Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jan Češnjevar

Multimedijski predvajalnik

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: viš. pred. dr. Alenka Kavčič

Ljubljana 2012

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00224/2012
Datum: 03.04.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **JAN ČEŠNJEVAR**

Naslov: **MULTIMEDIJSKI PREDVAJALNIK**
MULTIMEDIA PLAYER

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Proučite tehnologije in orodja za izdelavo predvajalnika večpredstavnostnih vsebin. Izdelajte multimedijski predvajalnik, ki bo omogočal urejanje besedila, prikaz slik, predvajanje glasbe in predvajanje videa. Proučite tudi možnost integracije spletnih storitev za pridobivanje podatkov o predvajani glasbi in videu ter za samodejno pridobivanje filmskih podnapisov. V izdelan predvajalnik smiselno vključite tudi uporabo navedenih spletnih storitev.

Mentor:
Alenka Kavčič
viš. pred. dr. Alenka Kavčič

Dekan:
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 1

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Jan Češnjevar, z vpisno številko **63090208**, sem avtor diplomskega dela z naslovom:

Multimedijski predvajalnik

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Alenke Kavčič,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 6. julija 2012 Podpis avtorja:

Zahvaljujem se mentorici viš. pred. dr. Alenki Kavčič za prijaznost, pomoč, nasvete za izboljšave in popravke pri izdelavi diplomskega dela. Zahvaljujem se družini za vso podporo v času študija.

Kazalo

Seznam uporabljenih kratic in simbolov

API - Application Programming Interface
AWT - Abstract Window Toolkit
CSS - Cascading Style Sheets
FFT - Fast Fourier Transform
HTML - HyperText Markup Language
HTTP - HyperText Transfer Protocol
JAR - Java ARchive
JMF - Java Media Framework
JRE - Java Runtime Environment
JSON - JavaScript Object Notation
JVM - Java Virtual Machine
MDI - Multiple Document Interface
PDF - Portable Document Format
URL - Uniform Resource Locator
VLCJ - Java Framework for VLC
XML - Extensible Markup Language
XML-RPC - XML Remote Procedure Call

Povzetek

Diplomska naloga opisuje multimedijski predvajalnik. V diplomski nalogi so opisane tehnologije, orodja, funkcionalnosti in storitve, ki smo jih uporabili pri razvoju. Predvajalnik sestavlja program za urejanje besedil, slik, predvajanje zvoka in video vsebin. Razvili smo ga v programskem jeziku Java in Processing, pri čemer smo za gradnjo uporabniškega vmesnika uporabili okolje NetBeans IDE. Programom smo poleg osnovnih operacij, ki jih ima večina urejevalnikov ali predvajalnikov, dodali dodatno vrednost oziroma dodatne značilnosti preko najdenih knjižnic (arhiv JAR). Pri razvoju smo poudarili povezovanje programa s spletom. Preko aplikacijskih programskih vmesnikov smo dostopali do storitev EchoNest in OpenSubtitles, pri čemer smo zgradili dodatna dva svoja vmesnika s pomočjo orodja Firebug.

Ključne besede:
Multimedija, spletne storitve, Java, aplikacijski programski vmesnik, knjižnice.

Abstract

This article describes a multimedia player. It contains a review of technology, tools, functionalities and services which were used during its development. The whole player is made of several components: programs for text manipulation, image manipulation, playing music and videos. It was written in the programming languages Java and Processing. The graphic user interface was built with NetBeans IDE. The player supports basic operations typical for other multimedia players and it has some advanced operations, which were made with the help of additional external libraries. We encouraged the use of the world wide web in our application. This was done by using application programming interfaces. We used EchoNest and OpenSubtitles API, and developed two of our own with the help of a tool called Firebug.

Keywords:
Multimedia, web services, Java, Application Programming Interface, libraries.

Poglavje 1

Uvod

Pojem "multimediji"v računalništvu razumemo kot skupek več stvari: besedila, slike, zvoka, videa in animacije.

Namen diplomske naloge je bil izdelati splošen multimedijski predvajalnik, ki bi znal delati z besedilom, sliko, zvokom in video vsebinami. Uporabniki programa bi na enemu mestu lahko pridobili večino orodij za upravljanje z multimediji. Za vsakega izmed pojmov bi se ustvaril svoj podprogram, vsi skupaj pa bi bili združeni v celoto. Posamezen podprogram (*npr. urejevalnik besedil*) bi poleg osnovnih operacij (*prilepi, kopiraj*) omogočal bolj kompleksne operacije (*npr. pridobivanje besedila iz dokumenta pdf*), te pa bi bile izdelane s pomočjo že obstoječih knjižnic. Ideja je bila raziskati že obstoječe knjižnice in storitve, ki so na voljo, in jih uporabiti v predvajalniku. Poudarek je bil na uporabi spleta.

Ker je tema diplome razvoj multimedijskega predvajalnika, smo se ukvarjali predvsem s programskimi problemi. Programski jezik Java je zelo dobro podprt s primeri kode in dokumentacijo, kar je bil eden izmed razlogov za njegovo uporabo.

Poglavje 2

Tehnologije in orodja

2.1 Razvojno okolje NetBeans IDE

NetBeans IDE [1] je integrirano razvojno orodje, ki ga je moč uporabiti na operacijskih sistemih Windows, Mac, Linux in Solaris. Uporablja se za gradnjo namiznih, spletnih, mobilnih in drugih aplikacij. Razvijalcem omogoča uporabo programskega jezika Java, Groovy, C/C++, strežniškega skriptnega jezika PHP in skriptnega jezika JavaScript s tehniko Ajax.

Orodje poenostavlja gradnjo uporabniških vmesnikov z že narejenimi komponentami, kot so glavno okno (*JFrame*), vsebovalnik plošča (*JPanel*) in druge. Deluje na konceptu nanašanja komponent na delovno površino. Komponentam se določijo lastnosti in dogodki, ki se prožijo ob določenih trenutkih. Samo orodje zgradi del kode in pohitri razvoj aplikacij.

2.2 Programski jezik Java

Java [2] je popularni splošno namenski in objektno orientiran programski jezik. Njegova odlika je prenosljivost, kar pomeni, da naj bi se programi, napisani v Javi, izvajali podobno na vsaki platformi. To je doseženo s pretvorbo programa v Java bitno kodo in ne v strojno kodo. Prevedena koda se nato izvede na navideznemu stroju, imenovanem JVM (*Java Virtual Ma-*

chine). Za poganjanje programov na JVM potrebuje uporabnik izvajalno okolje JRE (*Java Runtime Environment*), ki priskrbi potrebne knjižnice za JVM.

Sam programski jezik ima na spletu zelo veliko primerov (*tutorial*) in dokumentacije. Prenosljivost in velika podpora s primeri sta bila glavna razloga, da smo se odločili za uporabo Jave.

2.3 Programski jezik Processing

Programski jezik Processing lahko razumemo kot nekakšno poenostavitev Jave. Celotno okolje je napisano v Javi in tudi programi, spisani v Processingu, se pretvorijo v Javo in se nato izvajajo kot Java programi. Processing v primerjavi z Javo poenostavlja programiranje, saj uporabniku ni potrebno poznati konceptov, kot sta razred in objekt [3]. To je že integrirano v Processingu. Hkrati pa so programi tudi krajši. Primer kode, ki poskrbi za predvajanje glasbene datoteke:

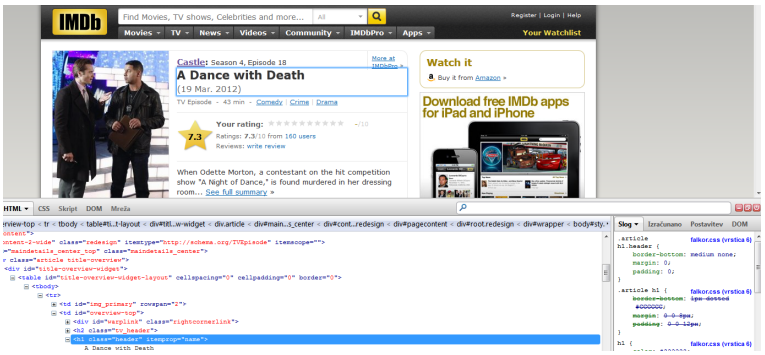
```
Minim minim;
AudioPlayer player;
player = minim.loadFile(''something.mp3'');
player.play();
```

V Processingu so potrebne štiri vrstice kode, da se predvaja pesem. Pri Javi bi morali implementirati tudi metodo play().

