



Št. naloge: 01667/2010

Datum: 05.04.2010

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

Kandidat: **VALENTIN ILJAŽ**

Naslov: **SISTEM ZA DISTRIBUCIJO VIDEA V SPLETNE STRANI**
SYSTEM FOR VIDEO DISTRIBUTION INTO WEB PAGES

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Diplomsko delo je posvečeno razvojnemu ciklu spletne aplikacije, ki omogoča uporabnikom vključevanje transparentnih videov v lastne spletne strani. Vključitev videa naj bo izvedena tako, da se video predvaja nad celotno vsebino gostujoče spletne strani. S tem naj bi bila izboljšana prepoznavnost in povečana odzivnost obiskovalcev gostujoče strani. Pri razvoju aplikacije uporabite razvojno ogrodje CakePHP, ki temelji na filozofiji hitrega in učinkovitega razvoja spletnih aplikacij. Za predvajanje transparentnega videa uporabite tehnologijo Adobe FLASH.

Mentor:

prof. dr. Saša Divjak



Dekan:

prof. dr. Franc Solina

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Valentin Iljaž

SISTEM ZA DISTRIBUCIJO VIDEA V SPLETNE STRANI

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

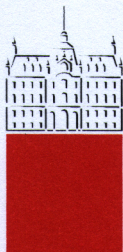
Mentor: prof. dr. Saša Divjak

Ljubljana, 2010

Univerza
v Ljubljani

Fakulteta *za računalništvo
in informatiko*

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si



Št. naloge: 01667/2010

Datum: 05.04.2010

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

Kandidat: **VALENTIN ILJAŽ**

Naslov: **SISTEM ZA DISTRIBUCIJO VIDEO V SPLETNE STRANI
SYSTEM FOR VIDEO DISTRIBUTION INTO WEB PAGES**

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Diplomsko delo je posvečeno razvojnemu ciklu spletne aplikacije, ki omogoča uporabnikom vključevanje transparentnih videov v lastne spletne strani. Vključitev videa naj bo izvedena tako, da se video predvaja nad celotno vsebino gostujoče spletne strani. S tem naj bi bila izboljšana prepoznavnost in povečana odzivnost obiskovalcev gostujoče strani.

Pri razvoju aplikacije uporabite razvojno ogrodje CakePHP, ki temelji na filozofiji hitrega in učinkovitega razvoja spletnih aplikacij. Za predvajanje transparentnega videa uporabite tehnologijo Adobe FLASH.

Mentor:

prof. dr. Saša Divjak



Dekan:

prof. dr. Franc Solina

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani **Valentin Iljaž**

z vpisno številko **63030127**

sem avtor diplomskega dela z naslovom:

Sistem za distribucijo videa v spletne strani.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom **prof. dr. Saša Divjak**
- 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 28.6.2010

Podpis avtorja:

Kazalo

Seznam uporabljenih kratic.....	14
Povzetek.....	15
Abstract.....	16
1 Uvod.....	17
2 Sistemi za distribucijo videov v spletne strani.....	18
3 Opredelitev problema.....	19
3.1 Vsebinske zahteve.....	19
3.2 Tehnološke zahteve.....	23
4 Tehnološka platforma.....	24
4.1 Strežniške tehnologije.....	24
4.1.1 Linux.....	24
4.1.2 Apache.....	24
4.1.3 MySQL.....	26
4.1.4 PHP.....	26
4.1.4.1 CakePHP.....	27
4.1.4.1.1 Hiter razvoj aplikacij.....	28
4.1.4.1.2 Model-Pogled-Kontroler.....	28
4.1.4.1.3 Konvencije.....	30
4.2 Tehnologije na strani klienta.....	32
4.2.1 HTML.....	32
4.2.2 CSS.....	34
4.2.3 JavaScript.....	35
4.2.4 Adobe FLASH.....	37
4.2.4.1 Actionscript.....	38
4.2.4.2 Flash video.....	39
4.2.4.3 Flash Player.....	40
5 Načrtovanje sistema.....	42
5.1 Programska arhitektura.....	43
5.2 Aktivnosti.....	43
5.3 Modeli primerov uporabe.....	48
5.3.1 Pregled videov v kategoriji.....	49
5.3.2 Pregled nazadnje dodanih videov.....	50
5.3.3 Pregled vsebinskih strani.....	51
5.3.4 Pregled posameznega videa (obiskovalci).....	51
5.3.5 Registracija.....	52
5.3.6 Pregled posameznega videa (uporabniki).....	54
5.3.7 Prijava.....	54
5.3.8 Izbira videa.....	55
5.3.9 Pregled izbranih videov.....	56
5.3.10 Pregled izbranega videa.....	57
5.3.11 Urejanje izbranega videa.....	58
5.3.12 Brisanje izbranega videa.....	59
5.3.13 Pregled statistike izbranega videa.....	59

5.3.14	Predogled izbranega videa.....	61
5.3.15	Pregled seznama lokacij.....	62
5.3.16	Dodajanje lokacije.....	63
5.3.17	Pregled posamezne lokacije.....	63
5.3.18	Brisanje lokacije.....	64
5.3.19	Predogled izbranih videov za lokacijo.....	65
5.3.20	Vključitev izbranega videa v spletno stran.....	66
5.3.21	Vključitev izbranih videov za lokacijo v spletno stran.....	67
5.3.22	Predvajanje videov.....	67
5.3.23	Prijava (urednik).....	68
5.3.24	Odjava (urednik).....	69
5.3.25	Pregled seznama kategorij.....	70
5.3.26	Dodajanje kategorije.....	71
5.3.27	Urejanje kategorije.....	71
5.3.28	Brisanje kategorije.....	72
5.3.29	Pregled posamezne kategorije.....	73
5.3.30	Pregled seznama videov.....	74
5.3.31	Dodajanje videa.....	75
5.3.32	Urejanje videa.....	76
5.3.33	Brisanje videa.....	77
5.3.34	Pregled posameznega videa.....	78
5.3.35	Ustvarjanje sličice za video.....	79
5.3.36	Pregled seznama vsebinskih strani.....	80
5.3.37	Dodajanje vsebinske strani.....	81
5.3.38	Urejanje vsebinske strani.....	82
5.3.39	Brisanje vsebinske strani.....	82
5.3.40	Pregled posamezne vsebinske strani.....	83
5.3.41	Nastavi kot domačo stran.....	84
5.3.42	Pregled seznama uporabniških skupin.....	84
5.3.43	Urejanje uporabniške skupine.....	85
5.3.44	Pregled posamezne uporabniške skupine.....	86
5.3.45	Pregled seznama uporabnikov.....	87
5.3.46	Dodajanje uporabnika.....	88
5.3.47	Urejanje uporabnika.....	89
5.3.48	Brisanje uporabnika.....	89
5.3.49	Pregled posameznega uporabnika.....	90
6	Razvoj sistema.....	92
6.1	Razvojno okolje.....	92
6.1.1	Eclipse.....	92
6.2	Nadzor razvoja.....	93
6.2.1	SVN.....	93
6.2.2	TRAC.....	94
6.3	Izdelava dizajna in postavitve.....	95
6.4	Izdelava aplikacije.....	98
6.4.1	Sloj podatkov.....	99

6.4.1.1 Konceptualni in logični podatkovni model.....	99
6.4.1.1.1 Entiteta “groups”.....	100
6.4.1.1.2 Entiteta “users”.....	101
6.4.1.1.3 Entiteta “acos”.....	101
6.4.1.1.4 Entiteta “aros”.....	102
6.4.1.1.5 Entiteta “aros_acos”.....	103
6.4.1.1.6 Entiteta “locations”.....	103
6.4.1.1.7 Entiteta “categories”.....	104
6.4.1.1.8 Entiteta “categories_videos”.....	104
6.4.1.1.9 Entiteta “videos”.....	105
6.4.1.1.10 Entiteta “selected_videos”.....	106
6.4.1.1.11 Entiteta “statistics”.....	106
6.4.1.1.12 Entiteta “pages”.....	107
6.4.1.2 Fizični podatkovni model.....	107
6.4.2 Sloj dostopa do podatkov.....	108
6.4.3 Sloj poslovne logike.....	110
6.4.3.1 Komponente.....	111
6.4.3.1.1 Avtentikacija in avtorizacija.....	111
6.4.3.1.2 FLV Metadata.....	112
6.4.3.1.3 FLV Upload.....	112
6.4.3.1.4 Ostranjevanje.....	113
6.4.3.1.5 Email.....	113
6.4.3.2.6 Cookie.....	113
6.4.3.2 Kontrolerji.....	113
6.4.3.2.1 Groups.....	113
6.4.3.2.2 Users.....	114
6.4.3.2.3 Categories.....	115
6.4.3.2.4 Videos.....	115
6.4.3.2.5 CategoriesVideos.....	117
6.4.3.2.6 Locations.....	117
6.4.3.2.7 SelectedVideos.....	118
6.4.3.2.8 Statistics.....	120
6.4.3.2.9 Pages.....	120
6.5 Sloj uporabniškega vmesnika.....	121
6.5.1 Elementi.....	121
6.5.2 Pomočniki.....	122
6.5.2.1 Format.....	122
6.5.2.2 FLV Upload.....	122
6.5.2.3 DatetimePicker.....	123
6.5.2.4 ColorPicker.....	124
6.5.2.5 Form.....	124
6.5.2.6 Html.....	125
6.5.2.7 Javascript.....	125
6.5.3 Pogledi.....	126
6.5.3.1 Dodajanje in urejanje izbranega videa.....	126

6.5.3.2 Pregled lokacije.....	127
6.5.3.3 Pregled statistike.....	128
6.6 Izdelava distribucijskega kanala.....	129
6.6.1 Predvajalnik videov.....	129
6.6.1.1 Params.....	130
6.6.1.2 UI.....	130
6.6.1.3 VideoStream.....	132
6.6.1.4 Player.....	134
6.6.2 Vključitev predvajalnika v spletno stran.....	134
6.6.2.1 Preverjanje ustreznosti brskalnika.....	135
6.6.2.2 Preverjanje ustreznosti predvajalnika Flash.....	135
6.6.2.3 Vključitev.....	136
6.7 Produciranje videov.....	139
6.7.1 Snemanje.....	139
6.7.2 Transparentnost videa.....	140
6.7.3 Kodiranje videa.....	140
6.8 Težave pri razvoju.....	141
7 Sklepne ugotovitve.....	143
Literatura.....	144

Kazalo slik

Slika 4.1: Tipičen tok procesiranja zahteve.....	27
Slika 4.2: Tok procesiranja pri vzorcu MVC.....	28
Slika 4.3: Graf razširjenosti multimedijskih dodatkov.....	36
Slika 5.1: Drevesna struktura aktivnosti projekta.....	43
Slika 5.2: Aktivnostna mreža projekta.....	45
Slika 5.3: Primeri uporabe za obiskovalce spletne strani.....	47
Slika 5.4: Primeri uporabe za uporabnika (prijavljenega obiskovalca).....	51
Slika 5.5: Primeri uporabe za uporabnika glede na lokacije.....	60
Slika 5.6: Primeri uporabe za uporabnika glede vključitve videov v spletno stran.....	64
Slika 5.7: Primer uporabe za obiskovalce gostujoče spletne strani.....	65
Slika 5.8: Primera uporabe za urednika (prijava in odjava).....	66
Slika 5.9: Primeri uporabe za urednika v okviru kategorij.....	68
Slika 5.10: Primeri uporabe za urednika v okviru videov.....	72
Slika 5.11: Primeri uporabe za urednika v okviru vsebinskih strani.....	78
Slika 5.12: Primeri uporabe za urednika v okviru uporabniških skupin.....	82
Slika 5.13: Primeri uporabe za urednika v okviru uporabnikov.....	85
Slika 6.1: Seznam aktivnosti v sistemu TRAC.....	93
Slika 6.2: Predogled dizajna.....	94
Slika 6.3: Logični podatkovni model.....	98
Slika 6.4: Gnezdene množice in vrednosti atributa lft in right (primer domene elektronskih naprav).....	100
Slika 6.5: Kontrole v predvajalniku.....	117
Slika 6.6: Prikazan gumb, če vključitev ni možna.....	117
Slika 6.7: Uporaba pomočnika "DatetimePicker".....	121
Slika 6.8: Uporaba pomočnika "ColorPicker".....	122
Slika 6.9: Obrazec za urejanje izbranega videa.....	125
Slika 6.10: Prikaz spreminjanje vrstnega reda videov za določeno lokacijo.....	126
Slika 6.11: Prikaz statistike (graf je bil generiran s knjižnico Raphael).....	126
Slika 6.12: Razredi in njihova soodvisnost.....	127
Slika 6.13: Vključen predvajalnik v spletno stran www.progmbh.com	137
Slika 6.14: Fotografiji studia.....	138

Kazalo tabel

Tabela 4.1: Struktura direktorijev za aplikacijo v ogrodju Cake.....	28
Tabela 4.2: Primeri konvencije ogrodja CakePHP.....	29
Tabela 6.1: Relacije entitete “groups”.....	98
Tabela 6.2: Atributi entitete “groups”.....	98
Tabela 6.3: Relacije entitete “users”.....	99
Tabela 6.4: Atributi entitete “users”.....	99
Tabela 6.5: Relacije entitete “acos”.....	100
Tabela 6.6: Atributi entitete “acos”.....	100
Tabela 6.7: Relacije entitete “aros”.....	100
Tabela 6.8: Atributi entitete “aros”.....	100
Tabela 6.9: Relacije entitete “aros_acos”.....	101
Tabela 6.10: Atributi entitete “arosa_acos”.....	101
Tabela 6.11: Relacije entitete “locations”.....	101
Tabela 6.12: Atributi entitete “locations”.....	102
Tabela 6.13: Relacije entitete “categories”.....	102
Tabela 6.14: Atributi entitete “categories”.....	102
Tabela 6.15: Relacije entitete “categories_videos”.....	103
Tabela 6.16: Atributi entitete “categories_videos”.....	103
Tabela 6.17: Relacije entitete “videos”.....	103
Tabela 6.18: Atributi entitete “videos”.....	103
Tabela 6.19: Relacije entitete “selected_videos”.....	104
Tabela 6.20: Atributi entitete “selected_videos”.....	104
Tabela 6.21: Relacije entitete “statistics”.....	105
Tabela 6.22: Atributi entitete “statistics”.....	105
Tabela 6.23: Atributi entitete “pages”.....	105
Tabela 6.24: Atributi komponente “FlvMetadata”.....	110
Tabela 6.25: Metode komponente “FlvMetadata”.....	110
Tabela 6.26: Metode komponente “FlvUpload”.....	110
Tabela 6.27: Metode kontrolerja “GroupsController”.....	111
Tabela 6.28: Metode kontrolerja “UsersController”.....	112
Tabela 6.29: Metode kontrolerja “CategoriesController”.....	113
Tabela 6.30: Metode kontrolerja “VideosController”.....	113
Tabela 6.31: Metode kontrolerja “CategoriesVideosController”.....	115
Tabela 6.32: Metode kontrolerja “LocationsController”.....	115
Tabela 6.33: Metode kontrolerja “SelectedVideosController”.....	116
Tabela 6.34: Metode kontrolerja “StatisticsController”.....	118
Tabela 6.35: Metode kontrolerja “PagesController”.....	118
Tabela 6.36: Metode pomočnika “Format”.....	120
Tabela 6.37: Metode pomočnika “FlvUpload”.....	120
Tabela 6.38: Metoda pomočnika “DatetimePicker”.....	121
Tabela 6.39: Metoda pomočnika “ColorPicker”.....	122
Tabela 6.40: Najpomembnejše metode pomočnika “Form”.....	122
Tabela 6.41: Najpomembnejše metode pomočnika “Html”.....	123

Tabela 6.42: Najpomembnejše metode pomočnika “Javascript”	123
Tabela 6.43: Metode razreda “Params”	128
Tabela 6.44: Metode razreda “UI”	129
Tabela 6.45: Metode razreda “VideoStream”	130
Tabela 6.46: Metode razreda “Player”	132

Seznam uporabljenih kratic

IP	Internet Protocol
URL	Uniform Resource Locator
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
CSS	Cascading Style Sheets
MVC	Model-View-Controller
RAD	Rapid Application Development
XML	Extensible Markup Language
SQL	Structured Query Language
Ajax	Asynchronous JavaScript And XML
CRUD	Create, Read, Update, Delete
GNU	Gnu's Not Unix
SGML	Standard Generalized Markup Language
URI	Uniform Resource Identifier
DOM	Document Object Model
IDE	Integrated Development Environment
SWF	Small Web Format ali ShockWave Flash
AS3	Actionscript 3
FLV	Flash Live Video
CODEC	COmpressor / DECompressor ali COder / DECoder
MP3	MPEG-1 Audio Layer 3 (MPEG – Motion Picture Experts Group)
WYSIWYG	What You See Is What You Get
PERT	Program Evolution and Review Technique
UML	Unified Modeling Language
PDT	PHP Development Tools
WTP	Web Tools Platform
DLTK	Dynamic Language ToolKit
AXDT	Actionscript 3 and FleX Development Tools
IMP	IDE Meta-Tooling Platform
SVN	Subversion
CVS	Concurrent Versions System
GIMP	GNU Image Manipulation Program
PNG	Portable Network Graphics
GIF	Graphics Interchange Format
MIME	Multipurpose Internet Mail Extensions
SVG	Scalable Vector Graphics
RGB	Red-Green-Blue

Povzetek

V diplomskem delu je opisan celoten razvojni cikel spletne aplikacije, ki omogoča uporabnikom vključevanje transparentnih videov v lastne spletne strani. Vključitev videa je izvedena tako, da se video predvaja nad celotno vsebino gostujoče spletne strani. S tem poskušamo izboljšati prepoznavnost in povečati odzivnost obiskovalcev gostujoče strani.

Aplikacijo smo izdelali z uporabo razvojnega ogrodja CakePHP, ki temelji na filozofiji hitrega in učinkovitega razvoja spletnih aplikacij. Za predvajanje transparentnega videa se uporablja tehnologija Adobe FLASH, ki predstavlja de facto standard na področju predvajanja video vsebin.

Poglavje 2 vsebuje predstavitev dveh obstoječih rešitev na temo vključevanja videa v spletne strani. V poglavju 3 so načrtani poglavitni cilji in zahtevane funkcionalnosti, ki jih mora aplikacija izpolniti. Poglavje 4 opredeljuje tehnološko platformo, na kateri je aplikacija zgrajena. V poglavju 5 je predstavljen postopek načrtovanja aplikacije, kjer so določene aktivnosti in opisani primeri uporabe. Poglavje 6 opisuje postopek razvoja aplikacije, ki obsega izdelavo administrativnega in uporabniškega dela aplikacije ter prikaže razvoj predvajalnika video posnetkov. V tem poglavju je opisan tudi postopek produkcije video vsebin.

KLJUČNE BESEDE: video, CakePHP, Adobe Flash

Abstract

Contents of the diploma depict entire development cycle for a web application which provides users with the ability to include transparent videos into their web pages. The inclusion is implemented in a manner that enables video to be played above the contents of the web page. The application enables the users to improve their web page recognition and increase their visitors' positive feedback.

The application was developed using CakePHP framework, which is based upon philosophy of rapid application development. Adobe FLASH technology, a de facto standard in the domain of video playback in web pages, was used to enable transparent video playback.

Chapter 2 presents an overview of two existing solutions to the video inclusion problem. The main project goals and required functionalities are explained in chapter 3. Chapter 4 introduces a technological platform used for application development. Project activities and use cases are part of application planning, which is presented in chapter 5. Administrative and user part of the application is included in chapter 6. Development of video player and production of videos are part of chapter 6 as well.

KEYWORDS: video, CakePHP, Adobe Flash

1 Uvod

V zadnjih letih smo doživeli izredni porast uporabe videov v spletnih straneh. Vsaka spletna stran želi v vsebino dodati tudi video. Naj bo to spletna stran, ki omogoča dostop do novic, ali spletna trgovina, kjer so artikli predstavljeni z videom. Video posnetki dodajo dodatno vrednost vsebini, v katero so vključeni, in hkrati povečajo pozornost obiskovalca spletne strani na želeno vsebino. Poglavitna ideja naše aplikacije je izboljšati pozornost obiskovalcev na določene vsebine, ki so predstavljene v spletni strani. To dosežemo z uporabo transparentnih videov, ki jih v spletno stran vključimo na poseben in zanimiv način.

Za ta namen je potrebno izdelati aplikacijo, ki bo lastnikom spletnih strani omogočala enostavno in učinkovito uporabo videov. Videi, ki so na razpolago, so posneti vnaprej in prilagojeni za različne skupine spletnih strani. Aplikacija omogoča enostavno distribucijo teh videov.

Aplikacijo smo izdelali z uporabo učinkovitega ogrodja, imenovanega CakePHP. Za predvajanje videov v gostujočih spletnih straneh smo uporabili tehnologijo Adobe FLASH, ki je trenutno de facto standard na področju predvajanja videov v spletnih straneh.

2 Sistemi za distribucijo videov v spletne strani

Med brskanjem po svetovnem spletu smo naleteli na mnoge strani, ki omogočajo distribucijo videov. Najpopularnejša med njimi je definitivno www.youtube.com. Distribucija videov je mišljena kot metoda, s katero lahko uporabniki teh spletnih strani uporabljajo njihove videe na svojih straneh. Na primer na strani www.youtube.com lahko vsak posamezni video uporabimo za objavo na lastni spletni strani. To dosežemo z uporabo posebne kode, katero nam posreduje spletna stran in jo nato uporabimo za dodajanje izbranega videa v lastno spletno stran. Primerjava naše aplikacije s portalom, kot je www.youtube.com, se tu konča, kajti videe, ki so objavljeni na tej strani, ne moremo primerjati z videi, ki jih bomo uporabljali v naši aplikaciji.

