potřeba definovat složku, kam budou skripty uloženy. Dokončení úlohy a vygenerování skriptů je provedeno po kliknutí na tlačítko **Generate**. Skipty jsou vygenerovány okamžitě.

C.2 Nastavení aplikace

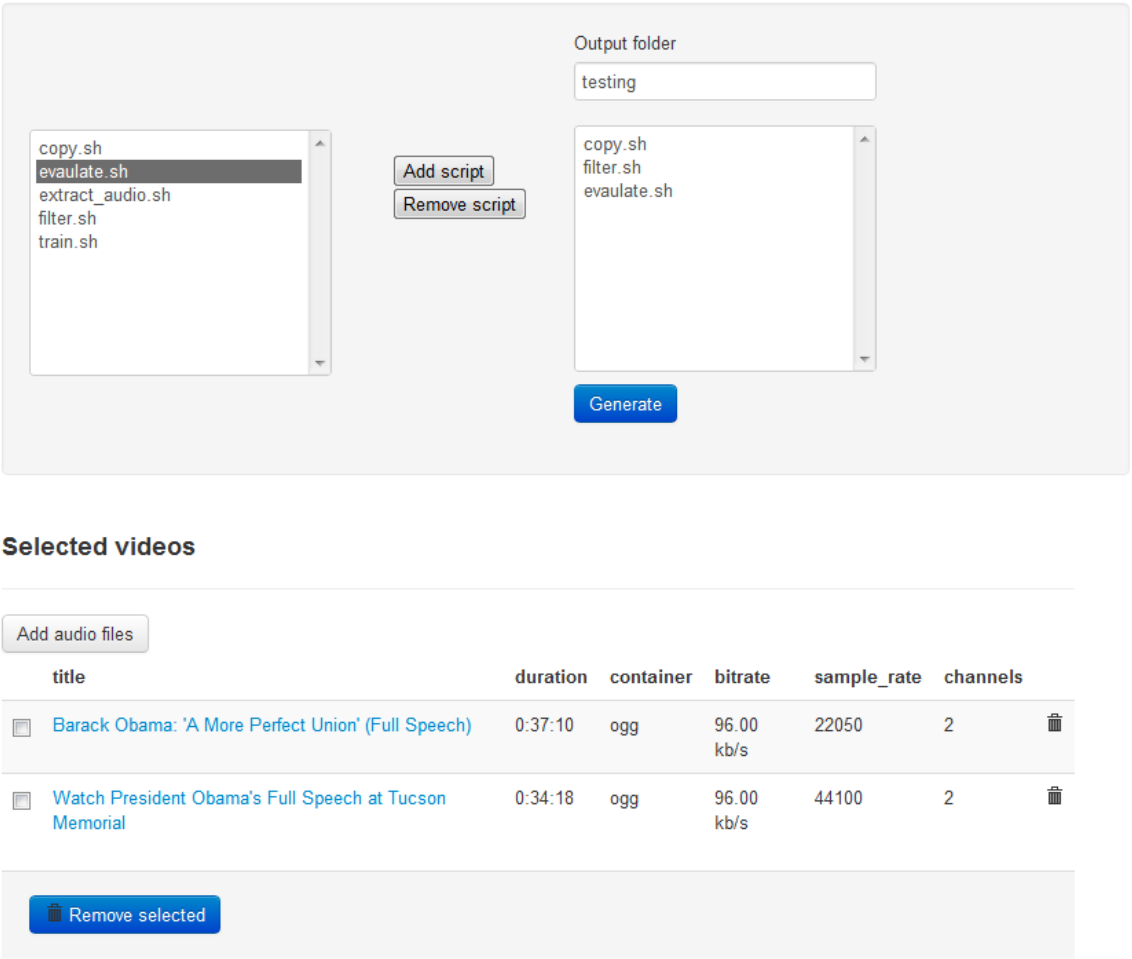
Formuláře pro nastavení aplikace jsou dostupné z Hlavního menu [obrázek C.1 (3)] pod odkazem **Settings**. Zde je možne provést základní nastavení aplikace.

AudioMiner Web server settings - nastavení IP adresy a portu, na kterých bude pracovat web server. Změny v nastavení se projeví po restartu aplikace.

AudioMiner Web server authentication - vypíná/zapína HTTP autentizaci pro přístup k webovému rozhraní. Změny v nastavení se projeví po restartu aplikace.

Download Settings - zde je možno specifikovat nastavení dočasné složky pro stahovaní a počet stahovacích vláken

Tools Folders - zde se nastavují složky s nástroji (FFmpeg) a výstupní složka, ve které jsou vytvořeny podsložky s vygenerovanými skripty nebo rozbaleným audiem.



Obrázek C.2: Formulář pro generování post-processing skriptů