Oba programska jezika smo uporabili pri realizaciji glasbenega predvajalnika.

2.4 Vtičniki in orodje Firebug

Obstajajo dodatki za spletne brskalnike, imenovani vtičniki (*plugin*), ki razširijo lastnosti brskalnika. Lahko predvajajo video vsebine, preverijo stran za viruse, omogočajo predvajanje animacij in podobno. "Add-on"je širši pojem



Slika 2.1: Uporaba orodja Firebug za spletno stran www.imdb.com. Levi del prikazuje kodo HTML, desni pa slog CSS.

od vtičnika in lahko pomeni tudi druge stvari. Vtičnikov je zelo veliko in so namenjeni za določene tipe brskalnikov. Brskalnik Firefox ima na primer več tisoč vtičnikov.

Pri delu nam je prav prišlo orodje Firebug, ki ga je možno vključiti v brskalnik Firefox. Je eno izmed najbolj priljubljenih orodij za razvoj v spletu [4]. Uporabnik lahko s klikanjem na določene dele strani dobi vpogled v kodo HTML (*HyperText Markup Language*) in slog CSS (*Cascading Style Sheets*), ki sestavljata del strani. Hkrati lahko uporabnik spreminja kodo in obliko strani v realnem času. Te zmogljivosti smo uporabili za pridobivanje podatkov iz strani www.podnapisi.net in www.imdb.com (*Internet movie database*) pri realizaciji predvajalnika videa. Podobna orodja obstajajo tudi za druge brskalnike. Opera, Explorer in Chrome imajo že vključena podobna orodja. Pri Operi do orodja dostopamo preko kombinacije tipk CTRL+SHIFT+I, pri slednjih dveh pa s klikom na F12. Slika 2.1 prikazuje gnezdeno kodo HTML in slog za izbran element.

2.5 HTML in CSS

HTML je označevalni jezik za izdelavo spletnih strani [18]. Znotraj strani HTML so elementi sestavljeni iz značk (<html>). Običajno so značke se-

stavljene iz dveh delov (začetna in končna značka) ali pa so prazne. Predstavljene so s špičastimi oklepaji. Znotraj teh značk pa je besedilo. Pri gledanju strani z brskalnikom se ne prikazujejo značke, ampak vsebina med značkami. Izgled spletne strani (barvo ali pa kakšen okvir) se lahko določi v samih značkah ali pa slog definiramo ločeno. Za slog spletne strani se danes uporabljajo tako imenovane stilne predloge (*CSS*). Te lako definiramo v glavi strani HTML, v posameznemu elementu HTML ali pa v ločeni datoteki. Prednosti stilnih predlog je veliko, ena izmed njih je, da zagotavljajo enotno obliko za vse elemente določenega razreda ali tipa.

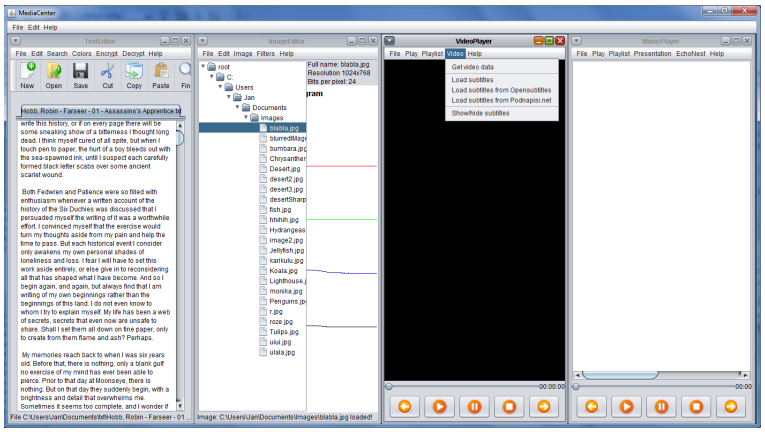
2.6 Aplikacijski programski vmesnik

Aplikacijski programski vmesnik omogoča komunikacijo med dvema programskima komponentama [20]. API (*Application Programming Interface*) lahko vsebuje podatkovne strukture, razrede, spremenljivke in ostalo. Lahko predstavlja celoten vmesnik, funkcijo ali pa serijo vmesnikov [20]. V objektno orientiranih programskih jezikih API predstavlja opise množice definicij razredov, ki imajo določena obnašanja. V primeru razvijanja aplikacij za splet se jim pravi spletne storitve. Te preko zahtev HTTP (*HyperText Transfer Protocol*) in vsebin, ki so priložene v formatu XML (*Extensible Markup Language*) ali JSON (*JavaScript Object Notation*), pošljemo na strežnik in dobimo odgovor HTTP. Primer storitve je Google Maps.

Poglavje 3

Izgled in funkcionalnosti predvajalnika

Multimedijskih programov je dandanes zelo veliko. Uporabljamo jih vsak dan bodisi za delo bodisi za zabavo. Za urejanje besedil in dokumentov uporabljamo orodja, podobna Microsoft Officeu, ali pa kakšne odprtokodne enačice. Picasa, Photoshop in drugi so nepogrešljivi za oblikovanje in pregledovanje slik. Predvajalnikov glasbe in video vsebin je veliko, od Winampa, BSPlayerja, VLCja in drugih. Vse to so kompleksna orodja z naprednimi zmogljivostmi. Zanimalo nas je, kako se zgradi predvajalnik, kaj vse je potrebno narediti zanj in kako obdelovati s podatki. V svoj predvajnik smo hoteli dodati značilnosti, ki smo jih pri uporabi prej naštetih predvajalnikov pogrešali. Uporabno se nam zdi, da je pri urejanju besedil možno odpreti datoteke, ki vsebujejo točno določene besede, ali pa iskanje vseh skladb avtorja, ki se trenutno predvaja. Zelo priročno je tudi, da je pri predvajanju video vsebin moč poiskati podnapise, ne da bi se za to morali sprehajati po določenih spletnih straneh. Kot smo povedali, so zgoraj našteta orodja kompleksna in omogočajo skoraj vse. V razvoju projekta smo se zato osredotočili le na eno ali dve funkciji iz posamezne teme in ju realizirali.



Slika 3.1: Prikaz izgleda aplikacije. Okna so razvščena drugo poleg drugega.

3.1 Izgled in Swing

Želja projekta je bila izdelati aplikacijo MDI (*Multiple Document Interface*). To pomeni, da so vsa okna vsebovana v enemu oknu (staršu). Imamo glavno okno, v katerem so lahko okno urejevalnika besedil, okno urejevalnika slik, okno predvajalnika glasbe in okno predvajalnika video vsebin. Glavno okno omogoča skupen meni, vsak podprogram ima pa tudi svojega. Okna imajo robno poravnavo. V vseh oknih se v zgornjem delu nahaja svoj lasten meni. Pri urejevalniku besedil in urejevalniku slik se v spodnjem delu nahaja statusna vrstica, kamor se izpisujejo sporočila. Predvajalnika glasbe in videa pa imata na tistem mestu gumbe za upravljanje. Pri obeh (urejevalnik slik, urejevalnik glasbe) je v sredinskem delu glavni prostor (pri besedilu tekstovno polje, pri slikah plošča), ki deluje na osnovi zavihkov. Pri slikah je ob levem robu drevo poti do slik, na desnem pa izris histograma in izpis podatkov slike. Urejevalnik besedil na robovih nima ničesar, medtem ko imata predvajalnika glasbe in video vsebin na robovih prostor za sezname predvajanja. Slika 3.1 prikazuje zgoraj opisan izgled aplikacije.