Bolj poglobljeno iskanje po svetovnem spletu odkrije spletno stran podjetja Rovion (www.rovion.com), ki omogoča vključevanje enakega tipa videov, kot jih uporablja naša aplikacija. Ti videi so deloma transparentni, predvajajo se pa nad celotno vsebino gostujoče spletne strani. Vključevanje videov poteka po enakem postopku kot pri portalu www.youtube.com. Razliko predstavlja postopek njihovega kreiranja in vključevanja. Podjetje Rovion ne ponuja vnaprej posnetih videov, ki bi jih uporabniki dodajali v svoje strani, ampak ima definiran proces, skozi katerega gre vsak video, ki bo vključen v določeno spletno stran. Proces vključuje naslednje korake:

1. Cilji in strategija (klient določi cilje, podjetje Rovion izbere strategijo).
2. Predprodukcija (izdelava scenarija).
3. Produkcija (snemanje videa).
4. Postprodukcija (digitalizacija posnetkov in izdelava kode za vključitev).
5. Vključitev videa v spletno stran (posebno kodo se doda v spletno stran).

Naša aplikacija bo uporabljala podoben proces, vendar s pomembno razliko. Proces podjetja Rovion se izvede enkrat za vsak video in vsako stranko. Pri generiranju posameznega videa za našo aplikacijo bomo izvedli samo prve štiri korake procesa. Nato pa vsaka stranka oz. uporabnik, ki bo želel vključiti določen video v svojo stran, izvede peti korak. Naša aplikacija je zbirka vnaprej posnetih videov, med katerimi uporabniki izbirajo. Ko si izberejo primeren video, ga lahko sami dodajo v svojo spletno stran.

3 Opredelitev problema

Izdelati je potrebno spletno aplikacijo, ki bo služila distribuciji vnaprej posnetih kratkih videov. Distribucija je mišljena kot platforma, s katero bodo uporabniki na enostaven način dodajali izbrane videe v svoje spletne strani. Posebnost teh videov je njihova transparentnost in način vključitve v spletno mesto. Videi se prikažejo nad celotno vsebino spletnega mesta. Na ta način dodan video v spletni strani omogoča boljše osebno komunikacijo spletnega mesta z obiskovalcem ter popestri izgled in vtis, ki ga spletno mesto pusti na obiskovalcu. Vendar pa je pri uporabi takšnega dodatka spletni strani potrebno biti pazljiv. Pri preobsežni uporabi lahko dosežemo ravno nasprotni učinek, kot si želimo. Zato smo v aplikacijo vgradili mehanizem, ki poskuša to pomanjkljivost odpraviti. Uporabnik lahko pri izbiri videa določi t. i. frekvenco predvajanja, ki onemogoči predvajanje videa vsakokrat, ko posamezni obiskovalec obišče to spletno stran. Vendar ne večkrat, kot je to določeno s frekvenco predvajanja. Tako obiskovalca ne silimo k ogledu videa vsakokrat, ko obišče spletno mesto.

Uporabniki aplikacije izbirajo med množico videov, ki jih želijo vključiti v svojo spletno stran. Ko si izberejo želen video, določijo parametre predvajanja in aplikacija jim posreduje kodo, ki jo nato vključijo v svojo spletno stran. Pred določitvijo parametrov uporabniki definirajo lokacijo, kjer se bo izbrani video predvajal. Lokacija predstavlja naslov URL (angl. *Uniform Resource Locator*) spletnega mesta. Na isti lokaciji se lahko predvaja več videov. Aplikacija omogoča spreminjanje vrstnega reda predvajanja vseh videov za določeno lokacijo. Uporabniki lahko z določanjem frekvence in spreminjanjem vrstnega reda predvajanja pripravijo dinamično vsebino, ki bistveno popestri predstavitev spletnega mesta. Da se predvajalnik čim bolj zlije z gostujočim spletnim mestom, uporabnik lahko določi barve predvajalnika. Tako dosežemo boljše integracijo predvajalnika s spletnim mestom.

Aplikacija je v grobem razdeljena na dva dela. Prvi del vsebuje strežniško aplikacijo, ki služi distribuciji, kontroli in analizi videov. Uredniki aplikacije urejajo že dodane videe in dodajajo nove, ki si jih nato uporabniki izbirajo in vključujejo v svoja spletna mesta. Drugi del pa obsega sistem za predvajanje videov v gostujočem spletnem mestu. Predvajalnik preveri skladnost obiskovalčeve platforme in v primeru ustreznosti prične z vključevanjem videa. Preden doda video v spletno mesto, preveri še parametre, ki jih je določil uporabnik. Če vrednosti parametrov omogočajo predvajanje, predvajalnik zažene video. Sicer pa prikaže gumb, s pomočjo katerega lahko obiskovalec ročno zažene video.

3.1 Vsebinske zahteve

Vsebinske zahteve so razdeljene v tri sklope. Prvi sklop določa funkcionalnosti, ki jih mora imeti na razpolago urednik aplikacije. Drugi sklop vključuje funkcionalnosti, ki jih uporabljajo uporabniki aplikacije. Tretji sklop pa definira funkcionalnosti predvajalnika video vsebin.

Funkcionalnosti na strani urednika aplikacije so sledeče:

1. Avtentikacija in avtorizacija urednikov.
2. Nadzor nad uporabniki (pregled, urejanje, brisanje) in uredniki (dodajanje, pregled, urejanje, brisanje) aplikacije.
3. Pregledovanje skupin, katerim pripadajo tako uredniki kakor tudi navadni uporabniki. Določeni naj bosta dve skupini: administratorji in uporabniki.
4. Specifične pravice, ki jih dovolimo urednikom in uporabnikom, določamo za posamezno skupino.
5. Upravljanje kategorij:
 1. Pregledovanje in brisanje kategorij.
 2. Dodajanje in urejanje kategorije naj omogoča določanje nadrejene kategorije, imena in opisa.
 3. Možnost urejanja vrstnega reda podkategorij in videov v posamezni kategoriji.
6. Upravljanje videov:
 1. Pregledovanje in brisanje videov.
 2. Dodajanje in urejanje videov naj omogoča določanje pripadnosti kategorijam, imena, opisa, jezika (v videu) in nalaganje datoteke, ki predstavlja video.
 3. Tvorjenje sličice, ki predstavlja video (angl. *thumbnail picture*).
 4. Avtomatsko pridobivanje podatkov o videu (trajanje, širina, višina).
7. Urejanje vsebinskih strani:
 1. Pregledovanje in brisanje vsebinskih strani.

2. Dodajanje in urejanje vsebinskih strani naj omogoča določanje imena, aliasa, vsebine in segmenta (navigacija v glavi in nogi spletne strani).
8. Za dodajanje in urejanje vsebin in opisov naj se uporabi WYSIWYG (angl. *What You See Is What You Get*) urejevalnik.

Funkcionalnosti na strani uporabnika aplikacije, ko še ni prijavljen v aplikaciji:

1. Prikaz videov, ki so bili nazadnje dodani.
2. Pregledovanje kategorij in videov znotraj posamezne kategorije (največ 6 videov na stran). Ob posameznem videu prikažemo podatke tega videa: trajanje in jezik.
3. Ogled posameznega videa, kjer prikažemo naslednje podatke: trajanje, višina, širina, jezik, velikost datoteke in pripadnost kategorijam. Če je uporabnik prijavljen v sistem, se prikaže tudi gumb za izbiro videa, sicer se prikaže gumb za prijavo v sistem.
4. Pregledovanje vsebinskih podstrani.
5. Registracija uporabnika, kjer uporabnik določi svoje uporabniško ime, geslo, e-naslov, ime in priimek.
6. Prijava uporabnika v aplikacijo z uporabo uporabniškega imena in gesla.

Funkcionalnosti na strani uporabnika aplikacije, ko je prijavljen v aplikaciji:

1. Avtentikacija in avtorizacija uporabnikov.
2. Upravljanje izbranih videov:
 1. Pregled vseh videov.
 2. Brisanje posameznega videa.
 3. Dodajanje in urejanje izbranega videa naj omogoča določanje:
 1. lokacije, kjer se bo video predvajal,
 2. pozicije na gostujoči spletni strani (levo zgoraj, desno zgoraj, sredina, levo spodaj, sredina spodaj, desno spodaj),
 3. načina pozicioniranja (plavajoč, fiksiran),

4. frekvence predvajanja,
 5. veljavnosti predvajanja videa (od datuma do datuma),
 6. širine in višine predvajalnika (prosto spreminjanje in spreminjanje, ki obdrži razmerje med višino in širino),
 7. barv predvajalnika (aktivna barva, neaktivna barva, barva robu in barva ozadja),
 8. povezave URL, kamor se preusmeri uporabnika, ko klikne na video.
4. Pri pregledu posameznega videa naj se, ob vseh zgoraj naštetih lastnosti, prikaže tudi koda, ki jo uporabnik uporabi za vključitev videa v svojo spletno stran. Hkrati se mu prikaže gumb, ki omogoča predogled videa v gostujoči spletni strani (demonstracija).
 5. Pregled statistike za posamezni video. Statistiko prikažemo z grafom. Uporabnik ima možnost določiti obdobje, natančnost (ura, dan in mesec) in vrsto (prikazi videov in kliki na videe ali oboje) prikaza statistike. Ko je statistika prikazana, ima uporabnik možnost izvoza statistike v preglednico in sliko.
3. Upravljanje lokacij:
 1. Pregled vseh lokacij.
 2. Brisanje posamezne lokacije.
 3. Pri dodajanju nove lokacije je potrebno določiti ime in URL, kjer bo uporabnik predvajal izbrane videe.
 4. Pregled posamezne lokacije naj prikaže ime in lokacijo URL. Ker lahko uporabnik na isti lokaciji predvaja več različnih videov, je potrebno prikazati vse videe za to lokacijo. Potrebno je tudi omogočiti možnost spreminjanja vrstnega reda predvajanja videov za to lokacijo. Ko ima uporabnik za posamezno lokacijo določenih več videov, je potrebno izpisati kodo, s katero uporabnik vključi videe v svojo spletno stran.
 4. Uporabnik ima možnost spreminjanja podatkov svojega profila.

Funkcionalnosti predvajalnika:

1. Omogočati mora kontrolo predvajanja, in sicer: trenutno prekinitev in nadaljevanje predvajanja, utišanje zvoka in izklop predvajalnika.
2. Ob izklopu predvajalnika se prikaže gumb, ki omogoča ponovni zagon.

3. Ko uporabnik klikne na video, se ga preusmeri na vnaprej določeno spletno mesto.
4. Potrebno je beležiti predvajanja in klike na video (statistika).
5. Preverjanje skladnosti brskalnika in predvajalnika Adobe Flash Player.

3.2 Tehnološke zahteve

Pri razvoju je potrebno upoštevati naslednje tehnološke zahteve:

- aplikacija mora omogočati iste funkcionalnosti v brskalnikih Mozilla Firefox 2.5+, Internet Explorer 7+, Opera 8+, Safari 3+ in Chrome 4+,
- predvajanje videa mora biti omogočeno z uporabo predvajalnika Adobe Flash Player 9+,
- vključevanje predvajalnika v zunanje spletne strani mora biti omogočeno z uporabo ene vrstice kode,
- aplikacija naj za delovanje uporablja odprtokodne programske rešitve.

4 Tehnološka platforma

4.1 Strežniške tehnologije

Vsaka spletna aplikacija za svoje delovanje potrebuje strežnik. Na razpolago imamo širok nabor možnih strežnikov. Za našo aplikacijo bomo uporabili t. i. strežnik LAMP. To je sklad programske opreme, ki teče na operacijskem sistemu Linux. Izkazal se je za učinkovit, robusten in varen sistem. V skladu so vključeni naslednji programski paketi:

- GNU/Linux,
- Apache,
- MySQL in
- PHP.

Vsak element v skladu je odprtokodni sistem. Vedno več podjetij uporablja LAMP, kajti stabilnost in hitrost posameznega programskega paketa prekašata mnoge ekvivalentne licenčne strežnike [6].

4.1.1 Linux

Linux je operacijski sistem, ki poganja uporabniške programe. Posebej je poznan po svoji stabilnosti, varnosti in hitrosti. Predstavlja osnovni gradnik pri postavitvi strežnika.

4.1.2 Apache

Ob operacijskem sistemu je najosnovnejša programska oprema, ki jo potrebujemo za uspešno delovanje spletnega mesta, spletni strežnik. Trenutno najpogosteje uporabljen spletni strežnik je Apache. Razlogi, da je Apache tako razširjen, so sledeči:

- je izredno hiter in zanesljiv,
- je odprtokodni sistem, kar pomeni, da je brezplačen in da ga vzdržuje in razvija številčna ekipa prostovoljcev,
- je eden izmed najboljših spletnih strežnikov za operacijske sisteme, ki bazirajo na Unix-u,

- načrtovan je bil kot izredno razširljiv.

Apache v osnovi ne vsebuje vseh možnih funkcionalnosti, ampak le osnovne funkcije. Vse dodatne funkcionalnosti lahko dodamo preko t. i. modulov. Vseh razvitih modulov je preko 500 [1].

Spletni strežnik konfiguriramo s pomočjo konfiguracijske datoteke *httpd.conf*. Posamezne nastavitve dosežemo z uporabo direktiv, ki jih vpišemo v konfiguracijsko datoteko. Za potrebe pričujočega sistema je potrebno izdelati konfiguracijsko datoteko, ki vsebuje direktivo *VirtualHost*. Z uporabo te direktive lahko strežnik Apache gosti več spletnih mest v okviru enega fizičnega strežnika [2].

```
<VirtualHost *:80>
    ServerName video-gadget.iljaz.com
    DocumentRoot /www/sites/video-gadget.iljaz.com/
    DirectoryIndex index.php
    <Directory /www/sites/video-gadget.iljaz.com/>
        Options FollowSymLinks -Indexes
        Order allow,deny
        Allow from all
    </Directory>
    ErrorLog /var/logs/error/video-gadget.iljaz.com.log
    CustomLog /var/logs/access/video-gadget.iljaz.com.log
    combined
</VirtualHost>
```

Primer 4.1: Vsebina konfiguracijske datoteke

V prvi vrstici primera 4.1 definiramo IP (angl. *Internet Protocol*) naslov in številko vrat, na katerih bo strežnik poslušal za prihajajoče povezave. V našem primeru * predstavlja katerikoli IP naslov. *ServerName* določa ime virtualnega gostitelja. *DocumentRoot* določa direktorij znotraj gostujočega operacijskega sistema, v katerem se nahajajo datoteke, potrebne za izvajanje aplikacije. Da bi tudi uporabniki, ki direktno dostopajo do korenskega direktorija, pridobili uporabne informacije, definiramo *DirectoryIndex* [2].

Direktiva *Directory* določa posebne lastnosti direktorija, v katerem se nahaja aplikacija. Vrednost *FollowSymLinks* direktive *Options* omogoča sledenje simboličnim povezavam znotraj direktorija. Vrednost *-Indexes* pa onemogoča prikaz seznama datotek znotraj direktorija. Nato definiramo tudi pravice za prikaz vsebine direktorija. V zgornjem primeru omogočimo vsakomur dostop do

direktorija [2].

Na koncu definiramo še direktivi *ErrorLog* in *CustomLog*, ki določata datoteki, v katerih se zapisujejo podatki o napakah med izvajanjem in podatki o vsaki zahtevi, na katero se je strežnik odzval [2].

Vsebina iz primera 4.1 omogoča Apache strežniku, da odgovarja na zahteve, ki prihajajo na URL naslov *video-gadget.iljaz.com*.

4.1.3 MySQL

Za čim učinkovitejše shranjevanje podatkov večina aplikacij uporablja relacijske podatkovne baze. Najpogostejše uporabljena relacijska podatkovna baza za spletne strežnike je MySQL. MySQL je odprtokodni sistem, in ravno brezplačna uporaba je eden izmed razlogov za njegov uspeh. Ostali razlogi njegovega uspeha so širok nabor funkcij, izjemna hitrost in dobra skalabilnost [3].

Za potrebe aplikacije smo na strežnik nastavili MySQL in ustvarili novo podatkovno bazo, ki bo hranila podatke, potrebne za delovanje aplikacije. Za uspešen dostop do podatkov v bazi je potrebno ustvariti uporabnika, ki ima pravice za dostop do teh podatkov.

```
CREATE DATABASE videogadget;
GRANT PRIVILEGES ON videogadget.* TO 'videogadget' IDENTIFIED BY
'password';
```

Primer 4.2: Ukaza, ki ustvarita podatkovno bazo in upravnika

Ukaz v prvi vrstici primera 4.2 je samoumeven – ustvari novo podatkovno bazo s podanim imenom. Druga vrstica vsebuje ukaz, ki ustvari novega uporabnika in mu dodeli pravice za uporabo vseh objektov v okviru podatkovne baze *videogadget*.

Objekte podatkovne baze lahko manevriramo z uporabo različnih orodij. Najbolj razširjeno orodje je *phpMyAdmin*. To je spletna aplikacija, ki vsebuje večino funkcij, potrebnih za uspešno upravljanje s podatkovno bazo. Do objektov podatkovne baze lahko dostopamo tudi z uporabo ukaznega vmesnika (CLI – angl. *Command Line Interface*), ki je vključen v distribuciji programskega paketa MySQL.

4.1.4 PHP

PHP je splošnonamenski skriptni jezik, ki ga uporabljamo na strežniku za generiranje dinamičnih vsebin. Za ta namen je PHP koda vključena v izvorni HTML (angl. *Hyper Text Markup Language*) dokument in interpretirana s spletnim strežnikom. PHP lahko uporabljamo na večini spletnih strežnikov, mnogih operacijskih sistemih ter ga lahko povežemo z mnogimi sistemi relacijskih podatkovnih baz. Najpogosteje je PHP uporabljen skupaj z Apache spletnim strežnikom, v katerega je vključen z uporabo modula *mod_php*, in MySQL podatkovno bazo.

4.1.4.1 CakePHP

Ko želimo dodati določeno dinamično funkcionalnost v obstoječo spletno stran, lahko prebrskamo internet in najdemo veliko različnih PHP skript, ki opravljajo želeno funkcijo. Izbrano PHP skripto vključimo v spletno stran in spremenimo zahtevane nastavitve. Težava pa nastane, ko želimo v skripti implementirano funkcionalnost spremeniti oz. dograditi glede na naše želje. V tem primeru je potrebno najti mesto v skripti, kjer je implementirana operacija, ki jo želimo spremeniti. Tako skripta upravlja s tokom kontrole, v smislu, da se vse operacije izvedejo, ko se skripta požene. Edini način, da prevzamemo kontrolo izvajanja operacij, je spreminjanje skripte.

Ogrodje (angl. *framework*) je ena izmed možnosti, ki imajo tok kontrole invertiran. Za implementacijo določene funkcionalnosti je potrebno definirati objekte, ki to funkcionalnost omogočajo. Ogrodje poveže te objekte v celoto in tako implementira želeno funkcionalnost. Dana nam je možnost prestrezanja operacij teh objektov in dodajanja novih operacij ali razširjanja obstoječih. Knjižnice ali izdelane skripte imajo implementirano celotno logiko za doseg zastavljenega cilja. Spremembe logike dosežemo s spreminjanjem originalne knjižnice ali skripte. Nasprotno, ogrodja za doseg želene funkcionalnosti ne spreminjamo. To izvedemo z dodajanjem in razširjanjem objektov v ogrodju. Ogrodje samo po sebi ne implementira nobene končne funkcionalnosti, šele naši dodatki in razširitve definirajo neko uporabno funkcionalnost [4].

CakePHP (na kratko Cake) je ogrodje, ne pa množica knjižnic, čeprav vsebuje določene knjižnice. Cake od razvijalca pričakuje, da razširi obstoječe objekte in doda nove kostumizirane resurse. Zagotovljena je kompatibilnost s PHP verzijo 4 in 5, kar predstavlja prednost v primerjavi z drugimi, podobnimi ogrodji, ki so kompatibilni le z določeno verzijo PHP. Razvoj aplikacij z uporabo Cake ogrodja je zelo enostaven in hiter, kajti krivulja učenja je kratka. Cake je enostaven in preprost, kljub temu pa predstavlja močno in zelo uporabno ogrodje za razvoj.

Glavne lastnosti ogrodja Cake so:

- uporablja MVC vzorec,
- vključuje povezljivost z mnogimi sistemi podatkovnih baz, npr. MySQL, PostgreSQL in ostalimi,
- enostavna vzpostavitev ogrodja na Linux in Windows platformah,
- uporablja enostavno in fleksibilno implementacijo predlog (angl. *templates*),
- vključuje pomočnike (angl. *helpers*), ki nudijo pomoč pri izpisu oznak HTML, izpisu obrazcev, uporabi Ajax tehnologije itd.,
- vključuje komponente (angl. *components*) za enostavnejše rokovanje s pošiljanjem e-sporočil, avtentikacijo, kontrolo dostopa, lokalizacijo, varnostjo itd.,
- generira URL naslove, ki so optimizirani za iskalnike [5].

4.1.4.1.1 Hiter razvoj aplikacij

Cake je prevzel filozofijo hitrega razvoja aplikacij (RAD – angl. *Rapid Application Development*), katerega namen je zmanjševanje časa, potrebnega za načrtovanje aplikacije. To doseže z uporabo številnih, vnaprej pripravljenih struktur, ki omogočajo enostavnejše vzdrževanje, večkratno uporabo iste kode, učinkovitejše timsko delo in hitrejši odziv na spremembe s strani naročnika. En način pospešitve je uporaba t. i. metode *scaffolding*, ki z uporabo ene vrstice kode omogoča generiranje predogledov posameznih segmentov aplikacije. Glede na strukturo podatkovne baze izpiše obrazce HTML, ki služijo dodajanju, urejanju, pregledovanju in brisanju predmetov iz baze. V primeru sprememb podatkovnega modela zaradi novih zahtev s strani naročnika ni potrebno spreminjanje programske kode. Spremembe so takoj vidne v aplikaciji. Na tak način lahko med načrtovanjem aplikacije preizkušamo različne možnosti brez dodatnega programiranja [4].

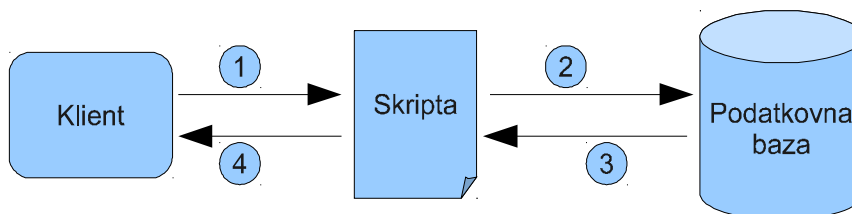
Strukture, pridobljene z uporabo metode *scaffolding*, niso primerne za vključitev v končno verzijo aplikacije. Kljub temu ni potrebno vseh struktur aplikacije izdelati ročno, kajti tudi za ta namen ima Cake razvito orodje. Preko ukaznega vmesnika upravljamo z orodjem Bake, ki nam omogoča generiranje posameznih struktur aplikacije. Orodje glede na strukturo baze generira programsko kodo, ki nam omogoča ustvarjanje, pregledovanje, urejanje in brisanje predmetov iz baze. Te štiri operacije definirajo t. i. funkcionalnost CRUD (angl. *Create, Read, Update, Delete*)

[4]. Večina aplikacij vsebuje predmete, s katerimi uporabniki manevrirajo. Osnovno upravljanje s predmeti pa je možno ravno s temi štirimi operacijami.

4.1.4.1.2 Model-Pogled-Kontroler

Cake pri razvoju aplikacije programerja sili v uporabo t. i. vzorca Model-Pogled-Kontroler (MVC – angl. *Model-View-Controller*). To je način strukturiranja programske kode, ki združuje programsko kodo v tri glavne skupine. Model vsebuje kodo, ki komunicira s podatkovno bazo. Pogled skrbi za izpis in prikaz struktur aplikacije. Kontroler pa vsebuje kodo, ki opredeljuje logiko aplikacije. Takšen princip programiranja spremeni način procesiranja posamezne zahteve na strani strežnika. Slika 4.1 prikazuje tok procesiranja zahteve, ki uporablja standardno strukturiranje programske kode.

Osnovna ideje procesiranja zahteve, kot je prikazana v sliki 4.1, je, da element *Skripta* (prikazan v sliki) vsebuje celotno programsko kodo. V večini primerov programerji uporabljajo klic PHP funkcije *include()*, ki vključi vse zunanje datoteke, kjer se nahajajo skupne funkcije. Na tak način odpravijo redudanco programske kode.



Slika 4.1: Tipičen tok procesiranja zahteve

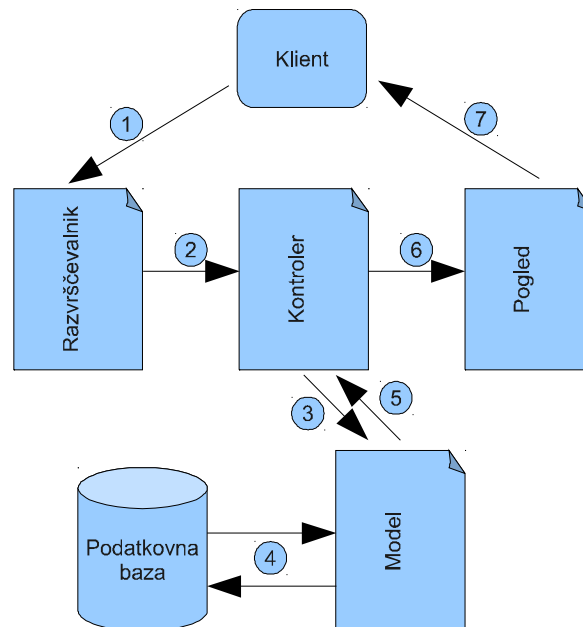
Vzorec MVC izboljšuje zgoraj prikazani tok procesiranja zahteve. Glavna ideja vzorca MVC je napisati vsako funkcijo v aplikaciji enkrat in samo enkrat. Cake se tega loteva z definiranjem lokacij, kjer lahko posamezno operacijo zapišemo [4]. Na tak način poveže vse potrebne resurse med sabo brez kakršnekoli konfiguracije. Ravno to pa je največja prednost Cake ogrodja pred ostalimi, ki za doseg istega cilja uporabljajo več konfiguracijskih datotek. Problem številnih konfiguracijskih datotek je težavno vzdrževanje aplikacije. Torej, Cake daje prednost konvencijam pred konfiguriranjem.

Tok procesiranja zahteve z uporabo MVC vzorca je sledeči (glej sliko 4.2):

1. Ko klient klikne neko povezavo ali vpiše URL naslov direktno, brskalnik pošlje zahtevo aplikaciji. Tipična struktura URL naslova je:

http://<domena>.com/<aplikacija>/<kontroler>/<akcija>/<parameter 1, itd.>

2. Skripta za razvrščanje (angl. *dispatcher*) razčleni URL naslov in ugotovi, kateri kontroler mora pognati. V izbranem kontrolerju pokliče akcijo (<akcije>) in ji pošlje parametre.
3. Akcija v kontrolerju sprocesira zahtevo, pri čemer uporablja model.
4. Model abstrahira tabelo podatkovne baze in ima definirane metode, ki operirajo s podatki v tej tabeli.
5. Ko je model zapisal in pridobil vse podatke iz tabele, jih vrne kontrolerju.
6. Kontroler nato ponovno sprocesira pridobljene podatke in jih pošlje pogledu.
7. Pogled oblikuje prikaz dobljenih podatkov in jih pošlje klientu.



Slika 4.2: Tok procesiranja pri vzorcu MVC

Z uporabo MVC vzorca lahko funkcije, ki jih uporabljamo večkrat, ločimo in tako omogočimo hitrejšo spreminjanje aplikacije. Tako ločene funkcije tudi pripomorejo med iskanjem napak oz. razhroščevanjem. Če se pri interakciji s podatkovno bazo pojavljajo napake, potem lahko predvidevamo, da je napaka prisotna v modelu.

4.1.4.1.3 Konvencije

Konvencije v ogrodju Cake opredeljujejo strukturo direktorijev pri gradnji aplikacije in poimenovanje datotek. Tabela 4.1 prikazuje strukturo direktorijev, v katerih kreiramo datoteke, ki omogočajo funkcionalnosti aplikacije.

Tabela 4.1: Struktura direktorijev za aplikacijo v ogrodju Cake

Direktorij aplikacije	Poddirektorij	Opis
config		Vsebuje konfiguracijske datoteke. Podatke za dostop do podatkovne baze.
controllers		Vsebuje kontrolerje.
controllers	components	Komponente razširjajo kontrolerje z dodatnimi funkcionalnostmi. Več kontrolerjev lahko uporablja isto komponento.
locale		Vsebuje datoteke, ki omogočajo internacionalizacijo aplikacije.
models		Vsebuje modele.
models	behaviours	Vsebuje razrede, ki razširjajo modele z dodatnimi funkcionalnostmi.
models	datasources	Vsebuje datoteke, ki omogočajo uporabo različnih podatkovni baz, npr. datoteke XML.
plugins		Vsebuje pakete dodatkov.
tmp		Cake tukaj shranjuje začasne datoteke.
vendors		Dodajamo datoteke in pakete tretjih strank.
views		Vsebuje poglede.
views	elements	Omogočajo večkratno uporabo istih segmentov v več pogledov.
views	errors	Vsebuje poglede, ki omogočajo posebne poglede v primeru različnih napak med izvajanjem aplikacije.
views	helpers	Razširjajo osnovne funkcionalnosti pogledov.
views	layouts	Določajo osnovne postavitve spletnih strani.
webroot		Določa lokacijo, ki je dostopna uporabnikom spletne aplikacije.
webroot	css	Vsebuje datoteke, ki določajo stile.
webroot	img	Vsebuje slike, ki se prikazujejo v aplikaciji.
webroot	js	Datoteke vsebujejo kodo v jeziku JavaScript.

Drugi del konvencij opredeljuje poimenovanje struktur pri gradnji aplikacije. V splošnem imena datotek zapisujemo s podčrtajem, medtem ko imena razredov v teh datotekah pišemo z velikimi začetnicami. Če kreiramo razred z imenom **MojRazred**, potem je potrebno ustvariti datoteko, poimenovano **moj_razred.php**. Konvencija tudi opredeljuje poimenovanje tabel v podatkovni bazi in razredov, ki predstavljajo te tabele (t. i. modeli). Tabele so poimenovane z imeni v množini, razredi pa v ednini, pri tem tudi upoštevamo prejšnjo konvencijo. Če imamo

tabelo z imenom `big_cars`, potem kreiramo razred z imenom `BigCar` v datoteki `big_car.php`. Imena kontrolerjev poimenujemo v množini, z velikimi začetnicami in na koncu imena dodamo besedo `Controller`. Primer imena kontrolerja, ki upravlja z modelom `BigCar`, je `BigCarsController` v datoteki `big_cars.php`. Predloge (angl. *templates*), ki predstavljajo poglede (angl. *views*), poimenujemo glede na akcije v kontrolerju, in sicer s podčrtaji in z malimi začetnicami. Če v kontrolerju `BigCarsController` definiramo akcijo `viewRedCars`, potem bo Cake poskušal naložiti predlogo z imenom `view_red_cars.ctp` v direktoriju `big_cars`. Vsa imena predlog oz. pogledov v ogrodju Cake zaključimo s končnico `.ctp`. Tabela 4.2 prikazuje vse konvencije poimenovanja in njihovo lokacijo v hierarhiji direktorijev.

Tabela 4.2: Primeri konvencije ogrodja CakePHP

Struktura	Datoteka	Lokacija
Tabela: <code>big_cars</code>	/	podatkovna baza
Model: <code>BigCar</code>	<code>big_car.php</code>	<code>app/models</code>
Kontroler: <code>BigCarsController</code>	<code>big_cars.php</code>	<code>app/controllers</code>
Pogled za akcijo: <code>viewRedCars</code> v kontrolerju <code>BigCarsController</code>	<code>view_red_cars.ctp</code>	<code>app/views/big_cars</code>

Omeniti je še potrebno posebnost pri poimenovanju primarnih in tujih ključev v podatkovnih entitetah. Cake za primarne ključe ne more uporabljati sestavljenih ključev. Po konvenciji ima vsaka entiteta primarni ključ, poimenovan z `“id”`. Če želimo uporabljati sestavljene ključe, je potrebno kreirati umetni ključ z imenom `“id”`. Tuji ključi v tabelah morajo biti poimenovani v ednini in na koncu vsebovati `“_id”`. Ime mora izhajati iz imena entitete, kateri tuji ključ pripada. Če imamo entiteto `“brands”` s primarni ključem `“id”`, potem je tuji ključ v entiteti `“big_cars”` poimenovan kot `“brand_id”`.

4.2 Tehnologije na strani klienta

Dostop do spletnih mest je omogočen z uporabo brskalnikov, ki interpretirajo vhodne podatke, podane s strani uporabnika, in dostopajo do dokumentov, ki se nahajajo na spletnih strežnikih. Brskalnik predstavlja klienta v internetnem okolju, kajti brskalnik inicializira komunikacijo s spletnim strežnikom. Spletni strežnik pa samo čaka na zahteve, podane s strani klienta.

Pri razvoju spletnega mesta je potrebno veliko pozornosti posvetiti tehnologijam, ki so implementirane v brskalnikih. Te tehnologije omogočajo prikaz spletnega mesta in interakcijo uporabnika z njim. Podobno kot na strežnikih imamo tudi pri klientih skupek določenih tehnologij, ki ne samo da omogočajo komunikacijo uporabnika s spletnim mestom, ampak tudi

izboljšujejo uporabniško izkušnjo z njim.

Skupek tehnologij, ki jih bomo uporabljali pri gradnji aplikacije, je naslednji:

- HTML – Hype Text Markup Language,
- CSS – Cascading Style Sheets,
- JavaScript in
- Adobe Flash.

4.2.1 HTML

Jezik HTML opisuje vsebino dokumentov, do katerih dostopajo uporabniki. Osnovni namen jezika HTML je bil različen od namena drugih jezikov za oblikovanje besedil, ki zelo natančno določajo barvo, velikost in pisavo besedila. HTML je bil načrtovan za določanje strukture dokumenta na višjem in bolj abstraktnem nivoju, kajti vsebina se mora prikazati na različnih platformah in brskalnikih.

Trenutno sta v uporabi dve verziji jezika HTML. Prva verzija je HTML 4.01, ki je nadgradnja prejšnjih verzij jezika HTML. Druga verzija je XHTML 1.0, ki je redefinicija jezika HTML 4.01 z uporabo jezika XML (angl. *Extensible Markup Language*). XHTML 1.0 predstavlja tri standarde: strikten (angl. *Strict*), tranzicijski (angl. *Transitional*) in standard z okvirji (angl. *Frameset*). Strikten standard zahteva popolno skladnost z definicijo jezika XHTML 1.0. Tranzicijski standard dovoljuje uporabo prej opuščenih elementov jezika XHTML 1.0. Standard z okvirji dovoljuje uporabo okvirjev (angl. *frames*), čeprav so opuščeni [1].

Prednost uporabe HTML 4.01 pred uporabo XHTML 1.0 je lažje generiranje dokumentov, kajti HTML 4.01 nima tako striktnih sintakasnih pravil kot XHTML 1.0 – Strict. Po drugi strani ima standard XHTML 1.0 določene probleme na starejših brskalnikih. Kljub temu ima uporaba standarda XHTML 1.0 – Strict veliko prednosti. Striktne sintaksne pravila omogočajo generiranje kvalitetnih in konsistentnih XHTML dokumentov. Njihovo sintaksno pravilnost lahko preverimo z mnogimi orodji, ki jih najdemo na internetu, npr. <http://validator.w3.org> [1].

Vsak XHTML mora vsebovati deklaracijo `<?xml` elementa, ki identificira dokument kot dokument XML. Ta element določa verzijo standarda XML in kodiranje znakov v dokumentu (glej primer 4.3).

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

Primer 4.3: Deklaracija xml elementa

Deklaraciji xml elementa mora slediti ukaz DOCTYPE, ki določa specifičen tip dokumenta, kot je definiran v standardu SGML (angl. *Standard Generalized Markup Language*). Standard SGML določa notacijo za opisovanje jezikov, ki so namenjeni oblikovanju besedil. Tip dokumenta določa sintaksna pravila za posamezno skupino dokumentov XHTML (glej primer 4.4).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Primer 4.4: Ukaz DOCTYPE

XHTML mora vsebovati štiri osnovne oznake: <html>, <head>, <title> in <body>. Oznaka <html> določa korenski element dokumenta in preko atributa xmlns definira imenski prostor dokumenta (glej primer 4.5). Imenski prostor dokumenta XML določa zbirko elementov in atributov, ki jih dokument lahko vsebuje. Ime imenskega prostora je podano z uniformnim identifikatorjem resursov (URI – angl. *Uniform Resource Identifier*). Znotraj oznake <html> moramo definirati dve oznaki: <head> in <body>. Ti dve oznaki določata strukturo dokumenta. Oznaka <title> se nahaja znotraj oznake <head> in določa ime dokumenta. Celotna vsebina dokumenta se nahaja znotraj oznake <body> [1].

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

Primer 4.5: Imenski prostor dokumenta

4.2.2 CSS

Jezik XHTML določa strukturo dokumenta, ne pa obliko besedila znotraj dokumenta. Način in obliko prikaza posameznega elementa v brskalniku dosežemo z uporabo posebnih prezentacijskih specifikacij. Te specifikacije določimo v obliki, ki jo definira standard kaskadnih stilov (CSS – angl. *Cascading Style Sheets*). Pole kaskadnih stilov določajo, kako bo posamezen element dokumenta XHTML prikazan. Kaskadna lastnost izvira iz razloga, da je stil posameznega elementa lahko določen na treh nivojih. Lastnost, določena na najnižjem nivoju, lahko prepiše lastnost, ki je definirana na drugem ali tretjem nivoju. Podobno lastnost, definirana na drugem nivoju, prepiše lastnost, ki se nahaja na tretjem nivoju.

Trije nivoji kaskadnih pol so: nivo elementa, nivo dokumenta in zunanji nivo. Stili na nivoju elementa določajo obliko vsebine tega elementa. Stili na nivoju dokumenta vplivajo na obliko elementov skozi celoten dokument. Zunanje stile pa lahko uporabljamo v več dokumentih. Specifikacija stilov na nivoju elementa se pojavlja v deklaraciji tega elementa (glej primer 4.6).

Stile, ki definirajo obliko elementov skozi celoten dokument, vstavimo znotraj oznake `<head>` dokumenta XHTML (glej primer 4.7). Zunanji stili so določeni v datoteki, ki ni del dokumenta XHTML (glej primer 4.9). Referenco na zunanje stile vstavimo znotraj oznake `<head>` dokumenta XHTML (glej primer 4.8),

```
<p style="font-size: 16px;"> ... </p>
```

Primer 4.6: Stil na nivoju elementa `<p>`, ki določa velikost pisave

```
<style type="text/css">
  p { font-size: 16px; }
</style>
```

Primer 4.7: Stil na nivoju dokumenta, ki določa velikost pisave za vse element `<p>` znotraj dokumenta

```
<link rel="stylesheet" type="text/css" href="style.css"/>
```

Primer 4.8: Referenca na zunanjo datoteko "style.css", ki določa stile

```
p { font-size: 16px; }
```

Primer 4.9: Zunanji stil, ki je določen v datoteki "styles.css"

4.2.3 JavaScript

Programski jezik JavaScript se deli na tri dele: jedro, del na strani klienta in del na strani strežnika. Jedro je osrednji del jezika, ki vsebuje operatorje, izraze, stavke in podprograme. Del jezika na strani strežnika je zbirka objektov, ki omogočajo njegovo uporabo na strežnikih (npr. komunikacija s podatkovnimi bazami). Za izvedbo naše aplikacije pa bomo uporabili del jezika na strani klienta.

JavaScript na strani klienta (v nadaljevanju JavaScript) je skriptni jezik. V osnovi je objektno orientiran jezik, vendar se njegov objektni model bistveno razlikuje od objektnega modela v programskem jeziku Java ali C++. V JavaScript-u so vsi konstrukti funkcije. Tudi razrede definiramo z uporabo funkcij (glej primer 4.10). Vendar v tem primeru vsak ustvarjen objekt vsebuje kopije vseh lastnosti in metod. Če določenih metod ne želimo replicirati v vseh objektih, uporabimo ključno besedo *prototype* (glej primer 4.11). Tako dosežemo, da se metode in lastnosti prenašajo po referenci. Obstaja torej le ena instanca lastnosti in metode. Druga razlika v primerjavi z Javo je dinamičnost objektov. V Javi so objekti statični. Vse lastnosti in metode definiramo v času programiranja in jih med tekom programa ne moremo spreminjati. V JavaScript-u pa lahko objektom med izvajanjem programa spreminjamo in dodajamo lastnosti in

metode [3].

```
function Car(brand, model) {
    this.brand = brand;
    this.model = model;
    this.show = function() {
        document.write("Brand: " + this.brand + "<br />");
        document.write("Model: " + this.model);
    }
}
maserati = new Car("Maserati", "GranCabrio");
```

Primer 4.10: Deklariranje razreda in kreiranje objekta v JavaScript-u

```
function Car(brand, model) {
    this.brand = brand;
    this.model = model;
    Car.prototype.show = function() {
        document.write("Brand: " + this.brand + "<br />");
        document.write("Model: " + this.model);
    }
}
```

*Primer 4.11: Deklariranje metode razreda z uporabo ključne besede **prototype***

JavaScript se od Jave razlikuje tudi v tipizaciji jezika. Java je močno tipiziran jezik. Vsi tipi so znani med prevajanjem in lahko preverimo njihovo kompatibilnost. Spremenljivke v JavaScript-u pa niti ni potrebno deklarirati. Edina skupna točka med Javo in JavaScript-om je njuna sintaksa. Ime jezika JavaScript sicer implicira bližnje sorodstvo z jezikom Java, vendar temu ni tako.