Celoten projekt je sestavljen večinoma iz komponent Swing. Kot alternativo bi lahko uporabili tudi AWT (*Abstract Window Toolkit*), a bi morali aplikacijo zasnovati drugače, saj AWT ne podpira toliko komponent kot

Swing.

Če v programu uporabljamo komponente AWT, se izgled programa prilagodi grafičnemu uporabniškemu vmesniku operacijskega sistema, na katerem se izvaja [6]. Vsebuje t.i. težke komponente, kot sta glavno okno in splošno pogovorno okno (*Dialog*), za katere mora za izris poskrbeti gostujoči operacijski sistem. Program z AWT komponentami skrije podatke o grafičnemu vmesniku, od tod ime Abstract. Vsebuje mnogo manj elementov kot Swing, saj vsebuje le tiste, ki so skupni vsem platformam [5].

Za razliko od AWT je Swing napisan celotno v Javi in temelji na lahkih komponentah [6]. Ne zanaša se na okna gostujočega operacijskega sistema, temveč se vsi elementi izrišejo v Javi. Omogoča zamenljiv videz uporabniškega vmesnika (*Pluggable Look and Feel*). Odločimo se lahko za platformno odvisen izgled ali za izgled, ki bo na vseh platformah enak (Java izgled). Vse komponente Swing, kot so gumb (*JButton*), drsnik (*JScrollbar*), oznaka (*JLabel*) in druge, so napisane celotno v Javi in vsebujejo vse funkcionalnosti kot AWT in še druge. Poleg tega so v Swingu dodani bolj napredni elementi, kot so vsebovalnik večstranska površina (*JTabbedPane*), drsna površina (*JScrollPane*), drevo (*JTree*) in drugi.

Pri Swingu zamenljiv videz pomeni, da se lahko obnašanje in vizualni izgled grafičnega elementa spremeni, brez spreminjanja programa [5]. Ko aplikacija ustvari gumb, ta ve, kako se mora izrisati na ekran in kako se odzivati na prelet in operacije miške [5]. Toda te naloge so predane specializiranim razredom. Če teh nalog element ne bi predal in bi vseboval to kodo, bi jo bilo potrebno spremeniti ali pa napisati novo metodo za izris [5]. Swing pa omogoča namestitev in izbiro poljubnega videza.

Naša želja je bila uporabljati napredne komponente in zagotoviti platformno enak izgled programov (Java izgled). Zato smo se odločili za uporabo komponent Swing.

3.2 Funkcionalne zahteve

Funkcionalne zateve so bile, da naj bi bili elementi iz menija dostopni preko bljižnic (kombinacij tipk na tipkovnici). Hkrati naj bi bilo možno izvajati vse programe (npr. hkratno gledanje filma in urejanje besedil). Vsaka možnost, naj si bo gumb ali pa končna izbira iz menija, naj bi imela svoj namig (*Tooltip*), ki razloži, kaj ta izbira naredi. Posamezne napake, ki se pojavijo v programu, naj bi se izpisale v statusno vrstico, ali pa se bi jih ulovilo in ignoriralo. Programi naj bi odpirali le formate, katere znajo obdelovati in ne drugih.

Nefunkcionalne zahteve so delno opisane v prejšnjem podpoglavju (MDI aplikacija, robna poravnava in uporaba komponent Swing). Poleg tega bi uporabili koncept niti, da bi lahko uporabnik opravljal več stvari hkrati.

3.3 Knjižnjice in arhivi JAR

Pri zahtevnejših programih, ki predvajajo glasbo ali pridobivajo podatke iz spleta, ni smiselno pisati na najnižjem nivoju, kako se bo recimo glasba predvajala. Namesto tega se uporabljajo že obstoječe knjižnice. Knjižnica je množica napisanih razredov (če govorimo o Javi), ki znajo delati z določenimi stvarmi. Npr. knjižnica minim zna prebrati pesem z metodo loadFile in jo predvajati z metodo play. Kako so te metode dejansko implementirane, razvijalca ne zanima. Seveda bi lahko tudi sami vse napisali, toda za delo bi porabili veliko več časa in truda, poleg tega pa bi morali posamezno stvar popolnoma razumeti.

Razredi iz knjižnic so za Java programe zapakirani v tako imenovane arhive JAR (*Java ARchive*). Arhivi temeljijo na formatu ZIP in vsebujejo končnico .jar [7]. Orodje NetBeans omogoča dodajanje arhivov JAR v projekte. Pri projektu smo uporabili knjižnice, kot so iText, Apache POI in druge. Bolj podrobno so opisane v podpoglavjih.

Poglavje 4

Rezultat razvoja projekta

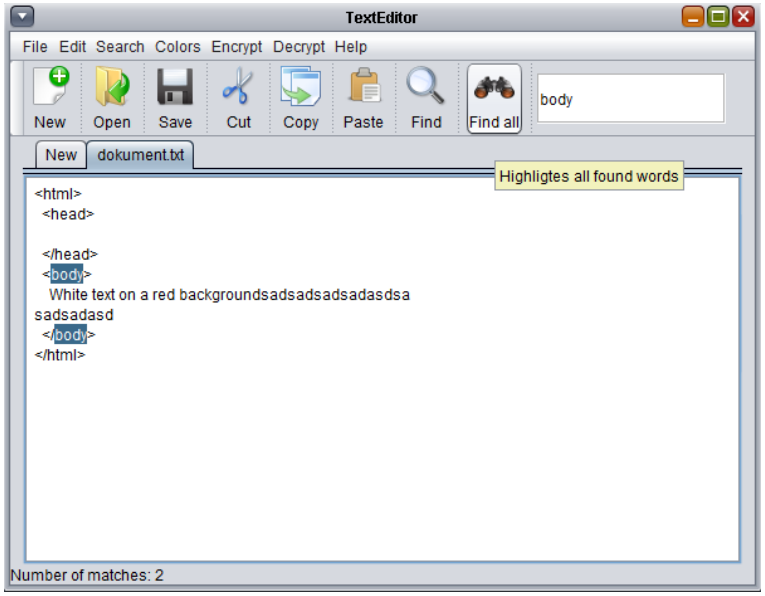
4.1 Urejevalnik besedil

Namen je bil ustvariti urejevalnik besedil, ki bi bil podoben beležnici (*Notepad++*) z nekaj svojevrstnimi značilnostmi.

Razvili smo urejevalnik besedil, ki omogoča odpiranje txt, doc, docx in pdf (*Portable Document Format*) formatov datotek. Iz teh datotek je mogoče pridobiti besedilo. Vsaka odprta datoteka se prikaže v svojem zavihku. Nad besedilom so omogočene operacije reži, kopiraj in prilepi. Po besedilu je možno iskati posamezne besede s klikom na ikono najdi (*find*). Metoda poišči vse (*FindAll*) najde vse pojavitve besede in jih obarva. Uporaba metode FindAll je prikazana na sliki 4.1, pri čemer sta odprti dve datoteki: New (nova prazna vsebina) in dokument.txt.

Za formata doc in docx Java ne vsebuje metod, s katerimi bi lahko iz njiju pridobili podatke. Dokument Word ni preprost format, ampak je oblikovan z množico XMLjev. Zato smo za obravnavanje teh formatov uporabili knjižnico Apache POI, napisano za Javo, ki omogoča manipuliranje z datotekami Word, Excel in PowerPoint [8]. Odpiranje datotek pdf smo realizirali s knjižnico iText, ki omogoča ustvarjanje in urejanje dokumentov pdf [9].

V program smo dodali kriptiranje. Kriptiranje je postopek, v katerem golo besedilo preko algoritma pretvorimo v takšno obliko, da je berljiva samo



Slika 4.1: Prikaz izgleda urejevalnika besedil. Za vsako odprto datoteko se ustvari nov zavihek.

tistim, ki poznajo posebno znanje (ponavadi ključ) [10]. Dekripcija je obraten postopek kot kriptiranje in kriptirano besedilo pretvori v originalno obliko.