Koda JavaScript je vključena v dokumentih XHTML in jo interpretira brskalnik. Vključitev kode v dokument XHTML lahko izvedemo na dva načina: implicitno ali eksplicitno. Eksplicitna vključitev pomeni, da je vsa koda vsebovana v dokumentu XHTML. Vključitev lahko izvedemo z uporabo posebnih atributov v oznakah XHTML (glej primer 4.12). Ti atributi dejansko predstavljajo dogodke, na katere se odziva brskalnik (npr. *onclick*, *onmouseover* itd.). Druga možnost eksplicitne vključitve je z uporabo oznake `<script>`, ki jo lahko uporabimo tako znotraj `<head>` kakor tudi `<body>` oznake (glej primer 4.13). Glavna slabost takšnega načina je prepletanje dveh popolnoma različnih programskih jezikov v enem dokumentu. Da odpravimo to pomanjkljivost, lahko JavaScript kodo zapišemo v ločeno datoteko. To predstavlja implicitno vključitev. Ločeno datoteko povežemo z dokumentom XHTML preko reference (glej primer 4.14) [1].

```
<div onclick="alert('Hello world!');"> ... </div>
```

Primer 4.12: Eksplicitna vključitev z uporabo atributa

```
<script type="text/javascript">
function showCar(car) {
    car.show();
}
</script>
```

Primer 4.13: Eksplicitna vključitev z uporabo oznake <script>

```
<script type="text/javascript" src="script.js"></script>
```

Primer 4.14: Implicitna vključitev datotoke "script.js"

Dejanska moč JavaScript-a se skriva v kontroli modela dokumentnega objekta (DOM – angl. *Document Object Model*). Ta objekt vsebuje posamezne dele dokumenta XHTML, ki so predstavljeni kot diskretni objekti v JavaScript-u. Vsak objekt predstavlja posamezno oznako v dokumentu, ki ji doda specifične lastnosti in metode. Vsi objekti so hierarhično urejeni. Ravno uporaba modela dokumentnega objekta pa predstavlja največji problem pri uporabi JavaScript-a.

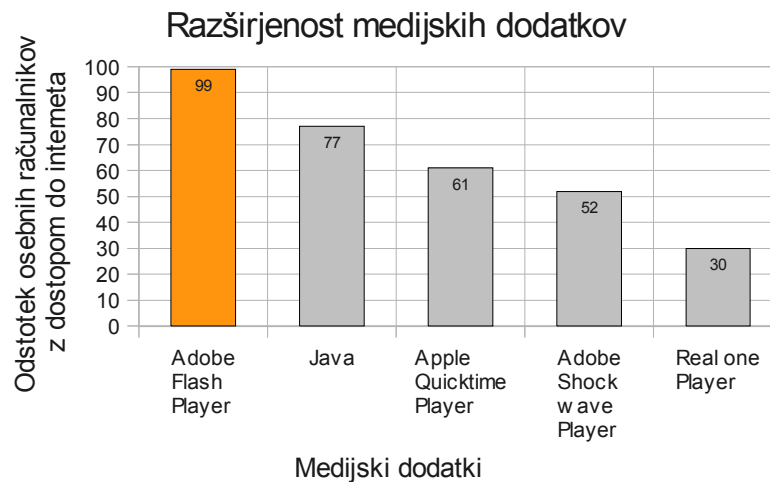
Vsak brskalnik je JavaScript implementiral na svoj način. Kljub majhnim razlikam je uporaba iste kode v vseh brskalnikih otežena. Isto kodo je potrebno napisati večkrat, pri tem pa upoštevati te majhne razlike med brskalniki. Da bi odpravili te pomanjkljivosti, so bile razvite posebne knjižnice, ki omogočajo uporabo iste kode v večini brskalnikov. Eno izmed teh knjižnic, in sicer jQuery, uporablja tudi naša aplikacija. Osnovni cilj knjižnice je spremeniti način, kako programerji razmišljajo o ustvarjanju funkcionalnosti na svojih spletnih straneh. Namesto da se ukvarjajo z zapletenostjo naprednega JavaScript-a, lahko z uporabo knjižnice jQuery izkoristijo obstoječe znanje stilov CSS, elementov dokumenta XHTML in osnovnega JavaScript-a ter tako drastično pohitrijo razvojni čas [7].

4.2.4 Adobe FLASH

Vse do sedaj našteje tehnologije omogočajo izdelovanje zanimivih in uporabnih spletnih strani in aplikacij. Vendar pa imajo še vedno določene pomanjkljivosti. Predvsem je težavno izdelati privlačne animacije, ki lahko zelo poživijo spletno stran. Do neke mere je to omogočeno s posebnimi knjižnicami JavaScript, ki imajo implementirane metode za lažje animiranje elementov spletnih strani. Poglavitno omejitev pa predstavlja predvajanje video in avdio datotek v okviru spletne strani. Nobena od zgornjih tehnologij nima definiranega načina, ki bi to omogočala. Tako je predvajanje videa ali glasbenih datotek problem platforme oziroma brskalnika, na katerem se spletna stran prikazuje. Težavo rešimo z uporabo posebne metode, ki je definirana v standardu XHTML (in tudi standardu HTML). Mehanizem omogoča vključitev

različnih medijev v dokument HTML, kot so npr. slike, videi, zvok in javanski programi (angl. *applet*). Brskalnik pa poskrbi za prikaz z uporabo zunanjih programov, t. i. dodatkov (angl. *add-on*). Vendar pa ta način ni konsistenten na vseh platformah in brskalnikih.

Problem rešuje posebna tehnologija, imenovana Adobe FLASH (na kratko Flash). Flash je posebna multimedijaska platforma, ki služi kreiranju animacij, predvajanju videa in dodajanju interaktivnosti. V brskalniku deluje kot dodatek, ki je navidezno del brskalnika. Tako lahko z uporabo istega dodatka prikazujemo veliko različnih medijskih datotek. Poglavitna prednost uporabe Flash tehnologije je njena razširjenost. Glede na analizo, izvedeno marca 2010, ima več kot 99 odstotkov uporabnikov interneta v brskalnikih vključen dodatek Flash (glej sliko 4.3) [8]. Dodatek, ki integrira Flash tehnologijo v brskalnike, se imenuje Adobe Flash Player.



Slika 4.3: Grafrazširjenosti multimedijskih dodatkov

Najpomembnejše komponente, ki sestavljajo platformo Flash, so:

- rokovalnik rasterskih slik,
- program za risanje vektorskih risb,
- program za animacijo na osnovi vektorjev,
- pogon za upravljanje z videom (in zvokom) in
- prevajalnik za programski jezik Actionscript.

Za potrebe naše aplikacije bomo uporabljali večinoma zadnji dve komponenti. Pogon za upravljanje z videom in zvokom bo skrbel za predvajanje videa. V jeziku Actionscript bomo

izdelali predvajalnik video vsebin, ki ga bomo prevedli s prevajalnikom.

4.2.4.1 Actionscript

Flash ima integrirano podporo za skriptni jezik Actionscript, ki ne omogoča samo lažjega kreiranja multimedijskih vsebin, ampak tudi ustvarjanje vsebin, ki ji sicer ne bi mogli. Na tem mestu predstavljamo tretjo, zadnjo različico jezika, kajti vse prejšnje različice nimajo popolne podpore razredom. Actionscript 3 (na kratko AS3) je objektno orientiran skriptni jezik, ki aplikacijam Flash dodaja interaktivnost. Za razvoj aplikacij z AS3 lahko uporabimo veliko različnih orodij. Na začetku je bil razvoj omogočen le z uporabo licenčnih orodij podjetja Adobe, ki je tudi lastnik jezika. Leta 2008 je podjetje Adobe v okviru odprtokodne licence izdalo ogrodje, s katerim je mogoče razvijati aplikacije brezplačno. Ogrodje z imenom Flex vsebuje knjižnice, ki nam omogočajo razvoj, in prevajalnik, s katerim aplikacijo pretvorimo v obliko, ki je primerna za vključitev v dokument HTML. Ta oblika je predstavljena v formatu SWF (angl. *Small Web Format* ali *ShockWave Flash*), ki je shramba za vse potrebne elemente, uporabljene pri izdelovanju aplikacije.

4.2.4.2 Flash video

Aplikacije Flash lahko vključujejo tudi video. Flash Live Video (na kratko FLV) je format, ki ga Flash uporablja za implementacijo videa. Načrtovan je bil za uporabo v spletnih straneh, zato omogoča zelo dobro kompresijo v primerjavi s kvaliteto prikazanega videa. FLV določa obliko datoteke, v kateri je shranjen video. Oblika definira, kateri podatki in kje v datoteki so shranjeni. Zraven videa in zvoka datoteka FLV vsebuje tudi meta podatke, ki opisujejo video. Meta podatke potrebuje dodatek v brskalnikih, ki izvajajo predvajanje videa. Med drugimi FLV vsebuje naslednje meta podatke:

- trajanje (v sekundah),
- velikost datoteke (v bajtih),
- število sličic na sekundo (angl. *framerate*),
- širino (v pikslih),
- višino (v pikslih),
- ime avdio kodeka (angl. *codec*),

- ime video kodeka [9].

Osnovna struktura v datoteki FLV je video in avdio. Format (verzija 10) predvideva 14 načinov predstavitve zvoka in 7 načinov predstavitve videa [9]. Način predstavitve definira kodek, ki ga uporabljamo za shranjevanje videa oziroma zvoka. Kodek je algoritem, ki določa kodiranje videa (ali zvoka) med shranjevanjem in odkodiranje videa (ali zvoka) med predvajanjem. Namen kodiranja videa je skrajšati velikost originalnega videa. Odkodiranje videa pa pomeni pretvorbo skrajšanega videa nazaj v obliko, ki je primerna za predvajanje.

Za naše potrebe bomo uporabili kodek On2 VP6 z alfa kanalom (na kratko VP6alpha). Ta kodek pri kodiranju uporablja metode, ki povzročijo izgubo informacij. Rezultat odkodiranja je zaradi tega različen od originalnega videa. Tako omogoča boljšo kompresijo, ob tem pa se poslabša kvaliteta. Poslabšanje je odvisno od parametrov, ki jih uporabimo za kompresijo. Ob pravilno izbranih parametrih je poslabšanje kvalitete človeškemu očesu skoraj neopazno. Poglavitni razlog izbire tega kodeka je lastnost, ki omogoča kodiranje in predvajanje transparentnega videa. Trenutno je VP6alpha edini kodek v družini Flash kodekov, ki omogoča to funkcionalnost.

Za kodiranje zvoka bomo uporabili MP3 kodek. To je najbolj razširjen standard za shranjevanje avdio datotek. Podobno kot VP6alpha je tudi algoritem MP3, ki pri kodiranju povzroči izgubo informacij.

4.2.4.3 Flash Player

Kodeki predstavljajo srž problema pri vključevanju videa v dokumente HTML. Ker standard HTML ne definira konkretnega kodeka, ki naj se uporablja za predvajanja videa, bi moral uporabnik poskrbeti za ustrezno konfiguracijo platforme. Ob upoštevanju dejstva, da je seznam kodekov relativno dolg, bi na takšen način preveč obremenili uporabnika. Rešitev predstavlja tehnologija Flash oziroma natančneje Flash Player.

Flash Player je dodatek v brskalnikih, ki omogoča izvajanje datotek SWF. Datoteka SWF je shramba. Vsebuje vse elemente, ki jih aplikacija potrebuje za izvajanje, npr. rasterske slike, vektorske risbe, video, zvok in kodo (Actionscript). Implementirane ima vse zgoraj omenjene video in avdio kodeke. Z vključitvijo enega dodatka v brskalnik uporabniki pridobijo konsistentno orodje za predvajanje videa in zvoka. Druga prednost razvijanja aplikacij Flash je množična uporaba predvajalnika Flash. S slike 4.3 je razvidno, da 99 odstotkov vseh brskalnikov, s katerimi uporabniki dostopajo do interneta, vključuje predvajalnik Flash.

Za naše potrebe je zelo pomembna še ena lastnost predvajalnika Flash, in sicer možnost

progresivnega nalaganja video in avdio datotek. Progresivno nalaganje omogoča začetek predvajanja videa (ali zvoka), preden se naloži celotna datoteka. Uporabnikom ni potrebno čakati, da se naloži celotna datoteka. Tako izboljšamo prijaznost aplikacije v odnosu do uporabnikov. Na drugi strani je progresivno nalaganje relativno enostavna in brezplačna rešitev v primerjavi z metodo toka podatkov (angl. *streaming*). Tok podatkov je naprednejša metoda progresivnega nalaganja. Omogoča sprotno prilagajanje količine prenešenih podatkov glede na zmožnosti odjemalca, tj. uporabnika. Uporabnik lahko med predvajanjem poljubno skače po časovnici videa, čeprav se določeni deli videa še niso prenesli. Vendar pa je za koriščenje vseh teh prednosti potrebno uporabiti (v večini primerov) drage licenčne strežnike. Prednosti, ki jih nudi metoda toka podatkov, so uporabne predvsem v sistemih, ki obsegajo veliko uporabnikov in videov, ki po dolžini trajanja presegajo 30 minut. V našem primeru bi bila takšna rešitev nepotrebna, kajti videi, ki jih bomo posredovali, so kratki (povprečna dolžina znaša 2 minuti). Predvajalnik pa ravno zaradi kratkega trajanja ne omogoča premikanja po časovnici videa.

Vzpostavitev sistema, ki omogoča progresivno nalaganje, je relativno enostavna. Potrebujemo samo spletni strežnik, ki klientom omogoča dostop do datotek. Večino dela opravi predvajalnik Flash, ki vee nalaga in jih predvaja. Pri progresivnem nalaganju se datoteka nalaga v predpomnilnik. Predvajalnik pa vrši predvajanje videa iz predpomnilnika. V primeru, ko se nalaganje videa zaradi napak na omrežju ali drugih razlogov ustavi, se video predvaja, dokler se predpomnilnik ne izprazni. Nato se predvajanje ustavi. Isto se zgodi, če je omrežje prepočasno in nalaganje ne sledi predvajanju videa. Vendar pa je število takih primerov relativno majhno, kajti trenutno večina uporabnikov za dostop do interneta uporablja širokopasovna omrežja.

5 Načrtovanje sistema

Načrtovanje sistema predstavlja najpomembnejši del razvoja aplikacije. Ob pravilnem in uspešnem načrtovanju je implementacija relativno enostavna. V primeru nenatančnega in površnega načrtovanja skoraj zagotovo naletimo na težave med izdelavo. Takšne težave pomenijo spreminjanje dela že izdelane aplikacije, kar za sabo povleče nekonsistentno aplikacijo in podaljša razvoj.

Izdelava naše aplikacije je opredeljena kot projekt. “Projekt je enkratna, praviloma zahtevna in kompleksna skupina nalog, ki mora biti končana v določenem roku, doseči mora vnaprej določene in morebitne kasneje odkrite cilje ter upoštevati vse podane in kasneje odkrite omejitve” [10]. Enkratnost projekta pomeni, da gre za vsebinsko in časovno enkratno nalogo. Zahtevnost projekta je opredeljena s kompleksno vsebino, kar pogojuje sodelovanje velike skupine različnih strokovnjakov. Projekt je razdeljen na večje število delov, ki so med seboj povezani in odvisni. Dela na projektu ne moremo začeti, preden ne odkrijemo strukture in povezave med elementi strukture. Izvedba projekta je skoraj vedno omejena s časom izdelave, ki je pogojen z aktualnostjo naloge in razpoložljivostjo virov. Za uspešno izvedbo projekta je nujno potrebno določiti cilje, ki jih želimo s projektom doseči. Določeni cilji omogočajo ocenjevanje napredka med izvajanjem projekta in predstavljajo merilo, s katerim lahko ob koncu projekta presodimo uspešnost projekta. Omejitve projekta pomenijo, da med izvajanjem projekta nimamo popolne svobode. Potrebno je upoštevati določena pravila, predpise, standardne, kulturo okolice itd. Poglavitne omejitve predstavljajo stroški za uresničitev projekta, razpoložljivi kadri ter programska in strojna oprema [10].

Izdelovanje naše aplikacije predstavlja projekt, s katerim je potrebno upravljati. Upravljanje oz. vodenja projekta predstavlja načrtovanje, organiziranje, vodenje in kontroliranje virov v specifičnem časovnem obdobju in z jasnimi enkratnimi cilji. Merilo uspešnosti pri vodenju projekta je opravljeno delo, ki je bilo določeno v okviru projektnih aktivnosti oz. nalog. Vodenje projekta je večplasten proces, kjer upravljamo z delom, časom, kadri, stroški, kvaliteto in komunikacijami. Potek vsakega projekta lahko razdelimo na tri faze:

1. faza zasnove (definicija ciljev projekta);
2. faza definiranja (določimo aktivnosti projekta);
3. faza izvajanja (izvajamo določene aktivnosti za dosego definiranih ciljev) [10].

Fazo zasnove smo opisali v poglavju 3. Faza definiranja je predmet tega poglavja, medtem ko je

faza izvajanja opredeljena v poglavju 6.

5.1 Programska arhitektura

V večini primerov je projekt kompleksna in zaključena celota. Za razumevanje kompleksnosti je potrebno celoto postopoma razbiti na manjše dele, t. i. podprobleme. Projekt razčlenimo na podprobleme, ki skupaj predstavljajo celoten projekt. Programska arhitektura prikazuje nadrejenost oz. podrejenost posameznih podproblemov projekta. Podproblemi so dejanske naloge, ki jih je potrebno izdelati. Izdelava posamezne naloge predstavlja rešitev tega podproblema. Da lahko izdelamo določeno nalogo, je včasih potrebno pred tem opraviti kakšno drugo nalogo, t. i. nadrejeno nalogo. Programska arhitektura torej predstavlja seznam nalog in njihovo medsebojno odvisnost.

5.2 Aktivnosti

Za uspešno načrtovanje moramo definirati naloge, ki jih je potrebno opraviti, da lahko dosežemo cilje projekta. Vsaka definirana naloga predstavlja aktivnost, ki vključuje določene vire in je časovno ovrednotena. Aktivnosti, ki smo jih definirali, bomo dodelili petim glavnim podciljem (angl. *milestone*) oz. podproblemom.

Glavni podcilji so izdelati:

- podatkovni model,
- administrativni del aplikacije,
- uporabniški del aplikacije,
- predvajalnik videov,
- skripto za vključitev predvajalnika v gostujočo spletno stran.

Ti podcilji so medsebojno odvisni. Da lahko izdelamo uporabniški del aplikacije, je najprej potrebno opraviti vse aktivnosti v okviru administrativnega dela aplikacije. Podobno velja za vse nadaljnje podcilje. Za vsak posamezni podcilj je potrebno določiti aktivnosti, s katerimi bomo dosegli ta določen podcilj.

Aktivnosti za doseg podcilja izdelave podatkovnega modela obsegajo:

- določitev entitet in njihovih atributov,
- določitev odvisnosti med entitetami,
- izdelavo konceptualnega podatkovnega modela,
- izdelavo logičnega podatkovnega modela,
- izdelavo fizičnega podatkovnega modela.

Aktivnosti za doseg podcilja izdelave administrativnega dela aplikacije obsegajo izdelavo:

- modula za avtentikacijo in avtorizacijo,
- modula za upravljanje uporabnikov in skupin,
- modula za upravljanje kategorij,
- modula za upravljanje videov,
- modula za upravljanje vsebinskih strani.

Aktivnosti za doseg podcilja izdelave uporabniškega dela aplikacije obsegajo izdelavo:

- dizajna in postavitve elementov spletnega mesta,
- modula za generiranje navigacije in pregledovanje vsebinskih strani,
- modula za pregledovanje kategorij in videov v posamezni kategoriji,
- modula za pregled posameznega videa in seznama nazadnje dodanih videov,
- modula za registracijo in prijavo uporabnikov ter urejanja profila uporabnika,
- modula za upravljanje z lokacijami,
- modula za upravljanje z izbranimi videi,
- modula za prikaz statistike predvajanj in klikov.

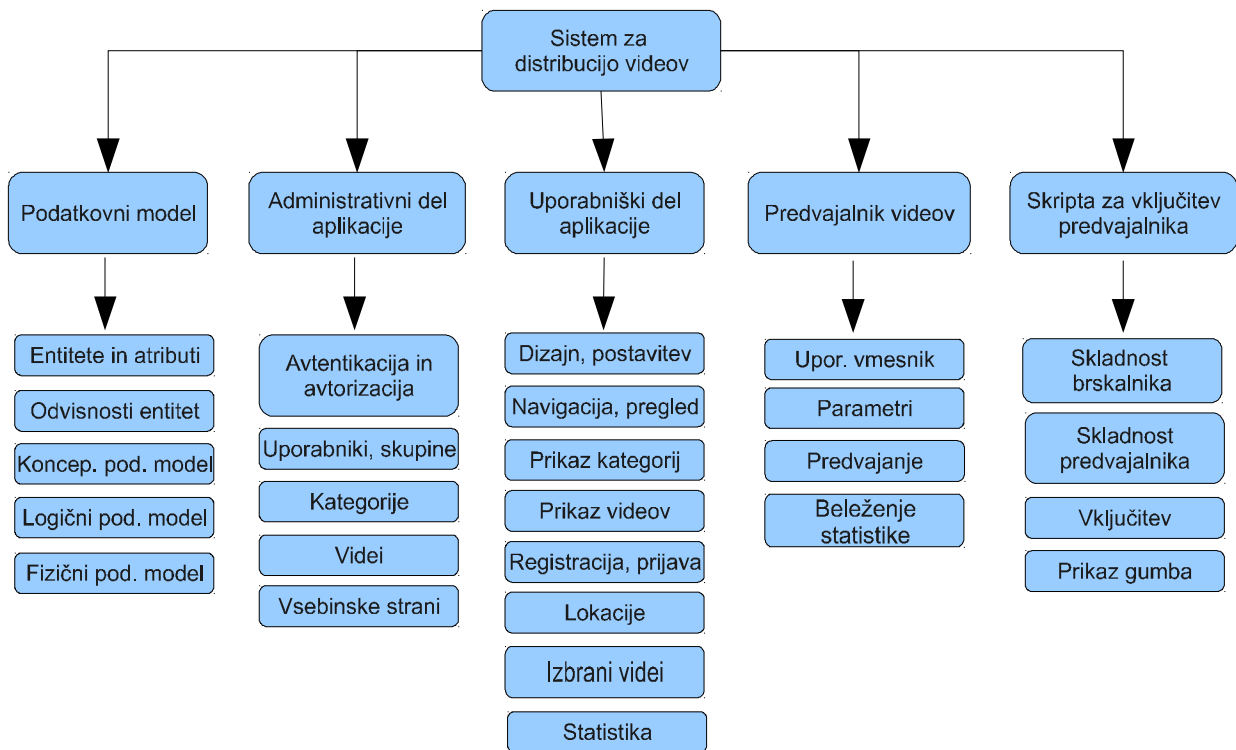
Aktivnosti za dosego podcilja izdelave predvajalnika videov obsegajo izdelavo:

- uporabniškega vmesnika,
- modula za branje parametrov,
- modula za predvajanje videov,
- modula za beleženje statistike predvajanj in klikov.