Kriptirne algoritme delimo na simetrične, asimetrične in zgoščevalne funkcije. Vsak od njih ima svoje prednosti. Pri simetričnem kriptiranju se uporablja en ključ za enkripcijo in dekripcijo, zato morata zanj vedeti le pošiljatelj in prejemnik. Prednost simetričnih algoritmov je v hitrosti, slabost pa v načinu izmenjave ključa. Asimetrična kriptografija uporablja zasebni in javni ključ. Javni ključ je znan vsem, zasebni nikomur. Pošiljatelj mora najprej zakriptirati sporočilo z svojim zasebnim ključem in nato s prejemnikovim javnim ključem. Pošiljatelj sporočilo dekriptira preko svojega privatnega ključa in pošiljateljevega javnega ključa. Algoritmi so zahtevnejši in zato tudi počasnejši, ni pa problema pri izmenjavi ključev. Za zagotovitev integritete sporočila uporabljamo zgoščevalne funkcije. Na sporočilu se izračuna nekakšen povzetek, ki se zakriptira s pošiljateljivim zasebnim ključem. Pošiljatelj dekriptira naš povzetek s pošiljateljevim javnim ključem.

Slika 4.2: Prikaz Cezarjevega algoritma. Beseda france se z zamikom 4 črk kriptira v judsfi.

Na originalnem sporočilu se nato izračuna povzetek. Če sta oba povzetka enaka, se sporočilo med prenosom ni spremenilo.

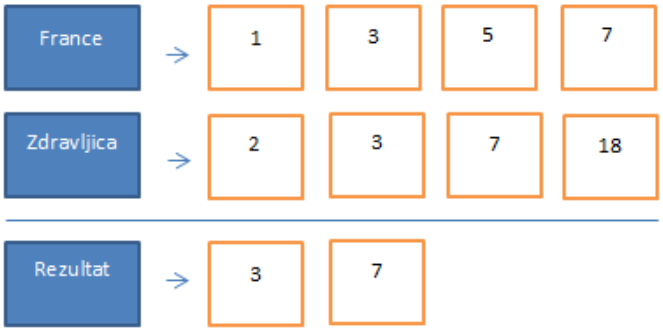
Cezarjev algoritem [11] je preprost algoritem, ki vsako črko iz besedila zamenja s črko, ki se v abecedi nahaja n črk za njo. Slika 4.2 prikazuje zamik na 4 črke naprej. Črka a se zamenja z d, b z e in tako dalje.

Polybiusov kvadrat [12] (*Polybius square*) je posotopek, ki črke razporedi v kvadrat velikosti n stolpcev in n vrstic. Črka v besedilu se zamenja s številkama vrstice in stolpca, kjer se črka nahaja. Polybiusovemu kvadratu smo v programu prilagodili velikost, saj smo dodali tudi slovenske in velike črke.

V programu lahko izvajamo več stvari hkrati. Lahko na primer čakamo, da se besedilo zakriptira in istem času pregledujemo neko drugo besedilo. Da je to mogoče, smo za kriptiranje uporabili koncept niti, ki omogočajo, da se lahko izvaja več delov kode "istočasno". Če niti v programu ne bi uporabili, bi morali pri dolgih besedilih čakati, da se kriptiranje izvede do konca. Šele nato bi lahko počeli kaj drugega.

Večnitnost aplikacij prinaša številne prednosti, kot so bojša izraba centralne procesne enote, boljša zanesljivost sistema in večja zmogljivost na večjedrnih procesorjih [13].

Kot napredno značilnost urejevalnika smo naredili odpiranje datotek na osnovi obrnjenega indeksa. Namen naprednega odpiranja datotek je, da



Slika 4.3: Delovanje obrnjenega indeksa. Rezultat je presek obeh seznamov dokumentov.

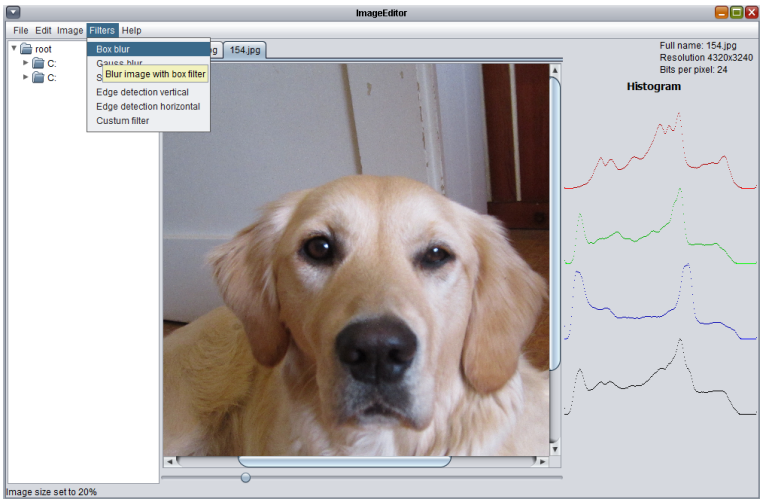
uporabniku ponudi datoteke, ki vsebujejo nek seznam besed. Uporabniku omogoča, da si izbere mapo z datotekami in po njej išče. Ob tem se zgradi obrnjen indeks (*Inverted index*).

Obrnjen indeks je način, ki omogoča hitro identifikacijo dokumentov, ki vsebujejo želene besede. Vsak dokument se razdeli na seznam besed. Za vsako besedo se ustvari seznam dokumentov, ki vsebujejo to besedo. Pri iskanju dokumentov, ki vsebujejo želene besede, se primerjajo seznami vseh besed in vrne rezultat glede na želeno operacijo. Slika 4.3 prikazuje njegovo delovanje. Pri iskanju dokumenta, ki vsebuje besedi France in Zdravljica, se išče presek med obema seznamoma besed.

4.2 Urejevalnik slik

Na spletu nismo našli knjižnic, ki bi znale upravljati z slikami. Ena izmed možnosti bi bila uporaba Processinga, a smo se odločilli, da bomo izdelali preko Javinega razreda Image in BufferedImage. Program omogoča odpiranje direktorija slik, zapiranje in shranjevanje slik, razveljavitev akcij (*undo*), obračanje slik, obrezovanje slik, izris histograma slik in izvajanje filtrov na slikah.

Za izrisovanje slik na komponenti vsebovalnik plošča je bilo potrebno v



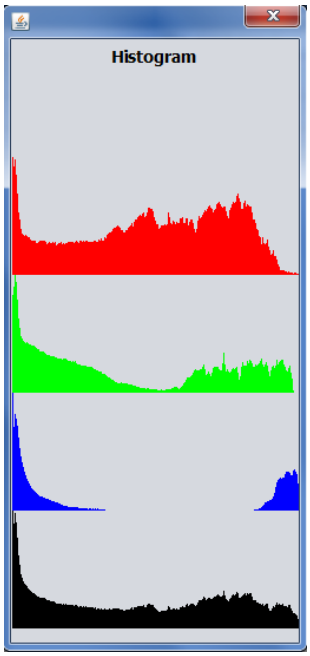
Slika 4.4: Primer izgleda urejevalnika slik. Levo je prostor za drevo, desno za podatke in histogram, v središču je slika, na dnu pa statusna vrstica.

vsebovalniku plošči razširiti zmožnosti in ponovno definirati metodo paint, tako da v ozadje izrisuje sliko. Razred BufferedImage omogoča ustvarjanje in urejanje posameznih slik v spominu. Zna dostopiti do posameznih pikslov slike, kar nam omogoča izvajanje filtrov nad sliko. Slika 4.4 prikazuje izris slike v vsebovalniku plošči. Ta je definirana z drsniki, ko slika preseže notranjo velikost prostora.

Pri pregledovanju direktorija smo realizirali možnost animiranega ogleda (*slideshow*). Nastaviti je možno čas trajanja in pa velikost prikaza slike.

Histogram je zelo uporabna stvar pri pregledovanju in urejanju slik. Primer histograma prikazuje slika 4.5. Prikaže porazdelitve barv na slikah in preko njega izvemo, kakšen je kontrast v sliki. Ob vsaki spremembi slike, kot je npr. dodajanje filtra za glajenje, se ponovno izračuna in izriše. V programu smo vsaki sliki izračunali in izrisali histogram za vsak barvni kanal (moder, rdeč, zelen in skupen histogram). Podatke smo dobili iz pikslov (najmanjši elementi slike, ki vsebujejo informacijo o barvi).

Program ima možnost obrezovanja slike. Beležijo se koordinate pritiska in sprostitve miškinega kazalca. V sliki se označi pravokotnik. V izrisu se



Slika 4.5: Primer histograma. Prikazuje, katere barve pikslov vsebuje slika.

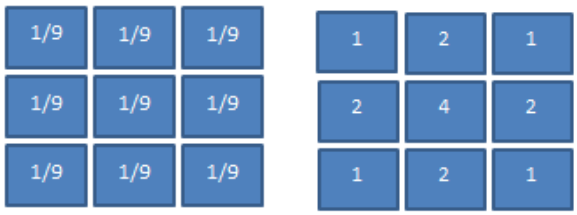
uporabijo le piksli slike znotraj pravokotnika.

Obračanje slik smo realizirali preko afinih transformacij. Afine transformacije ohranjajo kolinearnost. Razmerja dolžin se ohranjajo. Java v ta namen ponuja razred AffineTransform. Te dosežemo preko rotacij, skalacij, premika, zrcaljenja in striženja [14].

Sliko povečamo in pomanjšamo tako, da jo množimo s skalirnim faktorjem. Ta se spremeni z vsako spremembo drsnika v programu.

Na posamezni sliki je mogoče razveljaviti narejene akcije. Za razumevanje delovanja je potrebno poznati podatkovno strukturo Vector. Vector je struktura, ki omogoča hranjenje več objektov nekega tipa (v našem primeru slika - BufferedImage). Vsakič, ko se zgodi neka sprememba na sliki, se objekt spremenjene slike doda v ta Vector. Razveljavitev akcije je nato trivialna. Za razveljavitev akcije se odstrani zadnji element v Vectorju in kot aktualni prikaže predzadnji.

Filter je efekt, ki spremeni piksel glede na ostale piksle v okolici. Ponavadi



Slika 4.6: Primer glajenja s tekočim povprečjem (levo) in Gaussovega filtra (desno). Z upoštevanjem, da piksli v bližini vplivajo bolj, vrne Gaussov filter malo lepše rezultate.

ga predstavimo kot matriko velikosti n stolpcev in n vrstic. Filter premikamo po sliki in množimo vsak piksel z vrednostjo v oknu. Na koncu vrednosti seštejemo in nova vrednost se vpiše v piksel. Filtri se uporabljajo za detekcijo robov, ostrenje, glajenje, odstranjevanje šuma, spreminjanje velikosti slike in drugo.

V programu smo realizirali dva filtra za glajenje in detekcijo robov in enega za ostrenje. Slika 4.6 prikazuje dva filtra za glajenje. Uporabljena je matrika velikosti treh stolpcev in vrstic. Dodana je bila možnost poljubnega filtra.

Pri glajenju s tekočim povprečjem (*Box filter*) vsak piksel dobi vrednost povprečja vseh pikslov v okolici [15]. Tak filter iz slike odstrani iz slike hitre prehode in sliko zgladi. Toda filter prepušča tudi visoke frekvence, kar se v sliki zazna kot nekakšne horizontalne in vertikalne črte. Njegova izboljšava je tako imenovani Gaussov filter [15], ki upošteva, da več prispevajo tisti piksli, ki so bližje. Posledično ne prepušča visokih frekvenc in rezultat je malo boljša slika kot pri glajenju s tekočim povprečjem.

Ostrenje poudarja robove in zmanjša zglajenost slike. To doseže z odvodi. Prvi odvod pove naraščanje ali padanje, drugi določa ukrivljenost. Ostrenje se doseže tudi z odštevanjem zglajene slike od originalne slike.

Pri detekciji robov želimo poudariti piksle levo in desno od danega piksla (horizontalni), ali pa piksle zgoraj in spodaj danega piksla (vertikalni).

Sliko je s programom moč shraniti v različne formate. PNG, JPG, BMP, GIF in WBMP so slikovni formati, za katere ima razred ImageIO vtičnike, s katerimi jih obdeluje. Format shranjevanja se lahko določi bodisi s klikom v pogovornem oknu bodisi z vpisom celotnega imena datoteke z končnico.

4.3 Predvajalnik glasbe

Pri gradnji predvajalnika glasbe smo uporabili tako Javo, kot tudi Processing. Processing ponuja glasbeno knjižnico minim. Minim je glasbena knjižnica, ki uporablja vmesnik JavaSound API, MP3SPI in delno Tritonus, s katerimi zgradi enostavno glasbeno knjižnico za uporabo, namenjeno razvijalcem v okolju Processing [16]. Glavna ideja minim je, da omogoča preprost način vključevanja zvoka v projekte in hkrati nudi zadostno fleksibilnost za bolj zahtevne uporabnike [16]. Za razliko od Jave se ni potrebno neposredno ukvarjati s seznami in ostalimi nižjimi strukturami. S tem se prihrani tako imenovano umazano delo. Poleg tega pa paket Processinga vsebuje veliko število primerov uporabe in dokumentacijo.

Trenutno knjižnica minim podpira predvajanje zvočnih formatov WAV, AIFF, AU, SND in MP3. V projekt smo vključili še branje metapodatkov iz značk ID3 (*ID3 tag*) in izris hitre Fourierjeve transformacije zvoka (*FFT*).

Namesto knjižnice minim bi lahko uporabili tudi knjižnico Beads, SoundCipher in druge. Za minim smo se odločili, ker smo se s knjižnico že srečali, vsebuje zelo veliko primerov, od predvajanja glabe pa vse do vizualizacije podatkov. Vse to pa je moč narediti z zelo preprostimi ukazi in malo kode. Kot slabosti minima smo odkrili predvsem počasnost pri preskakovanju delov pesmi. Minim uporablja metodo cue(int millis), ki začne predvajati pesem od želene sekunde dalje. Velikokrat traja nekaj sekund, da se predvajanje dejansko prestavi.

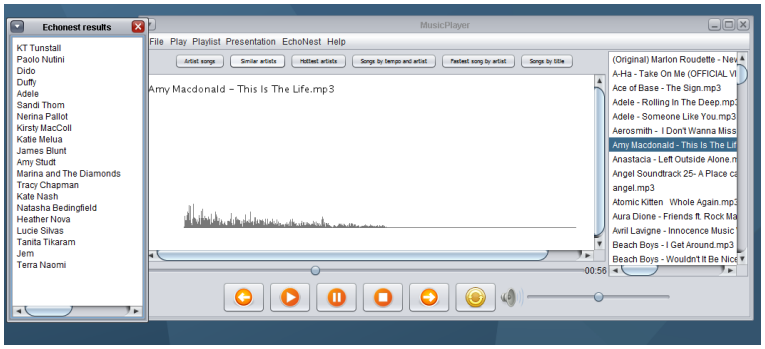
Kot napredno značilnost smo dodali uporabo storitev podjetja EchoNest. Echo Nest je glasbeno podjetje, ki poganja pametne glasbene aplikacije za širok nabor strank, kot so MTV, BBC ter okoli 7000 neodvisnih razvijal-

cev [17]. Podjetje vsebuje velik repozitorij glasbenih podatkov, ki so razvijalcem dostopni preko APIja. Preko njega je možno pridobiti podatke o avtorjih (novice, blogi, slike ...), skladbah (tempo, glasnost, predogledi ...) in uporabiti storitve, kot je npr. generiranje seznamov predvajanj. Za uporabo APIja je potreben ključ, ki ga razvijalec dobi ob registraciji na uradni strani. API je dejansko arhiv JAR, ki vsebuje razrede, ki pošiljajo podatke.