Aktivnosti za dosego podcilja izdelave skripte za vključitev predvajalnika obsegajo izdelavo:

- modula za preverjanje skladnosti brskalnika,
- modula za preverjanje skladnosti predvajalnika Flash,
- modula za vključitev predvajalnika,
- modula za vključitev gumba, ki omogoča ročni zagon predvajanja videa.

Projekt smo razdelili na aktivnosti z uporabo objektnega strukturiranja oz. razširjene objektno zasnovane metode (angl. *Work Breakdown Structure*). Objektno strukturiranje se izvaja glede na posamezni objekt dela. Objekt predstavlja posamezni podprojekt oz. podcilj, ki tvori sestavne dele ali objekte projekta. Strukturo, pridobljeno z zgornjo metodo, lahko predstavimo v obliki drevesa, kjer se v korenu drevesa nahaja celoten projekt; v naslednjem nivoju so prisotni glavni podcilji, ki skupaj tvorijo projekt. Rekurzivno izvajamo deljenje podciljev ali objektov projekta, dokler ne dosežemo skupin del, ki predstavljajo naravno zaključene delovne celote (glej sliko 5.1).

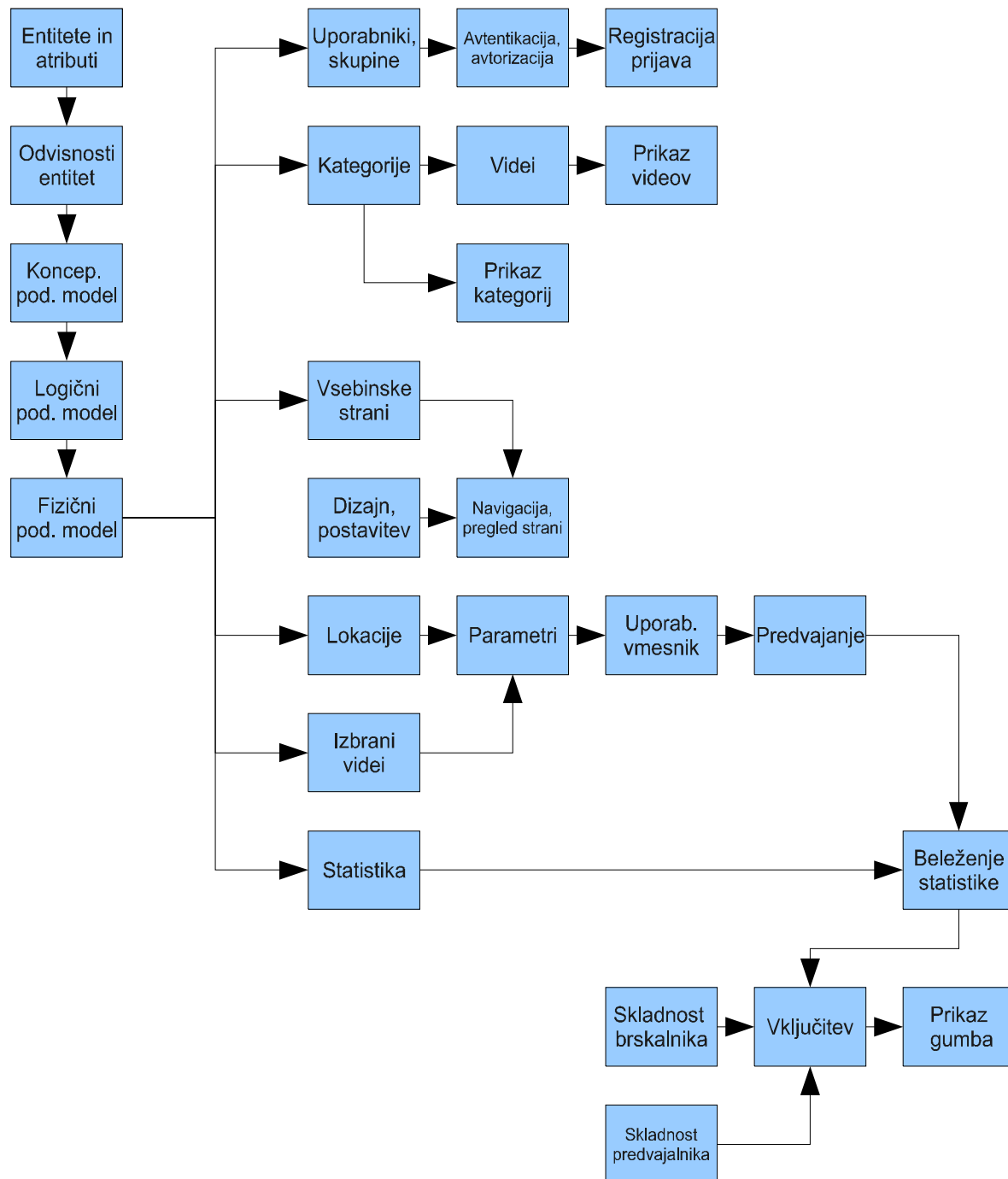


Slika 5.1: Drevesna struktura aktivnosti projekta

Izrednega pomena pri določanju aktivnosti je odkriti odvisnosti med aktivnostmi. Odvisnosti določajo povezanost aktivnosti, ki opredeljuje vrstni red izvajanja aktivnosti. Za izvajanje določenih aktivnosti je predtem potrebno izvesti aktivnosti, ki pogojujejo začetno stanje za izvedbo naslednje aktivnosti. Za odkrivanje povezanosti aktivnosti se uporablja tehnika petih vprašanj. Za vsako aktivnost moramo odgovoriti na naslednjih pet vprašanj:

1. Ali je opazovana aktivnost začetna aktivnost?
2. Ali je opazovana aktivnost zaključna aktivnost?
3. Katere aktivnosti se morajo končati neposredno pred opazovano aktivnostjo?
4. Katere aktivnosti se morajo pričeti tik po zaključku opazovane aktivnosti?
5. Katere aktivnosti lahko potekajo vzporedno opazovani aktivnosti?

Odgovori na zgornja vprašanja nam povedo vrstni red izvajanja aktivnosti. Ugotovimo, s katerimi aktivnostmi je potrebno začeti razvoj aplikacije in katere aktivnosti se lahko izvajajo vzporedno. Za lažji pregled vrstnega reda aktivnosti se uporablja predstavitev s t. i. aktivnostno mrežo. Aktivnosti so v mreži predstavljene v vozliščih grafa, povezave med vozlišči pa predstavljajo vrstni red (glej sliko 5.2).



Slika 5.2: Aktivnostna mreža projekta

Ob predstavitvi aktivnosti in njihovega vrstnega reda se aktivnostne mreže uporabljajo tudi za določanje trajanja celotnega projekta. Za izračun časa trajanja celotnega projekta je potrebno vedeti čas trajanja posamezne aktivnosti. Natančno določanje trajanja aktivnosti je v večini primerov težko in včasih celo nemogoče opravilo. Kvaliteta ocene časa je odvisna od vrste projekta, aktivnosti in izkušenj načrtovalca. Vendar pa obstaja metoda za določitev trajanja

projekta, ki za uspešno planiranje ne potrebuje natančnih časovnih ocen aktivnosti. To je metoda PERT (angl. *Program Evolution and Review Technique*), ki za ocenitev trajanja aktivnosti uporablja tri ocene časa. Določiti je potrebno optimistični čas (10 % verjetnost za izpolnitev), najbolj verjetni čas in pesimistični (10 % verjetnost za izpolnitev) čas. Po določitvi teh časovnih ocen lahko s časovno analizo aktivnostne mreže neposredno izračunamo rok zaključka projekta [10]. Za naš projekt se bomo izognili časovni analizi, kajti projekt nima določene časovne omejitve.

5.3 Modeli primerov uporabe

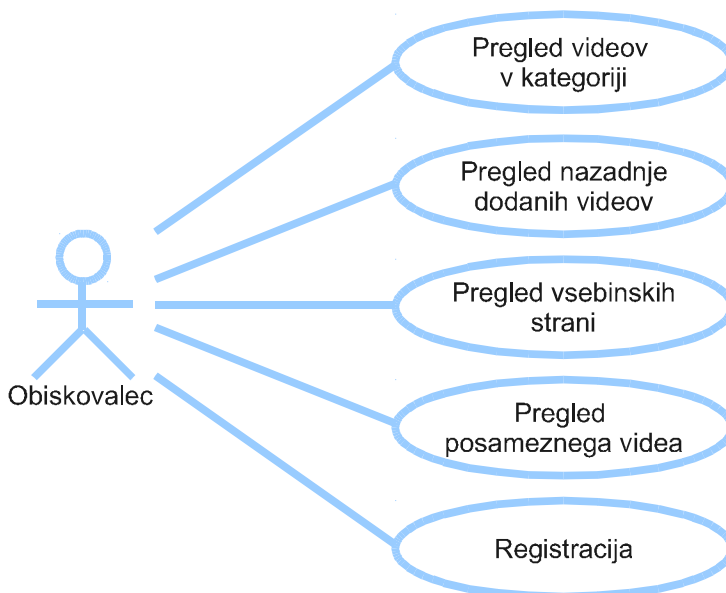
Modeli primerov uporabe podajajo strukturiran pogled na zahtevano obnašanje aplikacije. Definirajo, “kdo” lahko počne “kaj”. Aplikacija definira vmesnik, preko katerega uporabniki izvajajo akcije v aplikaciji. Vsak model primera uporabe opisuje interakcijo uporabnika z aplikacijo. Namen interakcije je izpolnitev določenega cilja ali naloge. Glede na podane funkcionalnosti, ki jih aplikacija mora implementirati, začnemo graditi modele primerov uporabe. Vsak model primera uporabe lahko opisuje obnašanje več funkcionalnosti. Podobno lahko določeno funkcionalnost opišemo z več modeli primerov uporabe. Rezultat modeliranja primerov uporabe je boljše razumevanje obnašanja aplikacije in funkcionalnosti, ki jih je potrebno implementirati.

Modeli primerov uporabe obravnavajo aplikacijo kot “črno škatlo”. Interakcije z aplikacijo so predstavljene kot akcije, ki prihajajo iz zunanjega sveta. Takšen način razmišljanja prisili načrtovalca, da razmišlja o tem, kaj mora aplikacija izvajati, ne pa kako bo to izvajala. Za predstavitev modelov primerov uporabe se uporablja standardiziran jezik UML (angl. *Unified Modeling Language*), ki določa množico grafičnih notacij. Tehnika grafičnih notacij omogoča vizualno predstavitev modelov primerov uporabe. Ob vizualni predstavitvi je potrebno modelom dodati opise naslednjih elementov:

- ime,
- opis,
- tok dogodkov (osnovni in alternativni),
- zahteve,
- predpogoji,

- popogoji.

Ime in opis določata model primera uporabe. S ciljem podamo namen posameznega modela. Poglavitni element je tok dogodkov. V večini primerov opišemo dva toka dogodkov: prvi opisuje normalno interakcijo uporabnika z aplikacijo, drugi pa predstavi izjemne dogodke. Preden se lahko izvedejo akcije, ki jih opisuje model, morajo biti izpolnjene določene zahteve in predpogoji. Popogoji določajo stanje aplikacije po izvedbi akcij.



Slika 5.3: Primeri uporabe za obiskovalce spletne strani

V okviru spodnjih primerov uporabe predstavljamo 3 različne tipe uporabnikov. Prvi je obiskovalec, ki predstavlja navadnega uporabnika spletnega mesta, kar pomeni, da uporabnik ni niti registriran niti prijavljen v aplikacijo. Drugi je uporabnik, ki predstavlja registriranega in prijavljenega uporabnika. Tretji je urednik, ki predstavlja uporabnika s posebnimi pravicami. Te pravice omogočajo upravljanje s celotno aplikacijo.

5.3.1 Pregled videov v kategoriji

1. Opis

Primer prikazuje, kako obiskovalci pregledujejo videe v posamezni kategoriji.

2. Tokovi dogodkov

1. Osnovni tok

Obiskovalcu je prikazana hierarhija kategorij. S klikom izbere želeno kategorijo. Nato se prikažejo vsi videi, ki se nahajajo v tej kategoriji. Ob posameznem videu se prikaže sličica (predogled), trajanje videa in jezik videa. S klikom na sličico lahko obiskovalec predvaja video.

Seznam videov se prikaže na večih straneh. Na posamezni strani se prikaže največ 6 videov. Obiskovalec se lahko pomika po teh straneh, tako da:

1. klikne na povezavo, ki prikaže naslednjo stran ali
 2. klikne na povezavo, ki prikaže prejšnjo stran.
2. Alternativni tok

Če v izbrani kategoriji ni videov, se prikaže sporočilo, ki obiskovalca o tem obvesti.

3. Zahteve

V aplikaciji morajo biti definirane kategorije.

4. Predpogoji

Jih ni.

5. Popogoji

Jih ni.

5.3.2 Pregled nazadnje dodanih videov

1. Opis

Obiskovalci si lahko ogledajo nazadnje dodane videe.

2. Tokovi dogodkov

1. Osnovni tok

Ob kliku na povezavo (npr. "Novi videi") se prikaže seznam šestih nazadnje dodanih videov.

2. Alternativni tok

Ga ni.

3. Zahteve

Aplikacija mora vsebovati videe.

4. Predpogoji

Jih ni.

5. Popogoji

Jih ni.

5.3.3 Pregled vsebinskih strani

1. Opis

Obiskovalci imajo na razpolago navigacijo, s pomočjo katere dostopajo do vsebinskih strani. Vsebinske strani vsebujejo dodatne opise o aplikaciji.

2. Tokovi dogodkov

1. Osnovni tok

Obiskovalec klikne na povezavo, prikazano v navigaciji. Prikaže se vsebinska stran.

2. Alternativni tok

Ga ni.

3. Zahteve

V aplikaciji morajo biti določene vsebinske strani.

4. Predpogoji

Jih ni.

5. Popogoji

Jih ni.

5.3.4 Pregled posameznega videa (obiskovalci)

1. Opis

Prikažejo se dodatne lastnosti videa.

2. Tokovi dogodkov

1. Osnovni tok

Ob kliku na videe, prikazane v kategorijah in na seznamu nazadnje dodanih, se obiskovalcu prikažejo dodatne lastnosti videa. In sicer: sličica, trajanje, višina, širina, jezik in velikost datoteke. Ob kliku na sličico lahko obiskovalec predvaja video.

2. Alternativni tok

Ga ni.

3. Zahteve

Aplikacija mora vsebovati videe.

4. Predpogoji

Jih ni.

5. Popogoji

Jih ni.

5.3.5 Registracija

1. Opis

Obiskovalec se lahko registrira kot uporabnik aplikacije.

2. Tokovi dogodkov

1. Osnovni tok

Obiskovalcu se prikaže obrazec, s pomočjo katerega se lahko registrira v aplikacijo. Podatki, ki jih mora posredovati, so sledeči: uporabniško ime, geslo, e-naslov, ime in

priimek. Preden obiskovalca registriramo, je potrebno posredovane podatke validirati. Nato obiskovalca registrira in prijavi v sistem.

2. Alternativni tok

Če je obiskovalec že prijavljen v aplikacijo, se ga preusmeri na vstopno stran in o tem obvesti.

V primeru, da posredovane podatke ni mogoče validirati, se o neuspehu obvesti obiskovalca. Validacija obsega preverjanje, ali uporabniško ime in e-naslov že obstajata. Geslo mora vsebovati najmanj 6 znakov. Obvezno je potrebno posredovati tudi ime in priimek.

3. Zahteve

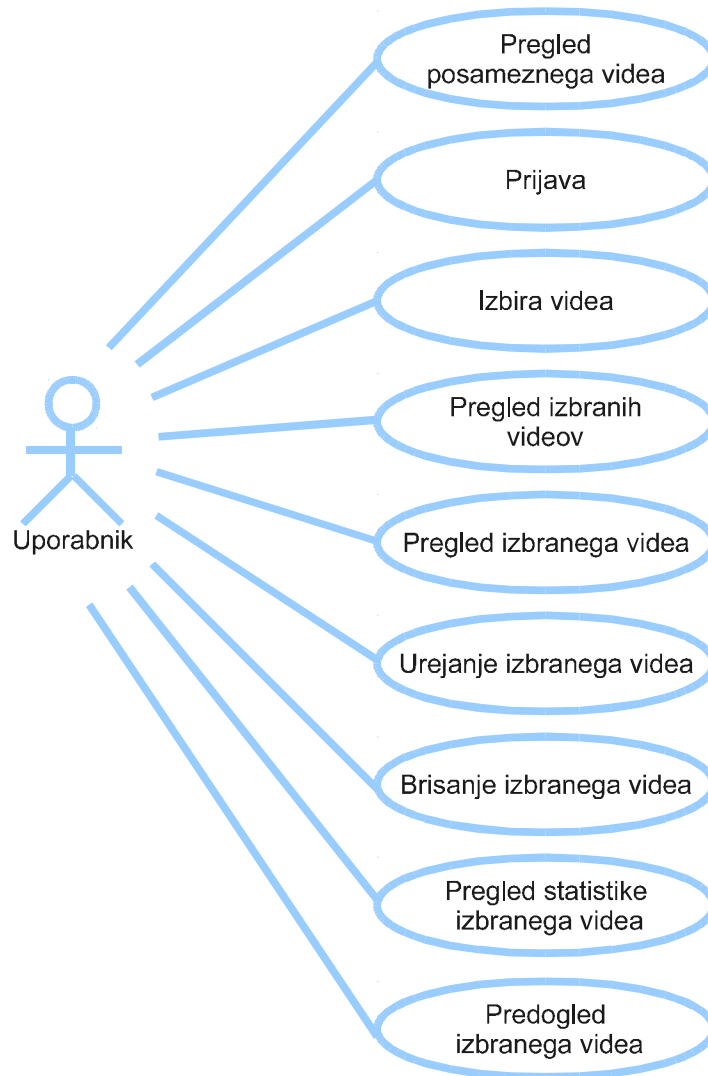
Jih ni.

4. Predpogoji

Obiskovalec ni prijavljen v sistem.

5. Popogoji

Uporabnik je registriran in prijavljen v sistem.



Slika 5.4: Primeri uporabe za uporabnika (prijavljenega obiskovalca)

5.3.6 Pregled posameznega videa (uporabniki)

1. Opis

Prikažejo se dodatne lastnosti videa.

2. Tokovi dogodkov

1. Osnovni tok

Ob kliku na videe, prikazane v kategorijah in na seznamu nazadnje dodanih, se uporabniku prikažejo dodatne lastnosti videa. In sicer: sličica, trajanje, višina, širina, jezik in velikost datoteke. Ob kliku na sličico lahko uporabnik predvaja video.

Dodatno se uporabniku prikaže povezava, s katero lahko izbere video, ki ga bo potencialno vključil v svojo spletno stran.

2. Alternativni tok

Ga ni.

3. Zahteve

Aplikacija mora vsebovati vee.

4. Predpogoji

Uporabnik mora biti prijavljen v aplikacijo.

5. Popogoji

Jih ni.

5.3.7 Prijava

1. Opis

Uporabnikom omogočamo prijavo v aplikacijo.

2. Tokovi dogodkov

1. Osnovni tok

Uporabniku se prikaže obrazec, s pomočjo katerega se lahko prijavi. Posredovati mora uporabniško ime in geslo. Aplikacija ga nato prijavi.

2. Alternativni tok

V primeru, da posredovani podatki ne omogočajo prijave, se o tem obvesti uporabnika.

3. Zahteve

Jih ni.

4. Predpogoji

Uporabnik mora biti registriran v aplikaciji.

5. Popogoji

Uporabnik je prijavljen v sistem.

5.3.8 Izbira videa

1. Opis

Uporabniki si izberejo video, ki ga želijo objaviti na svoji spletni strani. Po izbiri določijo parametre predvajanja.