V splošnem s storitvijo EchoNest delamo naslednje. Razrede, ki so vsebovani v arhivu JAR (Artist, Song ...), uporabimo, da zgradimo poizvedbo, ki jo nato preko zahteve HTTP POST pošljemo na strežnik. Strežnik obdela zahtevo in vrne rezultat v formatu JSON ali XML, kakor smo določili v poizvedbi. Rezultatov v obliki XML ali JSON nam ni potrebno razčlenjevati. EchoNest ima zato že narejene razrede. Tako nam metoda searchArtists(Params p) zgradi poizvedbo HTTP POST z danimi parametri (recimo nastavimo v parametrih za ime na avr), jo pošlje na strežnik in od strežnika dobi odgovor. Ta odgovor razčleni tako, da nam v podatkovno strukturo seznam napolni podatke o avtorjih. Z množico razredov lahko te strukturirane podatke enostavno pridobimo. Omejitev pri uporabi storitve je, da mora uporabnik imeti svoj razvijalski ključ. Druga pa je omejitev števila poslanih zahtev na minuto za ta ključ.Ta številka se spreminja odvisno od trenutne obremenjenosti storitve. S tem se izognejo morebitnim izpadom. Navadno pa je za nekomercialne uporabnike omejitev okoli 120 klicev na minuto. Ta podatek se izpiše v glavi HTTP odgovora pod polji X-RateLimit-Limit (pove omejitev klicev), X-RateLimit-Remaining (kolikokrat še lahko kličemo), X-RateLimit-Used (koliko klicev je bilo opravljenih). Slika 4.7 prikazuje rezultat poizvedbe glede na dan URL (*Uniform Resource Locator*). Vrnjen rezultat je v obliki JSON.

V programu smo omogočili uporabo storitve EchoNest pri predvajanju pesmi. Za trenutno predvajano pesem je v realnem času možno poiskati: vse pesmi avtorja predvajane pesmi, iskanje podobnih avtorjev trenutnega avtorja, pesmi s hitrim tempom tega avtorja, najhitrejšo pesem avtorja, pesmi z enakim naslovom in trenutno najbolj želene avtorje. Za bolj širši namen

Slika 4.7: Primer poizvedbe. Iščemo avtorje, katerih ime vsebuje besedo avr.



Slika 4.8: Primer izgleda glasbenega predvajalnika. Prikazana je orodna vrstica za storitev EchoNest in rezultat, izpisan v svojem oknu.

uporabe smo ustvarili novo okno. V njem lahko uporabnik vpiše avtorja in pesem, katera želi poiskati. Tako ni vezan le na trenutno pesem in izvajalca, ampak lahko išče poljubno. Ob vseh zgoraj naštetih podatkih se izriše tudi slika avtorja, če je ta ustrezne velikosti. Slika 4.8 prikazuje uporabo storitve EchoNest. Med predvajanjem pesmi iščemo podobne izvajalce, kot je npr. Amy McDonald. Rezultat se izpiše v svojem oknu.

Pri predvajanju pesmi so omogočeni trije tipi vizualnega prikaza. Prvi način je uporaba hitre Fourierjeve transformacije. Drugi na podoben način izrisuje na površino barvne kvadrate. Zadnji način pa v povšino izrisuje metapodatke, če se ti nahajajo v značkah ID3.

Predvajalnik zna odpreti datoteko ali mapo z datotekami. Te datoteke se prenesejo v seznam predvajanja (*playlist*). Seznam predvajanja omogoča dodajanje in odstranjevanje elementov. Možno ga je shraniti v datoteko, pri čemer se vanjo zapišejo absolutne poti do pesmi iz tega seznama. V kolikor uporabnik premika pesmi po različnih mapah, program ni več zmožen predvajati seznama, saj nekateri zapisi kažejo na neobstoječe datoteke. Če z datoteko in pesmimi ni nič narobe, je seznam v programu možno odpreti.

4.4 Predvajalnik videa

Glavna cilja pri gradnji predvajalnika videa sta bila, da bo zmožen predvajati veliko formatov in bo omogočal prenos ter predvajanje podnapisov.

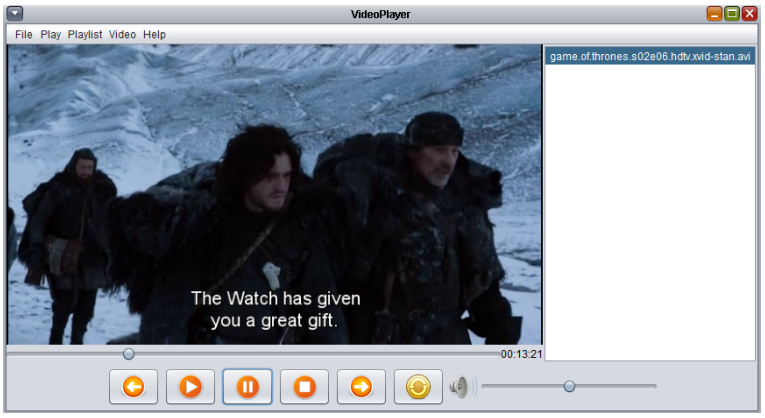
Predvajalnik videa smo poskusili implementirati s pomočjo knjižnice JMF (*Java Media Framework*). Takoj smo ugotovili, da knjižnica ne bo primerna. Na težave smo naleteli že pri predvajanju osnovnih video formatov. Ne podpira kodiranj MPEG-2 in MPEG-4, formatov WMV, RM, QuickTimovovih filmov in Flasha novejšega od verzije 2 [19]. To bi se dalo odpraviti z namestitvijo projekta FFmpeg, ki poda knjižnice za delo z zvokom in videom. Po prenosu teh knjižnic je potrebno hkrati urediti nastavitve in povezave z JMF. Na koncu smo prišli do ugotovitve, da uporaba JMF ni primerna, saj je potrebno ogromno časa za njegovo vzpostavitev, na spletu ni veliko

primerov, s katerimi bi si lahko začetnik pomagal in knjižnjica je zelo slabo vzdrževana (API ni bil nadgrajen vse od leta 1999 [19]).

Pri dodatnem iskanju smo naleteli na odprtokodni projekt VLCJ (*Java Framework for VLC*). VLCJ je projekt, ki prinaša povezave med VLC predvajalnikom in programskim jezikom Java. Omogoča vključitev primerka predvajalnika VLC v Javino glavno okno (JFrame za Swing, Window za AWT). Poleg povezave, je razvijalcu omogočen visokonivojski API, ki skrije zapletenost upravljanja z libvlc. Je močan projekt, ki omogoča vse od gradnje aplikacij v Javi za enostavno predvajanje video vsebin do vzpostavitve strežnika za video na zahtevo. Omogoča predvajanje vseh video formatov, ki jih VLC podpira. Za razliko od JMF tu ni potrebno konfigurirati povezav. Ključni za delovanje predvajalnika sta knjižnici libvlc in libvlccore in mapa plugins. Pridobimo jih pri predvajalniku VLC v mapi lib. Preko teh knjižnic je uporabniku omogočen dostop do lastnosti predvajalnika VLC. Poleg vsega omogoča tudi predvajanje podnapisov, kar je eden od razlogov, da smo se odločili zanj. Tako kot pri knjižnici minim (predvajalnik glasbe) tudi tu dostopamo do API na visokem nivoju, kar pomeni, da se ni potrebno ubadati z nizkonivojskimi problemi.

Kot alternative za predvajanje video vsebin smo na spletu našli še knjižnice QuickTime for Java in Xuggler, a smo se zaradi dobre izkušnje z VLCJ odločili za slednjo.

Slika 4.9 prikazuje splošen izgled predvajalnika in hkrati prikazuje uporabo podnapisov. Pri predvajanju tujih filmov ali televizijskih oddaj si velikokrat zaželimo imeti podnapise. Takrat je potrebno poiskati podnapise na spletni strani. To nam je velikokrat odveč, zato se je pojavila ideja, da bi se nam za predvajano vsebino avtomatično poiskali podnapisi na spletu. S tem nam ne bi bilo potrebno samim brskati po spletnih straneh za njimi. To funkcionalnost vsebuje priljubljen predvajalnik BSplayer. Na njegovi podlagi smo se odločili implementirati način avtomatičnega pridobivanja podnapisov. Uporabili smo stran www.podnapisi.net in storitev OpenSubtitles, kar je podrobno opisano v kasnejših odstavkih.