2. Tokovi dogodkov

1. Osnovni tok

Uporabnik si video izbere s klikom na gumb “Izberi video”, prikazan na strani, ki predstavlja posamezni video. Po izbiri se prikaže obrazec, kjer uporabnik določi parametre. Ti parametri so sledeči:

- lokacija videa (možna izbira obstoječe lokacije ali dodajanje nove),
- pozicija videa v okviru gostujoče spletne strani (levo zgoraj, desno zgoraj, na sredini, levo spodaj, na sredini spodaj, desno spodaj),
- način pozicioniranja (plavajoč ali fiksiran),
- frekvenca predvajanja (kolikokrat zaporedemo se video obiskovalcu gostujoče spletne strani ne prikaže),
- veljavnost predvajanja (od določenega datuma do določenega datuma),
- širina in višina (možnost prostega spreminjanja ali spreminjaja, ki ohranja razmerje med višino in širino),
- barve predvajalnika (aktivna, neaktivna, rob in ozadje) in
- URL, kamor se obiskovalca gostujoče spletne strani preusmeri ob kliku na video.

Določitev parametrov uporabnik potrdi s klikom na gumb “Shrani”.

2. Alternativni tok

Pred shranjevanjem je potrebno vrednosti parametrov validirati. V primeru, da vrednosti niso pravilne, se o tem obvesti uporabnika. V obrazcu se prikažejo nepravilne vrednosti parametrov in razlog za njihovo nepravilnost.

3. Zahteve

Jih ni.

4. Predpogoji

Uporabnik mora biti prijavljen v aplikaciji in si mora izbrati video.

5. Popogoji

Uporabnik je dodal nov video na seznam svojih izbranih videov.

5.3.9 Pregled izbranih videov

1. Opis

Uporabnikom prikažemo seznam vseh njegovih izbranih videov.

2. Tokovi dogodkov

1. Osnovni tok

Dostop do seznama je omogočen preko klika na povezavo v navigaciji, ki je prikazana v okviru uporabniškega računa. Seznam prikazuje naziv videa, lokacijo, frekvenco in veljavnost videa. V primeru, da ima uporabnik več kot 20 izbranih videov, se prikaz vseh razdeli na več strani. Uporabnik lahko z uporabo povezav “naslednja stran” ali “prejšnja stran” pregleduje vse strani.

Za vsak posamezni video na seznamu so na voljo naslednje akcije:

- pregledovanje,
- urejanje,

- brisanje in
- pregledovanje statistike.

2. Alternativni tok

V primeru, da uporabnik nima izbranih videov, se prikaže obvestilo.

3. Zahteve

Jih ni.

4. Predpogoji

Uporabnik mora biti prijavljen v aplikaciji.

5. Popogoji

Jih ni.

5.3.10 Pregled izbranega videa

1. Opis

Uporabnikom prikažemo izbran video in vrednosti določenih parametrov.

2. Tokovi dogodkov

1. Osnovni tok

Po kliku na povezavo s seznama izbranih videov uporabniku prikažemo izbrani video. Prikažejo se podatki o videu (jezik, trajanje, širina, višina in velikost datoteke) in vrednosti določenih parametrov, ki jih je uporabnik določil ob izbiri le-tega. Ob teh parametrih uporabniku prikažemo še kodo, ki jo uporabnik uporabi za vključitev videa v svojo spletno stran. Prikaže se tudi gumb, ki omogoča predogled vključitve videa v gostujočo spletno stran.

2. Alternativni tok

V primeru, da izbrani video ne obstaja, se prikaže obvestilo.

3. Zahteve

Izbrani video mora obstajati.

4. Predpogoji

Uporabnik mora biti prijavljen v aplikaciji.

5. Popogoji

Jih ni.

5.3.11 Urejanje izbranega videa

1. Opis

Uporabnikom omogočimo urejanje izbranega videa.

2. Tokovi dogodkov

1. Osnovni tok

Po kliku na povezavo s seznama izbranih videov uporabniku prikažemo obrazec za urejanje parametrov izbranega videa. Obrazec je enak kot pri 5.3.8 z razliko, da so vsi parametri nastavljeni na vrednosti, ki jih je določil uporabnik. Spreminjaje vrednosti deluje na enak princip kot pri 5.3.8.

2. Alternativni tok

V primeru, da izbrani video ne obstaja, se prikaže obvestilo.

3. Zahteve

Izbrani video mora obstajati.

4. Predpogoji

Uporabnik mora biti prijavljen v aplikaciji.

5. Popogoji

Vrednosti parametrov so spremenjene.

5.3.12 Brisanje izbranega videa

1. Opis

Uporabnikom omogočimo brisanje izbranega videa.

2. Tokovi dogodkov

1. Osnovni tok

Po kliku na povezavo s seznama izbranih videov uporabniku prikažemo sporočilo, ki ga sprašuje, če je prepričan, da želi izbrisati izbrani video. Po potrditvi se izbrani video izbriše. Izbriše se tudi statistika zanj.

2. Alternativni tok

Če uporabnik ne potrdi brisanja, umaknemo sporočilo in ničesar ne zberemo.

3. Zahteve

Izbrani video mora obstajati.

4. Predpogoji

Uporabnik mora biti prijavljen v aplikaciji.

5. Popogoji

Izbrani video je izbrisan.

5.3.13 Pregled statistike izbranega videa

1. Opis

Uporabnikom omogočimo pregledovanje statistike za izbrani video.

2. Tokovi dogodkov

1. Osnovni tok

Po kliku na povezavo s seznama izbranih videov, uporabniku prikažemo statistiko.

Prikažemo dva sklopa kumulativnih podatkov, in sicer seštevek vseh prikazov in klikov za celotno obdobje predvajanja videa ter seštevek vseh prikazov in klikov za obdobje zadnjih 30 dni. Ob tem se izrišeta dva grafa. Prvi prikazuje število prikazov videa po dnevih za zadnjih 30 dni. Drugi pa prikazuje število klikov na video po dnevih za zadnjih 30 dni.

Uporabnik ima tudi možnost spreminjaja parametrov prikaza statistike. Nastavi lahko naslednje parametre:

- vrsta prikaza (prikazi, kliki, oboje),
- natančnost (po urah, dnevih ali mesecih),
- od datuma,
- do datuma.

Po določitvi parametrov se uporabniku prikaže nova statistika, ki upošteva zgoraj določene parametre. Prikaže se kumulativna statistika in oba grafa za izbrano obdobje.

2. Alternativni tok

Če je kumulativna statistika prikazov ali klikov za izbrano obdobje enaka 0, se graf prikazov ali klikov ne prikaže.

3. Zahteve

Izbrani video mora obstajati.

4. Predpogoji

Uporabnik mora biti prijavljen v aplikaciji.

5. Popogoji

Jih ni.

5.3.14 Predogled izbranega videa

1. Opis

Uporabnikom omogočimo predogled izbranega videa v gostujoči spletni strani.

2. Tokovi dogodkov

1. Osnovni tok

Po kliku gumba “Predogled” v pregledu izbranega videa uporabniku odpremo njegovo spletno stran, v katero vključimo izbrani video. Pri vključitvi upoštevamo vse določene vrednosti parametrov.

2. Alternativni tok

Če gostujoča spletna stran ne obstaja, o tem obvestimo uporabnika.

3. Zahteve

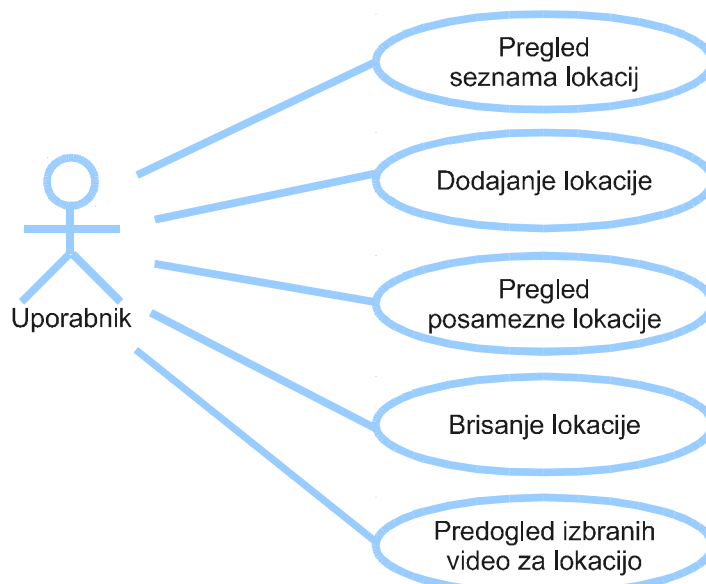
Izbrani video mora obstajati.

4. Predpogoji

Uporabnik mora biti prijavljen v aplikaciji.

5. Popogoji

Jih ni.



Slika 5.5: Primeri uporabe za uporabnika glede na lokacije

5.3.15 Pregled seznama lokacij

1. Opis

Uporabnikom prikažemo seznam vseh lokacij.

2. Tokovi dogodkov

1. Osnovni tok

Dostop do seznama je omogočen preko klika na povezavo v navigaciji, ki je prikazana v okviru uporabniškega računa. Seznam prikazuje naziv lokacije, URL naslov in datum kreiranja. V primeru, da ima uporabnik več kot 20 lokacij, se prikaz vseh razdeli na več strani. Uporabnik lahko z uporabo povezav “naslednja stran” ali “prejšnja stran” pregleduje vse strani.

Za vsak posamezni video na seznamu sta na voljo naslednji akciji:

- pregledovanje in
- brisanje.

2. Alternativni tok

V primeru, da uporabnik nima lokacij, se prikaže obvestilo.

3. Zahteve

Jih ni.

4. Predpogoji

Uporabnik mora biti prijavljen v aplikaciji.

5. Popogoji

Jih ni.

5.3.16 Dodajanje lokacije

1. Opis

Uporabnikom omogočimo dodajanje nove lokacije.

2. Tokovi dogodkov

1. Osnovni tok

Uporabniku se prikaže obrazec, preko katerega doda novo lokacijo. Posredovati mora naslednje podatke: naziv lokacije in URL naslov lokacije. Pred shranjevanjem nove lokacije je potrebno validirati posredovane podatke.

2. Alternativni tok

Če posredovani podatki niso uspešno validirani, se o tem obvesti uporabnika.

3. Zahteve

Jih ni.

4. Predpogoji

Uporabnik mora biti prijavljen v aplikaciji.

5. Popogoji

Lokacija je dodana.

5.3.17 Pregled posamezne lokacije

1. Opis

Uporabnikom prikažemo posamezno lokacijo.

2. Tokovi dogodkov

1. Osnovni tok

Po kliku na povezavo s seznama lokacij uporabniku prikažemo lokacijo. Prikažejo se podatki o lokaciji (naziv, URL, datum kreiranja). Uporabniku prikažemo še kodo, ki jo uporabnik uporabi za vključitev vseh videov za lokacijo v svojo spletno stran. Prikaže se tudi gumb, ki omogoča predogled vključitve videov v gostujočo spletno stran.

Prikaže se tudi seznam videov, ki so povezani z lokacijo. Na seznamu videov prikažemo naslednje podatke: naziv videa, frekvenco predvajanja, veljavnost predvajanja in URL povezavo. Ob tem so za posamezni video na seznamu na voljo naslednje akcije: pregled, urejanje, brisanje in pregled statistike. V okviru tega seznama je uporabniku na voljo urejanje vrstnega reda videov. S spreminjanjem vrstnega reda uporabnik določa vrstni red predvajanja videov glede na to lokacijo.

2. Alternativni tok

V primeru, da izbrani video ne obstaja, se prikaže obvestilo. Če lokacija ne vsebuje povezanih videov, se o tem obvesti uporabnika.

3. Zahteve

Lokacija mora obstajati.

4. Predpogoji

Uporabnik mora biti prijavljen v aplikaciji.

5. Popogoji

Če je uporabnik spremenil vrstni red videov, se spremenjeni vrstni red shrani.

5.3.18 Brisanje lokacije

1. Opis

Uporabnikom omogočimo brisanje lokacije.

2. Tokovi dogodkov

1. Osnovni tok

Po kliku na povezavo s seznama lokacij uporabniku prikažemo sporočilo, ki ga sprašuje, če je prepričan, da želi izbrisati lokacijo. Po potrditvi se lokacija izbriše. Izbrišejo se tudi vse povezave na videe za to lokacijo.

2. Alternativni tok

Če uporabnik ne potrdi brisanja, umaknemo sporočilo in ničesar ne zbrišemo.

3. Zahteve

Izbrani video mora obstajati.

4. Predpogoji

Uporabnik mora biti prijavljen v aplikaciji.

5. Popogoji

Lokacija je izbrisana.

5.3.19 Predogled izbranih videov za lokacijo

1. Opis

Uporabnikom omogočimo predogled izbranih videov za posamezno lokacijo v gostujoči spletni strani.

2. Tokovi dogodkov

1. Osnovni tok

Po kliku gumba "Predogled" v pregledu lokacije uporabniku odpremo njegovo spletno

stran, v katero vključimo izbrane videe. Pri vključitvi upoštevamo vse določene vrednosti parametrov izbranih videov in vrstni red predvajanja, določenega za to lokacijo.

2. Alternativni tok

Če gostujoča spletna stran ne obstaja, o tem obvestimo uporabnika.

3. Zahteve

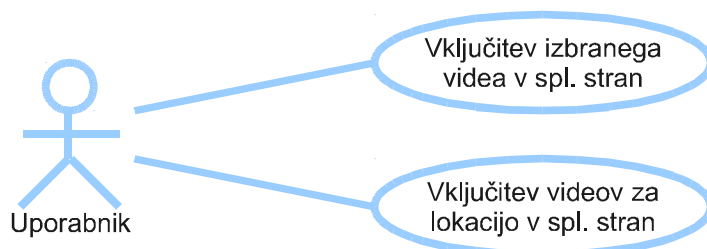
Lokacija mora obstajati.

4. Predpogoji

Uporabnik mora biti prijavljen v aplikaciji.

1. Popogoji

Jih ni.



Slika 5.6: Primeri uporabe za uporabnika glede vključitve videov v spletno stran

5.3.20 Vključitev izbranega videa v spletno stran

1. Opis

Uporabniki izbrane videe vključujejo v svoje spletne strani.

2. Tokovi dogodkov

1. Osnovni tok

Vključitev je omogočena z uporabo kode, ki jo uporabniki pridobijo pri pregledu izbranega videa (5.3.10). To kodo dodajo na konec XHTML dokumenta, ki predstavlja njihovo spletno stran.

2. Alternativni tok

Ga ni.

3. Zahteve

Izbrani video mora obstajati. Gostujoča spletna stran mora obstajati.

4. Predpogoji

Jih ni.

5. Popogoji

Jih ni.

5.3.21 Vključitev izbranih videov za lokacijo v spletno stran

1. Opis

Uporabniki izbrane videe v posamezni lokaciji vključujejo v svoje spletne strani.

2. Tokovi dogodkov

1. Osnovni tok

Vključitev je omogočena z uporabo kode, ki jo uporabniki pridobijo pri pregledu lokacije (5.3.17). To kodo dodajo na konec XHTML dokumenta, ki predstavlja njihovo spletno stran.

2. Alternativni tok

Ga ni.

3. Zahteve

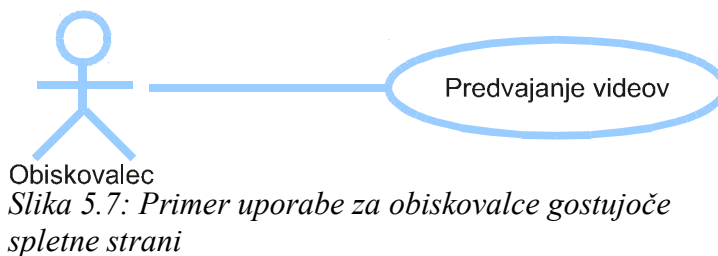
Lokacija mora obstajati. Gostujoča spletna stran mora obstajati.

4. Predpogoji

Jih ni.

5. Popogoji

Video je vključen v spletno stran.



5.3.22 Predvajanje videov

1. Opis

Obiskovalci gostujočih spletnih strani predvajajo videe.

2. Tokovi dogodkov

1. Osnovni tok

Ko obiskovalec obiše gostujočo spletno stran, se izbrani video začne predvajati, če to omogočajo vrednosti parametrov.

2. Alternativni tok

Če vrednosti parametrov ne omogočajo predvajanja, se prikaže gumb, ki obiskovalcu omogoča ročni zagon predvajanja.

3. Zahteve

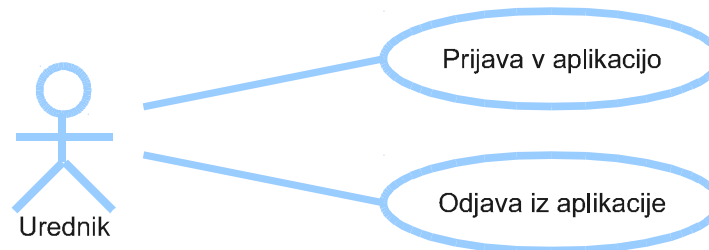
Gostujoča spletna stran mora obstajati.

4. Predpogoji

Določajo jih parametri, ki jih je določil uporabnik.

5. Popogoji

Video se predvaja.



Slika 5.8: Primera uporabe za urednika (prijava in odjava)

5.3.23 Prijava (urednik)

1. Opis

Urednikom omogočamo prijavo v aplikacijo.

2. Tokovi dogodkov

1. Osnovni tok

Uredniku se prikaže obrazec, s pomočjo katerega se lahko prijavi. Posredovati mora uporabniško ime in geslo. Aplikacija ga nato prijavi.

2. Alternativni tok

V primeru, da posredovani podatki ne omogočajo prijave, se o tem obvesti urednika.

3. Zahteve

Jih ni.

4. Predpogoji

Urednik mora biti registriran v aplikaciji.

5. Popogoji

Urednik je prijavljen v sistem.

5.3.24 Odjava (urednik)

1. Opis

Urednikom omogočamo odjavo iz aplikacije.

2. Tokovi dogodkov

1. Osnovni tok

Urednik klikne na povezavo, ki omogoča odjavo. Aplikacija ga odjavi.

2. Alternativni tok

Ga ni.

3. Zahteve

Jih ni.

4. Predpogoji

Uporabnik mora biti prijavljen v aplikaciji.

5. Popogoji

Uporabnik je odjavljen iz sistema.



Slika 5.9: Primeri uporabe za urednika v okviru kategorij

5.3.25 Pregled seznama kategorij

1. Opis

Urednikom prikažemo seznam vseh kategorij.

2. Tokovi dogodkov

1. Osnovni tok

Dostop do seznama je omogočen preko klika na povezavo v navigaciji, ki je prikazana v okviru uredniškega dela aplikacije. Seznam prikazuje naziv kategorije in naziv nadrejene kategorije. V primeru, da je v aplikaciji več kot 20 kategorij, se prikaz vseh razdeli na več strani. Urednik lahko z uporabo povezav “naslednja stran” ali “prejšnja stran” pregleduje vse strani.

Za vsako posamezno kategorijo na seznamu so na voljo naslednje akcije:

- pregledovanje,
- urejanje in
- brisanje.

2. Alternativni tok

V primeru, da v aplikaciji ni kategorij, se prikaže obvestilo.

3. Zahteve

Jih ni.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Jih ni.

5.3.26 Dodajanje kategorije

1. Opis

Urednikom omogočimo dodajanje nove kategorije.

2. Tokovi dogodkov

1. Osnovni tok

Uredniku se prikaže obrazec, preko katerega doda novo kategorijo. Posredovati mora naslednje podatke: nadrejeno kategorijo (lahko je brez), naziv in opis. Pred shranjevanjem nove kategorije je potrebno validirati posredovane podatke.

2. Alternativni tok

Če posredovani podatki niso uspešno validirani, se o tem obvesti urednika.

3. Zahteve

Jih ni.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Kategorija je dodana.

5.3.27 Urejanje kategorije

1. Opis

Urednikom omogočimo urejanje obstoječe kategorije.

2. Tokovi dogodkov

1. Osnovni tok

Uredniku se prikaže obrazec, preko katerega urejajo kategorijo. Spreminja lahko naslednje podatke: nadrejena kategorija, naziv in opis. Pred shranjevanjem nove kategorije je potrebno validirati posredovane podatke.

2. Alternativni tok

Če posredovani podatki niso uspešno validirani, se o tem obvesti urednika.

3. Zahteve

Izbrana kategorija mora obstajati.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Kategorija je spremenjena.

5.3.28 Brisanje kategorije

1. Opis

Urednikom omogočimo brisanje kategorije.

2. Tokovi dogodkov

1. Osnovni tok

Po kliku na povezavo s seznama kategorij uredniku prikažemo sporočilo, ki ga sprašuje, če je prepričan, da želi izbrisati kategorijo. Po potrditvi se kategorija izbriše. Izbrišejo se tudi vse povezave na videe za to kategorijo in seznam vseh podrejenih kategorij.

2. Alternativni tok

Če urednik ne potrdi brisanja, umaknemo sporočilo in ničesar ne zberemo.

3. Zahteve

Izbrana kategorija mora obstajati.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Kategorija je izbrisana.

5.3.29 Pregled posamezne kategorije

1. Opis

Urednikom prikažemo posamezno kategorijo.

2. Tokovi dogodkov

1. Osnovni tok

Po kliku na povezavo s seznama kategorij uredniku prikažemo kategorijo. Prikažejo se podatki o kategoriji (naziv, opis, nadrejena kategorija).

Prikaže se seznam podrejenih kategorij, ki so povezane s to kategorijo. Na seznamu podrejenih kategorij prikažemo samo nazive podrejenih kategorij. Ob tem so za posamezno podrejeno kategorijo na seznamu na voljo naslednje akcije: pregled, urejanje in brisanje. V okviru tega seznama je uredniku na voljo urejanje vrstnega reda podrejenih kategorij. S spreminjanjem vrstnega reda urednik določa vrstni red prikazovanja podrejenih kategorij glede na to kategorijo.

Prikaže se tudi seznam videov, ki so povezani s to kategorijo. Na seznamu videov prikažemo naslednje podatke: naziv videa in ime video datoteke. Ob tem so za posamezni video na seznamu na voljo naslednje akcije: pregled, urejanje in brisanje. V okviru tega seznama je uredniku na voljo urejanje vrstnega reda videov. S spreminjanjem vrstnega reda urednik določa vrstni red prikazovanja videov glede na to kategorijo.

2. Alternativni tok

V primeru, da izbrana kategorija ne obstaja, se prikaže obvestilo. Če kategorija ne vsebuje povezanih videov ali podrejenih kategorij, se o tem obvesti urednika.

3. Zahteve

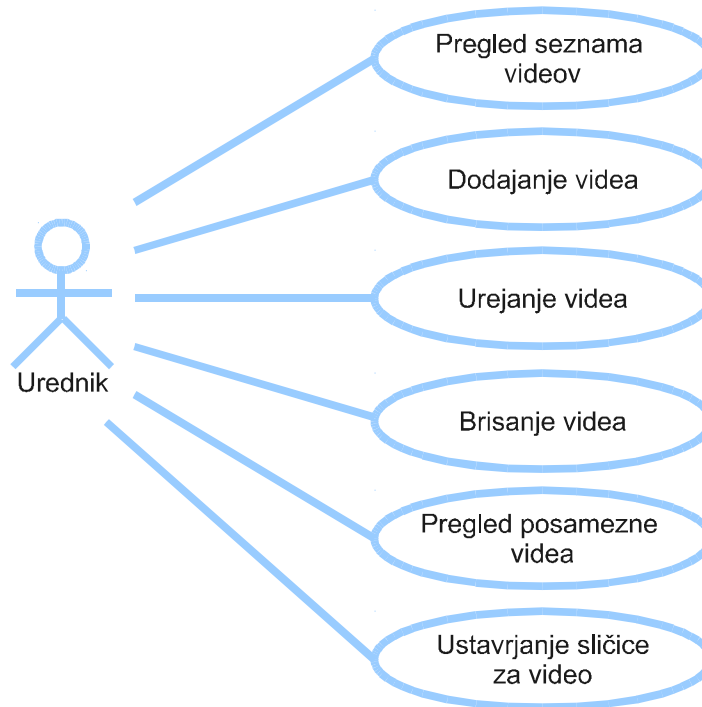
Kategorija mora obstajati.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Če je urednik spremenil vrstni red videov ali podrejenih kategorij, se spremenjeni vrstni red shrani.



Slika 5.10: Primeri uporabe za urednika v okviru videov

5.3.30 Pregled seznama videov

1. Opis

Urednikom prikažemo seznam vseh videov.

2. Tokovi dogodkov

1. Osnovni tok

Dostop do seznama je omogočen preko klika na povezavo v navigaciji, ki je prikazana v okviru uredniškega dela aplikacije. Seznam prikazuje naziv videa, ime video datoteke, jezik, trajanje, velikost video datoteke in datum kreiranja. V primeru, da je v aplikaciji več kot 20 videov, se prikaz vseh razdeli na več strani. Urednik lahko z uporabo povezav “naslednja stran” ali “prejšnja stran” pregleduje vse strani.

Za vsak posamezni video na seznamu so na voljo naslednje akcije:

- pregledovanje,
- urejanje,
- brisanje in
- ustvarjanje sličice.

2. Alternativni tok

V primeru, da v aplikaciji ni videov, se prikaže obvestilo.

3. Zahteve

Jih ni.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Jih ni.

5.3.31 Dodajanje videa

1. Opis

Urednikom omogočimo dodajanje novega videa.

2. Tokovi dogodkov

1. Osnovni tok

Uredniku se prikaže obrazec, preko katerega doda nov video. Posredovati mora naslednje podatke:

- seznam kategorij, katerim pripada,
- naziv,

- opis,
- jezik in
- referenco do datoteke na lokalnem sistemu, ki predstavlja video.

Pred shranjevanjem novega videa je potrebno validirati posredovane podatke.

2. Alternativni tok

Če posredovani podatki niso uspešno validirani, se o tem obvesti urednika.

3. Zahteve

Jih ni.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Video je dodan.

5.3.32 Urejanje videa

1. Opis

Urednikom omogočimo urejanje obstoječega videa.

2. Tokovi dogodkov

1. Osnovni tok

Uredniku se prikaže obrazec, preko katerega urejajo video. Spreminja lahko naslednje podatke:

- seznam kategorij, katerim pripada video,
- naziv,
- opis,

- jezik in
- datoteko, ki predstavlja video.

Pred shranjevanjem novega videa je potrebno validirati posredovane podatke.

2. Alternativni tok

Če posredovani podatki niso uspešno validirani, se o tem obvesti urednika.

3. Zahteve

Izbran video mora obstajati.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Video je spremenjen.

5.3.33 Brisanje videa

1. Opis

Urednikom omogočimo brisanje videa.

2. Tokovi dogodkov

1. Osnovni tok

Po kliku na povezavo s seznama videov uredniku prikažemo sporočilo, ki ga sprašuje, če je prepričan, da želi izbrisati video. Po potrditvi se video izbriše. Izbriše se tudi datoteka, ki predstavlja ta video.

2. Alternativni tok

Če urednik ne potrdi brisanja, umaknemo sporočilo in ničesar ne zbrišemo.

3. Zahteve

Izbran video mora obstajati.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Video je izbrisan.

5.3.34 Pregled posameznega videa

1. Opis

Urednikom prikažemo posamezni video.

2. Tokovi dogodkov

1. Osnovni tok

Po kliku na povezavo s seznama videov uredniku prikažemo video. Prikažejo se podatki o video (naziv, opis, naziv datoteke, slička, jezik, širina, višina, trajanje, velikost datoteke, meta podatki o videu in datum kreiranja). Ob kliku na sličico urednik lahko predvaja video.

Prikaže se tudi seznam kategorij, ki so povezane s tem videom. Na seznamu kategorij prikažemo samo naziv kategorije. Ob tem so za posamezno kategorijo na seznamu na voljo naslednje akcije: pregled, urejanje in brisanje.

2. Alternativni tok

V primeru, da izbrani video ne obstaja, se prikaže obvestilo. Če video ne vsebuje povezanih kategorij, se o tem obvesti urednika.

3. Zahteve

Video mora obstajati.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Jih ni.

5.3.35 Ustvarjanje sličice za video

1. Opis

Urednikom omogočamo ustvarjanje sličice (angl. *thumbnail*) za posamezni video.

2. Tokovi dogodkov

1. Osnovni tok

Po kliku na povezavo s seznama videov uredniku prikažemo video. Urednik predvaja video in, ko zagleda sličico, ki bi lahko ustrezno predstavljala video, izvede akcijo, da to sliko pridobi in jo shrani.

2. Alternativni tok

V primeru, da izbrani video ne obstaja, se prikaže obvestilo.

3. Zahteve

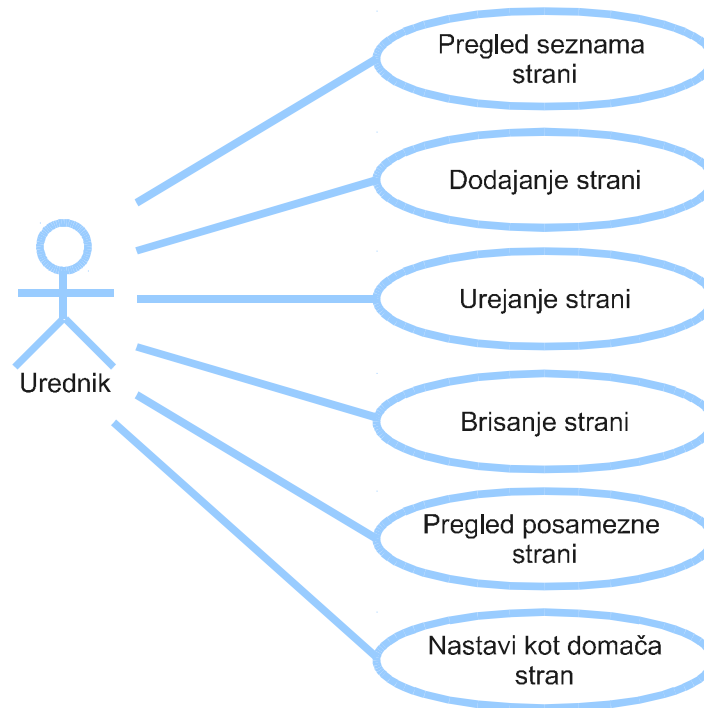
Video mora obstajati.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Video ima ustvarjeno novo sličico.



Slika 5.11: Primeri uporabe za urednika v okviru vsebinskih strani

5.3.36 Pregled seznama vsebinskih strani

1. Opis

Urednikom prikažemo seznam vseh vsebinskih strani.

2. Tokovi dogodkov

1. Osnovni tok

Dostop do seznama je omogočen preko klika na povezavo v navigaciji, ki je prikazana v okviru uredniškega dela aplikacije. Seznam prikazuje naziv strani, URL alias, postavitev in datum ustvarjanja. V primeru, da je v aplikaciji več kot 20 strani, se prikaz vseh razdeli na več strani. Urednik lahko z uporabo povezav “naslednja stran” ali “prejšnja stran” pregleduje vse strani.

Za vsako posamezno stran na seznamu so na voljo naslednje akcije:

- pregledovanje,
- urejanje,

- brisanje in
- določanje domače strani (stran, ki jo vidi obiskovalec, ko obiše naše spletno mesto).

2. Alternativni tok

V primeru, da v aplikaciji ni strani, se prikaže obvestilo.

3. Zahteve

Jih ni.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Jih ni.

5.3.37 Dodajanje vsebinske strani

1. Opis

Urednikom omogočimo dodajanje nove vsebinske strani.

2. Tokovi dogodkov

1. Osnovni tok

Uredniku se prikaže obrazec, preko katerega doda novo vsebinsko stran. Posredovati mora naslednje podatke: naziv, URL alias, vsebino in postavitev (navigacija v glavi spletne strani, navigacija v nogi spletne strani in ostalo). Pred shranjevanjem nove strani je potrebno validirati posredovane podatke.

2. Alternativni tok

Če posredovani podatki niso uspešno validirani, se o tem obvesti urednika.

3. Zahteve

Jih ni.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Vsebinska stran je dodana.

5.3.38 Urejanje vsebinske strani

1. Opis

Urednikom omogočimo urejanje obstoječe vsebinske strani.

2. Tokovi dogodkov

1. Osnovni tok

Uredniku se prikaže obrazec, preko katerega ureja stran. Spreminja lahko naslednje podatke: naziv, URL alias, vsebino in postavitev. Pred shranjevanjem sprememb vsebinske strani je potrebno validirati posredovane podatke.

2. Alternativni tok

Če posredovani podatki niso uspešno validirani, se o tem obvesti urednika.

3. Zahteve

Izbrana vsebinska stran mora obstajati.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Vsebinska stran je spremenjena.

5.3.39 Brisanje vsebinske strani

1. Opis

Urednikom omogočimo brisanje vsebinske strani.

2. Tokovi dogodkov

1. Osnovni tok

Po kliku na povezavo s seznama strani uredniku prikažemo sporočilo, ki ga sprašuje, če je prepričan, da želi izbrisati stran. Po potrditvi se stran izbriše.

2. Alternativni tok

Če urednik ne potrdi brisanja, umaknemo sporočilo in ničesar ne zberemo.

3. Zahteve

Izbrana vsebinska stran mora obstajati.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Vsebinska stran je izbrisana.

5.3.40 Pregled posamezne vsebinske strani

1. Opis

Urednikom prikažemo posamezno vsebinsko stran.

2. Tokovi dogodkov

1. Osnovni tok

Po kliku na povezavo s seznama strani uredniku prikažemo vsebinsko stran. Prikažejo se podatki o strani (naziv, URL alias, vsebina, postavitve in datum ustvarjanja).

2. Alternativni tok

V primeru, da izbrana stran ne obstaja, se prikaže obvestilo.

3. Zahteve

Vsebinska stran mora obstajati.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Jih ni.

5.3.41 Nastavi kot domačo stran

1. Opis

Urednikom omogoča, da posamezno vsebinsko stran nastavijo kot domačo stran. Domača stran je dejansko stran, ki jo obiskovalci vidijo, ko obišejo našo spletno mesto.

2. Tokovi dogodkov

1. Osnovni tok

Po kliku na povezavo s seznama strani aplikacija izbrano stran nastavi kot domačo stran. Trenutna domača stran se nastavi kot normalna stran.

2. Alternativni tok

V primeru, da izbrana stran ne obstaja, se prikaže obvestilo.

3. Zahteve

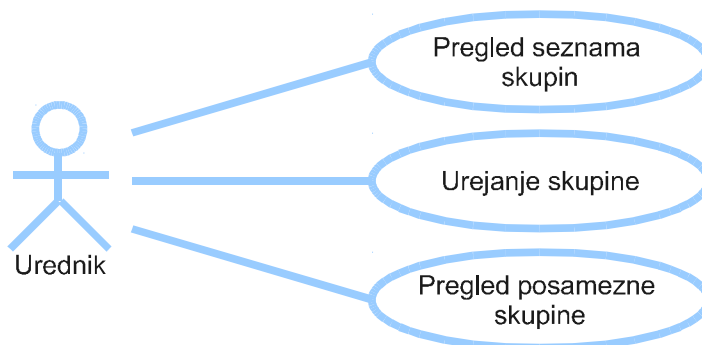
Vsebinska stran mora obstajati.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Vsebinska stran je nastavljena kot domača stran.



Slika 5.12: Primeri uporabe za urednika v okviru uporabniških skupin

5.3.42 Pregled seznama uporabniških skupin

1. Opis

Urednikom prikažemo seznam vseh uporabniških skupin.

2. Tokovi dogodkov

1. Osnovni tok

Dostop do seznama je omogočen preko klika na povezavo v navigaciji, ki je prikazana v okviru uredniškega dela aplikacije. Seznam prikazuje naziv skupine, datum ustvarjanja in datum zadnjega spreminjanja. V aplikaciji sta na voljo dve uporabniški skupini, in sicer administratorji in uporabniki.

Za vsako posamezno skupino na seznamu sta na voljo naslednji akciji:

- pregledovanje in
- urejanje.

2. Alternativni tok

Ga ni.

3. Zahteve

Jih ni.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Jih ni.

5.3.43 Urejanje uporabniške skupine

1. Opis

Urednikom omogočimo urejanje obstoječe uporabniške skupine.

2. Tokovi dogodkov

1. Osnovni tok

Uredniku se prikaže obrazec, preko katerega urejajo skupino. Spreminja lahko samo naziv skupine. Pred shranjevanjem naziva skupine je potrebno validirati posredovane podatke.

2. Alternativni tok

Če posredovani podatki niso uspešno validirani, se o tem obvesti urednika.

3. Zahteve

Jih ni.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Naziv uporabniške skupine je spremenjen.

5.3.44 Pregled posamezne uporabniške skupine

1. Opis

Urednikom prikažemo posamezno uporabniško skupino.

2. Tokovi dogodkov

1. Osnovni tok

Po kliku na povezavo s seznama strani uredniku prikažemo skupino. Prikažejo se podatki o skupini (naziv, datum ustvarjanja in datum zadnjega spreminjanja).

Prikaže se tudi seznam vseh uporabnikov, ki pripadajo tej skupini. Za posameznega uporabnika prikažemo naslednje podatke: uporabniško ime, datum ustvarjanja in datum zadnjega spreminjanja. Na voljo so tudi naslednje akcije: pregledovanje, urejanje in brisanje.

2. Alternativni tok

Če v skupini ni uporabnikov, se o tem obvesti urednika.

3. Zahteve

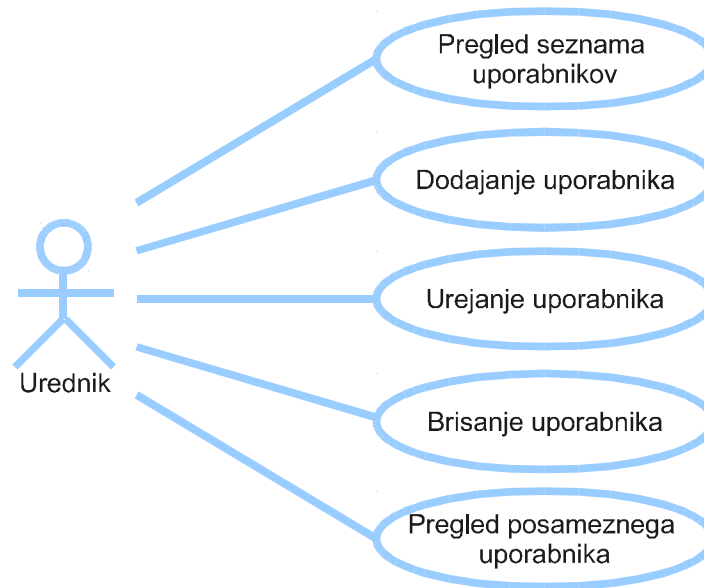
Uporabniška skupina mora obstajati.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Jih ni.



Slika 5.13: Primeri uporabe za urednika v okviru uporabnikov

5.3.45 Pregled seznama uporabnikov

1. Opis

Urednikom prikažemo seznam vseh uporabnikov.

2. Tokovi dogodkov

1. Osnovni tok

Dostop do seznama je omogočen preko klika na povezavo v navigaciji, ki je prikazana v okviru uredniškega dela aplikacije. Seznam prikazuje uporabniško ime, e-naslov, uporabniško skupino, datum kreiranja in datum zadnjega spreminjanja. V primeru, da je v aplikaciji več kot 20 uporabnikov, se prikaz vseh razdeli na več strani. Urednik lahko z uporabo povezav “naslednja stran” ali “prejšnja stran” pregleduje vse strani.

Za vsakega posameznega uporabnika na seznamu so na voljo naslednje akcije:

- pregledovanje,
- urejanje in
- brisanje.

2. Alternativni tok

V primeru, da v aplikaciji ni uporabnikov, se prikaže obvestilo.

3. Zahteve

Jih ni.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Jih ni.

5.3.46 Dodajanje uporabnika

1. Opis

Urednikom omogočimo dodajanje novega uporabnika.

2. Tokovi dogodkov

1. Osnovni tok

Uredniku se prikaže obrazec, preko katerega doda novega uporabnika. Posredovati mora naslednje podatke: e-naslov, ime, priimek, uporabniško ime, geslo in uporabniško skupino. Pred shranjevanjem novega uporabnika je potrebno validirati posredovane podatke.

2. Alternativni tok

Če posredovani podatki niso uspešno validirani, se o tem obvesti urednika.

3. Zahteve

Jih ni.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Uporabnik je dodan.

5.3.47 Urejanje uporabnika

1. Opis

Urednikom omogočimo urejanje obstoječega uporabnika.

2. Tokovi dogodkov

1. Osnovni tok

Uredniku se prikaže obrazec, preko katerega urejajo uporabnika. Spreminja lahko naslednje podatke: e-naslov, ime, priimek, uporabniško ime, geslo in uporabniško skupino. Pred shranjevanjem sprememb je potrebno validirati posredovane podatke.

2. Alternativni tok

Če posredovani podatki niso uspešno validirani, se o tem obvesti urednika.

3. Zahteve

Uporabnik mora obstajati.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Podatki uporabnika so spremenjeni.

5.3.48 Brisanje uporabnika

1. Opis

Urednikom omogočimo brisanje uporabnika.

2. Tokovi dogodkov

1. Osnovni tok

Po kliku na povezavo s seznama uporabnikov uredniku prikažemo sporočilo, ki ga sprašuje, če je prepričan, da želi izbrisati stran. Po potrditvi se uporabnik izbriše. Izbrišejo se tudi vsi njegovi izbrani videi, lokacije in statistika.

2. Alternativni tok

Če urednik ne potrdi brisanja umaknemo sporočilo in ničesar ne zberemo.

3. Zahteve

Uporabnik mora obstajati.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Uporabnik je izbrisan.

5.3.49 Pregled posameznega uporabnika

1. Opis

Urednikom omogočamo prikaz posameznega uporabnika.

2. Tokovi dogodkov

1. Osnovni tok

Po kliku na povezavo s seznama uporabnikov uredniku prikažemo uporabnika. Prikažejo se podatki o uporabniku (e-naslov, ime, priimek, uporabniško ime, uporabniška skupina, datum ustvarjanja in datum zadnjega spreminjanja).

2. Alternativni tok

V primeru, da uporabnik ne obstaja, se prikaže obvestilo.

3. Zahteve

Uporabnik mora obstajati.

4. Predpogoji

Urednik mora biti prijavljen v aplikaciji.

5. Popogoji

Jih ni.

6 Razvoj sistema

6.1 Razvojno okolje

Preden se lotimo razvoja aplikacije, je potrebno postaviti razvojno okolje, ki nam omogoča učinkovitejše in enostavnejše razvijanje. Na razpolago imamo široko paleto možnosti. Lahko uporabimo številne licenčne pakete, ki vključujejo večino potrebnih in želenih funkcionalnosti. Vendar, če želimo uporabiti določene funkcionalnosti, ki jih ti paketi ne omogočajo, imamo možnost dokupiti te funkcionalnosti ali pa jih ne uporabljati. Po drugi strani pa lahko uporabimo številne odprtokodne rešitve, ki vključujejo mnoge funkcionalnosti, potrebne za razvoj. Tudi če ti paketi ne omogočajo želenih funkcionalnosti, obstaja množična skupnost, ki razvija dodatke za te pakete. Ena od teh odprtokodnih rešitev je tudi programski paket Eclipse, ki ga bomo uporabljali za razvoj naše aplikacije.