Slika 4.9: Primer izgleda predvajalnika videa. V filmu so omogočeni podnapisi, ki se jih da pridobiti preko storitev OpenSubtitles in Podnapisi.net

Včasih želimo izvedeti kaj več o trenutno predvajani vsebini, npr. kdo so igralci, kakšen je naslov, žanr, opis in drugo. Stran www.imdb.com vsebuje ogromno podatkov o filmih, igralcih in vsem, kar spada zraven. Na žalost ni uradnih aplikacijskih vmesnikov, s katerimi bi lahko razvijalci pridobivali podatke. V programu smo implementirali svoj vmesnik za dostop do podatkov. Več je opisano v kasnejših odstavkih.

Aplikacijski programski vmesnik smo uporabili, da naš program omogoča dostop do spletnih strani in storitev.

OpenSubtitles.org je spletna stran, ki ponuja kopico podnapisov za serije in filme. Podnapise in ostale podatke poleg tega nudijo preko storitev. Celotna stvar deluje podobno kot storitev EchoNest. Storitev OpenSubtitles uporablja protokol XML-RPC (*XML Remote Procedure Call*). XML-RPC sporočilo je zahteva HTTP-POST, ki jo zgradimo s pomočjo razredov, ki so dostopni razvijalcem. V telesu sporočila se posreduje struktura XML. Ta se izvrši na strežniku in dobljen odgovor se prav tako posreduje v formatu XML. Slika 4.10 prikazuje izgled strukture. Poleg glave, v kateri izvemo, da gre za zahtevo HTTP tipa XML in dolžine 187 znakov, se v telesu podaja struktura XML. V našem primeru smo klicali metodo example.getFacultyName s parameterom FRI.

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: jan.something.com
Content-Type: text/xml
Content-length: 187


<?xml version="1.0"?>
<methodCall>
   <methodName>example.getFacultyName</methodName>
   <params>
          <param>
                 <value><string>FRI</string></value>
          </param>
         </params>
</methodCall>
```

Slika 4.10: Primer izgleda strukture XML-RPC sporočila.

V projekt je vključenih 8 razredov, ki sestavljajo vmesnik OpenSubtitles. Iz njega lahko pridobimo podatke o podnapisih. Primeri nekaj podatkov: format podnapisov, povezava do prenosa podnapisa, zapakiranega v formatu ZIP, ime podnapisa, ocena serije ali filma na imdb, povezava do podnapisa, identifikator filma ali serije na imdb, ki se pripiše v URL in drugo.

Po pridobitvi zgoraj naštetih podatkov je potrebno prenesti arhiv zip, ki vsebuje podnapis. Za prenos podnapisa, shranjenega v arhivu zip, smo razvili svoj razred. Ta prenese arhiv iz strežnika na računalnik, nato ga razširi v isto mapo kot predvajana vsebina in ga preimenuje v isto ime kot predvajana vsebina. To omogoča, da se podnapisi avtomatično naložijo ob začetku predvajanja vsebine.

Poleg storitve OpenSubtitles smo želeli pridobivati podnapise iz strani www.podnapisi.net. Prednost našega APIja je, da najde več rezultatov (stran www.podnapisi.net vsebuje več podnapisov), a njegova slabost je počasnost. Storitev ne deluje na podlagi protokola XML-RPC, ampak na pridobivanju podatkov iz izvorne kode spletne strani. Slabost je predvsem v tem, da če se v prihodnosti spremeni slog strani ali pa preimenuje atribute (Npr. razred v CSS), API ne bo več deloval pravilno. Po drugi strani je API enostaven. Podamo le ime želene vsebine oz. kar absolutno pot do datoteke. Iz imena se izločijo naslov vsebine in po možnosti tudi del (če gre za serijo). Na

podlagi teh podatkov se program poveže na ustrezen URL in shrani izvorno kodo spletne strani v niz. Vse preostalo pa je dejansko iskanje za podatki v tem nizu. Te podatki se nahajajo znotraj enoličnih HTML značk. Rezultat APIja vrne podatke o verziji filma/serije (release), jeziku podnapisov in pot URL do datoteke zip na strežniku. Te podatke vsebuje vsak podnapis, vsi skupaj pa se uporabniku vrnejo v seznamu nizov. Počasnost izvira iz tega, da je potrebno za vsak podnapis odpreti svojo stran in jo v celoti prebrati. Hitrost je torej odvisna od količine podnapisov, ki so na voljo za posamezen film oz. epizodo.

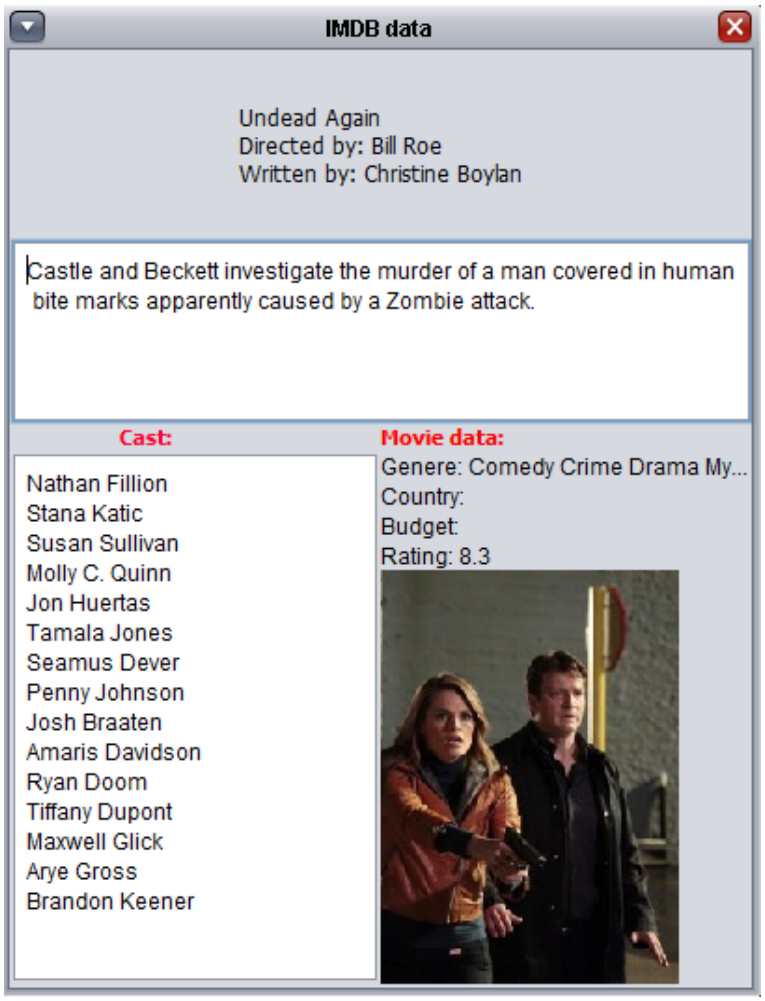
Postopek komunikacije za OpenSubtitles poteka sledeče. Najprej se program prijavi na storitev OpenSubtitles. Nato poda APIju konkretno datoteko. API se poveže na strežnik in prenese zgoraj naštete podatke. Te podatke izpiše v splošno pojavno okno, kjer jih uporabnik lahko izbere.

Postopek komunikacije pri Podnapisi.net je malo drugačen. Razložili ga bomo za film Thor. Najprej se iz konkretne datoteke izloči, ali gre za film ali serijo. Pri seriji je namreč v URL potrebno dodati dodatne parametre. Potem se program poveže na spletno stran. Prikažejo se rezultati iskanja. Za vsak rezultat iskanja se shrani povezava na podstran, ki vsebuje dejanske podatke. Vsaka podstran se shrani v dolg niz. Iz tega niza program pridobi podatke o verziji filma/serije, jeziku podnapisov in poti URL do datoteke zip na strežniku ter jih vrne uporabniku v seznamu nizov.

Po uporabi katere koli od teh dveh storitev, se uporabniku pokaže okno s stikali (*JCheckBox*). Vsak podnapis ima svoje stikalo. Uporabnik nato obkljuka želen podnapis. Če uporabnik izbere več podnapisov, se upošteva in prenese zadnji podnapis, ki se nato avtomatično vklopi v predvajanje.

Na podoben način smo realizirali tudi storitev IMDB. Spletna stran www.imdb.com vsebuje podatke o filmih, direktorjih, igralcih, opisih in podobno. V programu smo omogočili, da si uporabnik ogleda informacije o trenutno predvajani vsebini. IMDB API za delovanje enako kot storitev Podnapisi.net potrebuje naslov datoteke, ki jo želimo predvajati.

Tu smo uporabili storitev OpenSubtitles, ki kot eno izmed polj rezul-



Slika 4.11: Izgled vmesnika za prikaz podatkov iz spletne strani www.imdb.com. Vmesnik dobi podatke od naslova, zgodbe, igralcev pa vse do slik.

tata vrne identifikator filma ali serije na imdb. Identifikator filma ali serije, se je izkazal kot zelo uporabnega, saj avtomatično vrne identifikacisko številko filma ali epizode serije. Želimo na primer gledati serijo z imenom Castle in sicer četrto sezono in osemnajsti del. Te podatke posredujemo storitvi OpenSubtitles, ta pa nam za identifikator vrne število 2231927. To števiko dodamo k obstoječemu URL www.imdb.com/title/tt. Rezultat je www.imdb.com/title/tt2231927 in kaže na podatke o delu serije ali pa filmu. Nato smo enako kot pri strani Podnapisi.net iskali podatke znotraj značk HTML. Za razliko od Podnapisi.net smo za hranjenje podatkov uporabili isti razred (pri Podnapisi.net se podatki ne shranijo, temveč vrnejo). Razred omogoča pridobitev informacij o naslovu, oceni in opisu zgodbe predvajane vsebine, o direktorjih, piscih in igralcih predvajane vsebine ter osnovnih podatkih, kot so država snemanja, slika, proračun za snemanje in žanr. Pridobljeno vsebino smo v programu izpisali v ločenem pojavnem oknu. Uporabniku se izpiše sporočilo, če program podatkov o video vsebini ne najde. Slika 4.11 prikazuje izgled vmesnika za zgoraj navedeni primer.

V grobem primer komunikacije poteka sledeče. S storitvijo OpenSubtitles program pridobi identifikacijsko številko filma/serije na www.imdb.com. Program doda številko k URL, da je v sledeči obliki www.imdb.com/title/tt2231927. Nato se poveže na stran in prebere HTML kodo v niz. Iz niza pridobi podatke o filmu, igralcih in drugem. Vmesnik je hitrejši od Podnapisi.net, saj se poveže le eno stran, za razliko od Podnapisi.net, kjer je število povezav odvisno od števila podnapisov.

4.5 Morebitne težave in napake pri uporabi APIjev

Do napak pri uporabi spletnih storitev lahko pride iz več razlogov. Spletna storitev mogoče ni na razpolago. Npr. storitev EchoNest zaradi preobremenjenosti z zahtevami ne deluje, ali pa uporabnik nima vključene internetne povezave.

EchoNest zna v določenih primerih vrniti napačne rezultate. Zasnovali smo ga namreč tako, da podatke (avtor in naslov pesmi) najprej pridobi iz ID3 značk v datoteki. Če značke ne vsebujejo zapisov, se ime avtorja in naslov pesmi prebere iz naslova datoteke. Tukaj pa lahko pride do napak, saj smo program zasnovali tako, da predpostavi, da je naslov sestavljen iz imena avtorja in naslova pesmi, ločenih z vezajem. Npr. za pesem Granade Bruna Marsa, program predpostavlja, da je naslov v obliki "Bruno Mars - Grenade.mp3". Zapis v kakšni drugi obliki ne bi bil ustrezen.

Pri predvajalniku videa lahko pride do pomanjkljivosti, ko storitev ne more identificirati nobenega filma ali serije. Posledično ne deluje tudi API IMDB, ki sloni na storitvi OpenSubtitles, saj preko njega dobi identifikacijsko številko strani, iz katere pridobiva podatke.

Poglavje 5

Zaključek

Izdelek ima še veliko prostora za nadgradnjo, vendar smo ocenili, da je dovolj praktičen in zanesljiv za uporabo. Svetovni splet je čezdalje bolj pomemben in dostopen, zato menimo, da bi aplikacije morale izkoriščate njegove storitve. Z izbiro programskega jezika smo uporabo izdelka omogočili večjemu občinstvu. Pri gradnji smo programe dopolnili z edinstvenimi lastnostmi, kot je npr. uporaba obrnjenega indeksa v urejevalniku besedil ali pa prikaz podatkov o trenutno predvajani vsebini (IMDB). S projektom VLCJ smo za programski jezik Java odkrili preprost način za upravljanje z video vsebinami na višjem nivoju. Program bi sicer lahko izboljšali na mnogih področjih s pohitritvijo programa in dodajanjem novih lastnosti.

V celoti ocenjujemo, da smo uresničili cilje, ki smo si jih zadali. Menimo, da bo izdelek uporaben uporabnikom, ki si v enem paketu želijo upravljati z besedilom, sliko, zvokom in video vsebinami.

Literatura

[1] NetBeans IDE 7.1.2 Release Information. Dostopno na:
http://netbeans.org/community/releases/71/index.html.

[2] Java. Dostopno na:
http://en.wikipedia.org/wiki/Java_(programming_language).

[3] Java comparison. Dostopno na:
http://wiki.processing.org/w/Java_Comparison.

[4] Firebug. Dostopno na:
http://getfirebug.com/.

[5] Thomas Künneth (2004) Using Swing's Pluggable Look and Feel. Dostopno na:
http://today.java.net/pub/a/today/2004/02/27/laf.html.

[6] Josh Fletcher AWT vs Swing. Dostopno na:
http://edn.embarcadero.com/article/26970.

[7] JAR. Dostopno na:
http://en.wikipedia.org/wiki/JAR_(file_format).

[8] Apache POI - the Java API for Microsoft Documents. Dostopno na:
http://poi.apache.org/.

[9] iText. Dostopno na:
http://itextpdf.com/.

[10] Encryptio. Dostopno na:
http://en.wikipedia.org/wiki/Encryption.

[11] Caesar cipher. Dostopno na:
http://en.wikipedia.org/wiki/Caesar_cipher.

[12] Polybius square. Dostopno na:
http://en.wikipedia.org/wiki/Polybius_square.

[13] National Instruments (2011) Differences Between Multithreading and Multitasking for Programmers. Dostopno na:
http://www.ni.com/white-paper/6424/en.

[14] AffineTransform. Dostopno na:
http://docs.oracle.com/javase/1.4.2/docs/api/java/awt/geom/AffineTransform.html.

[15] Derek Hoiem (2011) Pixels and Image Filtering. Dostopno na:
http://www.cs.illinois.edu/class/sp11/cs543/lectures/Lecture

[16] Minim. Dostopno na:
http://code.compartmental.net/tools/minim/.

[17] EchoNest about us. Dostopno na:
http://the.echonest.com/company/.

[18] HTML. Dostopno na:
http://sl.wikipedia.org/wiki/HTML.

[19] Java Media Framework. Dostopno na:
http://en.wikipedia.org/wiki/Java_Media_Framework

[20] Application programming interface. Dostopno na:
http://en.wikipedia.org/wiki/Application_programming_interface.

[21] Mark Roulo (1997) Java's three types of portability. Dostopno na:
http://www.javaworld.com/javaworld/jw-05-1997/jw-05-portability.html.