6.1.1 Eclipse

Eclipse je integrirano razvojno okolje (angl. IDE – *Integrated Development Environment*), ki je bilo v osnovi izdelano za razvoj programskih rešitev v jeziku Java. Poglavitno prednost tega okolja predstavlja odlična zmožnost razširjanja osnovnega okolja. Tako so številni člani skupnosti razvili veliko število dodatkov, ki omogočajo razvijanje programskih rešitev v številnih programskih jezikih. Za naše potrebe potrebujemo v osnovi dve razširitvi, in sicer dodatka PDT in AXDT.

PDT (angl. *PHP Development Tools*) predstavlja skupek orodij, ki omogočajo učinkovito razvijanje rešitev PHP v okviru platforme Eclipse. V osnovi ta orodja temeljijo na projektu WTP (angl. *Web Tools Platform*) in projektu DLTK (angl. *Dynamic Language ToolKit*). Orodja vsebujejo urejevalnik za jezik PHP, razhroščevalnik in omogočajo prikaz orisa projekta. Urejevalnik vključuje številne uporabne funkcionalnosti, npr. sintaksno obarvanje kode, dopolnjevanje kode. PDT ne vsebuje razhroščevalnika, ampak omogoča transparentno povezavo in nadzor nad obstoječimi razhroščevalniki. Za naše potrebe smo uporabili razhroščevalnik podjetja Zend, ki vključuje vse potrebne funkcionalnosti za uspešno razhroščevanje. Projekt WTP predstavlja pomembno okolje za učinkovito generiranje dokumentov XHTML. Med kreiranjem in urejanjem dokumenta nam sproti pregleduje skladnost sintakse z definiranim standardom. Tako ni potrebno naknadno preverjanje ustreznosti z uporabo orodja W3C, ki je relativno časovno potratno.

Za razvoj aplikacij v okolju Adobe Flash je potrebno platformo Eclipse razširiti z dodatkom

AXDT (angl. *Actionscript 3 and Flex Development Tools*). Tudi ta dodatek za svoje delovanje uporablja rešitve drugega projekta, in sicer IMP (angl. *IDE Meta-Tooling Platform*). Vsebuje urejevalnik in prevajalnik za skriptni jezik ActionScript 3. Prevajalnik je del odprtokodnega projekta FlexSDK, ki ga je podjetje Adobe leta 2008 predal odprtokodni skupnosti.

Za našo aplikacijo potrebujemo tudi orodje, ki omogoča enostavnejše razvijanje rešitev z uporabo skriptnega jezika JavaScript. Dodatek, ki platformi Eclipse doda urejevalnik za JavaScript, se imenuje JSEclipse. Omogoča dopolnjevanje programske kode, sintaksno obarvanje kode, prikaz orisa projekta itd.

Z uporabo vseh zgoraj naštetih orodij pridobimo močno okolje za razvoj internetnih aplikacij. Čeprav porabimo nekaj več časa, da takšno okolje vzpostavimo, je pridobitev ogromna. V enem okolju imamo v vsakem trenutku na razpolago vsa orodja, ki jih potrebujemo. Pri razvoju internetnih aplikacij velikokrat preklapljammo med različnimi tehnologijami, v našem primeru med jeziki PHP, JavaScript, ActionScript in XHTML. Če imamo vsa potrebna orodja na enem mestu, nam ni potrebno vsakokrat, ko želimo urejati datoteko v določenem programskem jeziku, uporabiti drugo platformo. Poglavitna prednost uporabe platforme Eclipse je v tem, da se ni potrebno za vsak nov programski jezik naučiti tudi uporabe novega razvojnega okolja. Platforma je vedno ista, ki jo nato samo razširimo z dodatki, da pridobimo želene funkcionalnosti.

6.2 Nadzor razvoja

Pri razvoju projekta je izrednega pomena nadzor nad celotnim izvajanjem le-tega. Namen projekta je doseči vnaprej zastavljene cilje. Da bi lahko dosegli te cilje učinkovito in pravočasno, je potrebno izvajati nadzor nad razvojem projekta. Nadzor je še toliko bolj pomemben, če projekt razvija več programerjev oz. razvijalcev. V tem primeru je potrebno vzpostaviti sistem, ki bo omogočal nadzor nad spremembami posameznih programerjev. Če takšnega nadzora ni, se lahko zgodi, da določen programer prepiše delo drugega programerja.

6.2.1 SVN

Za ta namen bomo uporabili rešitev SVN (angl. *Subversion*), ki predstavlja nadslednika podobnega sistema, imenovanega CVS (angl. *Concurrent Versions Systems*). SVN je produkt organizacije Apache. Omogoča učinkovito in pregledno sledenje spremembam med razvojem programske kode. Celotna programska koda projekta je shranjena v t. i. shrambi (angl. *repository*). Vsi razvijalci projekta shranjujejo svoje spremembe v to shrambo. Posamezni razvijalec spreminja svojo kopijo te shrambe, ki ji pravimo delovni direktorij (angl. *working directory*). Ko je razvijalec zadovoljen z rezultatom v svojem delovnem direktoriju, te

spremembe shrani v centralno shrambo. Ostali razvijalci si lahko nato te nove spremembe prenesejo v svoje delovne direktorije in nadaljujejo z razvojem.

Za vzpostavitev takšnega sistema je potrebno postaviti strežnik SVN, ki predstavlja centralno shrambo. Vsi razvijalci projekta dostopajo do tega strežnika z namenom shranjevanja sprememb in pridobivanja sprememb ostalih razvijalcev. Posamezni razvijalec potrebuje klienta SVN, ki omogoča sledenje spremembam v delovnem direktoriju in komunicira s centralno shrambo. Po nastavitvi strežnika SVN je potrebno ustvariti centralno shrambo (glej primer 6.1). Naslednji korak predstavlja prvotni uvoz projekta v novo ustvarjeno shrambo (glej primer 6.2). Prvotni uvoz je potreben, če imamo že ustvarjene določene datoteke projekta, katerih spremembam bi želeli slediti z uporabo sistema SVN [11]. V našem primeru imamo na začetku datoteke ogrodja CakePHP, ki jih je potrebno dodati v shrambo.

```
svnadmin create /home/repos/videogadget.iljaz.com
```

Primer 6.1: Kreiranje nove shrambe

```
svn import /home/projects/videogadget.iljaz.com \
file:///home/repos/videogadget.iljaz.com -m "Prvotni uvoz"
```

Primer 6.2: Prvotni uvoz projekta v shrambo

Drugi del sistema SVN predstavlja program na strani klienta, ki omogoča sledenje spremembam v lokalnem delovnem direktoriju. Ker je naše razvojno okolje postavljeno na platformi Eclipse, bomo uporabili dodatek Subclipse, ki to platformo razširi z zmožnostjo uporabe centralne shrambe SVN. V platformi Eclipse je potrebno ustvariti nov projekt, ki pridobi datoteke iz centralne shrambe in ustvari delovni direktorij. Potem smo pripravljeni za delo na projektu. Ko smo zadovoljni s spremembami našega dela projekta, lahko te spremembe shranimo v centralno shrambo. Vse to v okviru platforme Eclipse.

6.2.2 TRAC

V poglavju 5.2 smo določili vse aktivnosti, ki jih je potrebno izvesti za doseg zastavljenih ciljev projekta. Za učinkovito nadzorovanje izvajanja teh aktivnosti lahko uporabimo produkt TRAC, ki je napreden sistem za sledenje razvoja projekta. Omogoča dodajanje komentarjev med razvojem projekta, lahko kreiramo dokumentacijo projekta, predvsem pa omogoča sledenje odpravljanja problemov in izvajanja aktivnosti med razvojem. Tako lahko razvijalci ali ostali člani projekta komentirajo razvoj, beležijo izvajanje aktivnosti in odpravljanje napak oz. drugih problemov. TRAC pa vsebuje še eno zelo uporabno funkcionalnost, in sicer omogoča povezavo s shrambo SVN. Tako lahko z uporabo TRAC-a sledimo spremembam posameznih datotek in jih tudi komentiramo.

Za potrebe naše aplikacije bomo TRAC povezali s shrambo SVN in ga hkrati uporabili za sledenje izvajanja aktivnosti. Po postavitvi sistema TRAC vnesemo vse aktivnosti, katerih napredek bomo med razvojem projekta spremljali. Slika 6.1 prikazuje seznam vseh aktivnosti v sistemu TRAC in njihov status, ki določa, da so vse aktivnosti opravljene.

Administrativni del aplikacije (5 matches)							
Ticket	Summary	Component	Status	Resolution	Version	Type	Priority
#1	Avtentikacija in avtorizacija uporabnikov	component1	closed	fixed		task	major
#2	Upravljanje uporabnikov in skupin	None	closed	fixed		task	major
#8	Upravljanje kategorij	None	closed	fixed		task	major
#9	Upravljanje videov	None	closed	fixed		task	major
#10	Upravljanje vsebinskih strani	None	closed	fixed		defect	major

Podatkovni model (5 matches)							
Ticket	Summary	Component	Status	Resolution	Version	Type	Priority
#3	Entitete in atributi	None	closed	fixed		task	major
#4	Določiti odvisnosti med entitetami	None	closed	fixed		task	major
#7	Izdelati fizični pod. model	None	closed	fixed		task	major
#6	Izdelati logični pod. model	None	closed	fixed		task	major
#5	Izdelati konceptualni pod. model	None	closed	fixed		task	major

Predvajalnik videov (4 matches)							
Ticket	Summary	Component	Status	Resolution	Version	Type	Priority
#19	Izdelava upor. vmesnika	None	closed	fixed		task	major
#20	Parametri	None	closed	fixed		task	major
#22	Beleženje statistike	None	closed	fixed		task	major
#21	Predvajanje videov	None	closed	fixed		task	major

Skripta za vključitev predvajalnika (4 matches)							
Ticket	Summary	Component	Status	Resolution	Version	Type	Priority
#23	Preverjanje skladnosti brskalnika	None	closed	fixed		task	major
#26	Prikaz gumba	None	closed	fixed		task	major
#25	Vključitev v dokument XHTML	None	closed	fixed		task	major
#24	Preverjanje skladnosti predvajalnika Flash	None	closed	fixed		task	major

Uporabniški del aplikacije (8 matches)							
Ticket	Summary	Component	Status	Resolution	Version	Type	Priority
#11	Izdelava dizajna in postavitve	None	closed	fixed		task	major
#12	Izdelava navigacije in ogleda vsebinskih strani	None	closed	fixed		task	major
#13	Prikaz kategorij	None	closed	fixed		task	major
#15	Registracija in prijava uporabnikov	None	closed	fixed		task	major
#16	Upravljanje lokacij	None	closed	fixed		task	major
#17	Upravljanje izbranih videov	None	closed	fixed		task	major
#18	Pregled statistike	None	closed	fixed		task	major
#14	Prikaz videov	None	closed	fixed		task	major

Slika 6.1: Seznam aktivnosti v sistemu TRAC

6.3 Izdelava dizajna in postavitve

Dizajn je tisti del aplikacije, ki ga obiskovalci in uporabniki najhitreje opazijo, zato je pomembno, da nekaj časa posvetimo tudi izdelavi le-tega. Dizajn, ki ga želimo izdelati za našo aplikacijo, mora biti čim bolj pregleden in uporaben, hkrati pa privlačen za oko. Slika 6.2 prikazuje dizajn, ki ga bomo uporabili za našo aplikacijo, ampak samo za uporabniški del aplikacije. Administrativni del aplikacije bo uporabljal privzeti dizajn, ki je vključen v ogrodju CakePHP. Za izdelavo dizajna uporabimo programski paket za urejanje grafik, npr. GIMP (angl. *GNU Image Manipulation Program*). Ko smo zadovoljni z izdelanim predogledom v programu GIMP, posamezne slike izvozimo v formatu PNG (angl. *Portable Network Graphics*) ali GIF (angl. *Graphics Interchange Format*). Za postavitev strani s slike 6.2 potrebujemo samo tri slike,

in sicer ozadje za glavo strani, logo in ozadje gumbov v glavi. Vse ostale učinke lahko dosežemo z elementi XHTML in stili CSS.



Slika 6.2: Predogled dizajna

Naslednji korak je izdelava postavitve, ki predstavlja izdelavo dokumenta XHTML, s katerim opredelimo strukturo spletne strani. Potrebno je predvideti prostor za navigacijo v glavi, nogi in levi koloni strani. Podobno definiramo prostor, kjer se bo prikazovala osnovna vsebina.

Postavitev spletne strani zgradimo z uporabo posebnega mehanizma v ogrodju CakePHP, ki omogoča vstavljanje posebnih spremenljivk v dokumente XHTML. Te spremenljivke predstavljajo prostor, kjer se bo prikazala določena vsebina, npr. navigacija ali osnovna vsebina. CakePHP vse poglede grupira v poseben direktorij, imenovan "views". Tako so tudi osnovne postavitve spletne strani shranjene v tem direktoriju, in sicer v poddirektoriju "layouts". Uporabimo lahko več različnih osnovnih pogledov, med katerimi preklapljamo v kontrolerju. Za našo aplikacijo potrebujemo štiri osnovne poglede, in sicer osnovni pogled:

- za uredniški del aplikacije,
- za uporabniški del aplikacije v okviru uporabniškega računa,

- za uporabniški del aplikacije v splošnem,
- za prikaz napak med delovanjem sistema.

Postavitev je shranjena v posebnem tipu datoteke, kjer se prepletajo elementi XHTML in koda PHP. Z uporabo kode PHP vstavljamo posebne spremenljivke, ki se med delovanjem aplikacije zamenjajo z vsebino, generirano v aplikaciji (glej primer 6.3).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <?php echo $this->Html->charset(); ?>
    <base href="<?php print Configure::read('APP_URL'); ?>/" />
    <title>
        <?php __('Video Gadget'); ?> -
        <?php echo $title_for_layout; ?>
    </title>
    <?php
        echo $this->Html->meta('icon');
        echo $this->Html->css('cake.frontend');
        echo $scripts_for_layout;
    ?>
</head>
<body>
    <div id="header">
        <div id="logo">
            <a href="<?php print Configure::read('APP_URL'); ?>">
                <img alt="<?php __('Video Gadget LOGO'); ?>"
                    src="img/logo.gif"
                    title="<?php
                        print Configure::read('SITE_NAME'); ?>" />
            </a>
        </div>
        <div id="topmenu">
            <?php print $this->element('topmenu'); ?>
        </div>
    </div>
    <div id="main">
        <table width="1020" border="0" cellspacing="0"
            cellpadding="0">
            <tr>
                <td id="left" valign="top">
                    <h3><?php __('Kategorije'); ?></h3>
```

```

        <?php print $this->requestAction(
            '/categories/treemenu',
            array('return')); ?>
    </td>
    <td id="content" valign="top">
        <h1>
            <?php echo $title_for_layout; ?>
        </h1>
        <?php echo $this->Session->flash(); ?>

        <?php echo $content_for_layout; ?>
    </td>
</tr>
</table>
</div>
<div id="footer">
    <?php print $this->element('footer'); ?>
</div>
</body>
</html>

```

Primer 6.3: Osnovna postavitev za uporabniški del aplikacije za splošno uporabo

V primeru 6.3 vidimo uporabo dveh osnovnih spremenljivk: `$title_for_layout` in `$content_for_layout`. Prva predstavlja naslov strani, druga pa vsebuje osnovno vsebino. Omembe vredna sta še dva mehanizma, ki ju uporabljamo. Prvi je t. i. element, ki omogoča uporabo iste kode na večih mestih v postavitvah oz. pogledih. Elemente kreiramo v poddirektoriju pogledov, imenovanem “elements”. V zgornjem primeru uporabljamo element za prikaz navigacije v glavi in nogi. Primer 6.4 predstavlja element *topmenu*. Drugi mehanizem omogoča klic določene akcije v kontrolerju. Na takšen način prikažemo hierarhijo kategorij, kar je razvidno iz primera 6.5.

```

<?php print $this->element('topmenu'); ?>

```

Primer 6.4: Uporaba elementa za prikaz navigacije v glavi strani

```

<?php print $this->requestAction('/categories/treemenu',
    array('return')); ?>

```

Primer 6.5: Uporaba klica akcije v kontrolerju za prikaz hierarhije kategorij

6.4 Izdelava aplikacije

Uporaba ogrodja Cake narekuje izdelavo aplikacije s trinivojsko arhitekturo oz. z vzorcem model-pogled-kontroler. Najprej izdelamo podatkovni model, kjer bomo shranjevali podatke.

Sledi implementacija sloja, ki bo dostopal do teh podatkov. Nadaljujemo s slojem poslovne logike, ki izvaja akcije nad podatki in komunicira s slojem uporabniškega vmesnika. Na koncu izdelamo sloj pogledov, ki omogočajo prikazovanje podatkov, obdelanih s kontrolerji. V bistvu razvoj aplikacije ne sledi redosledu, ki je naveden zgoraj, ampak pri implementaciji določene funkcionalnosti konstantno skačemo med zgornjimi tremi sloji.

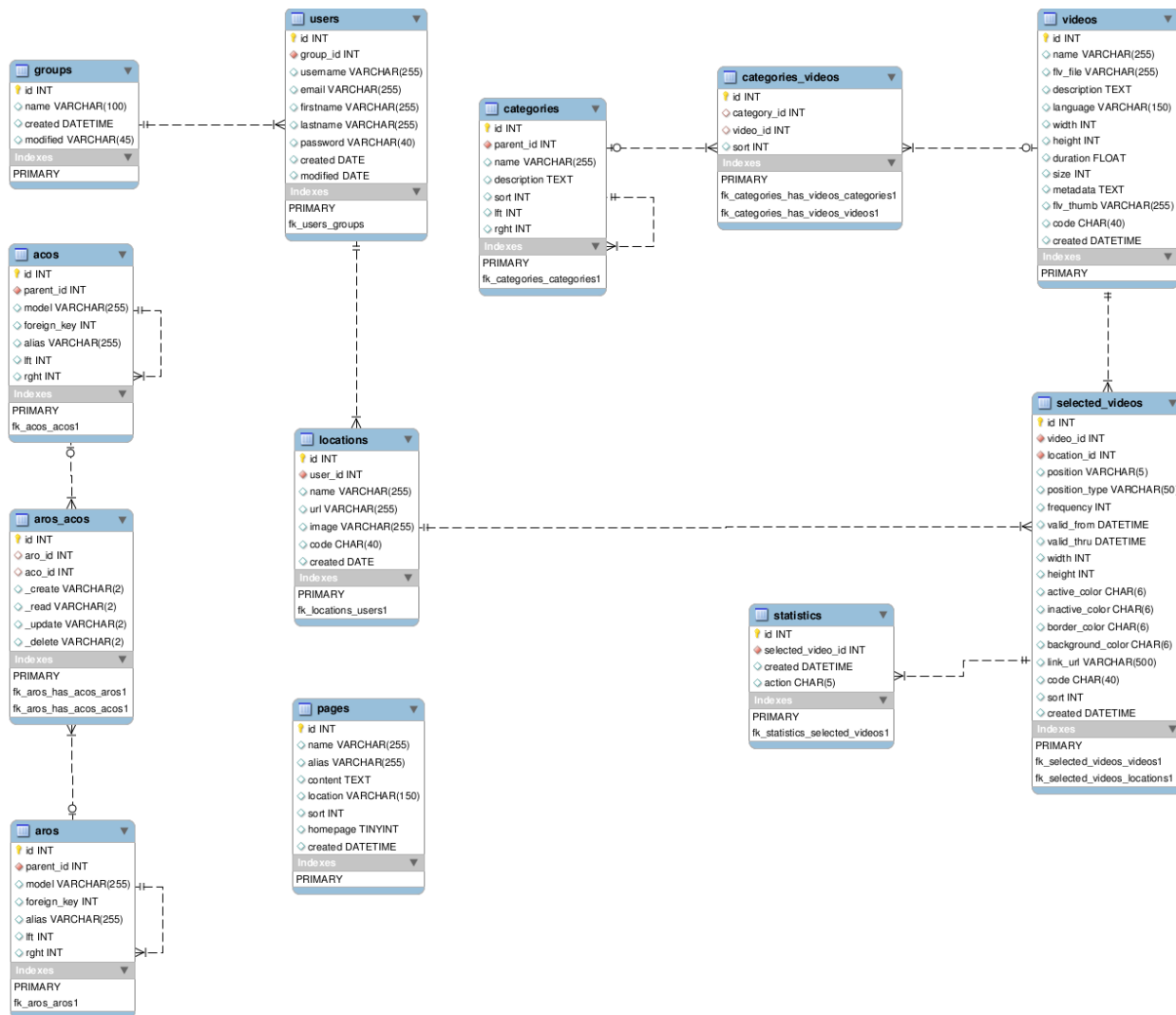
6.4.1 Sloj podatkov

Aplikacija se bo ukvarjala z obdelavo podatkov, zato je potrebno imeti podatke nekje shranjene. Za potrebe naše aplikacije bomo uporabili relacijsko podatkovno bazo MySQL. Preden se lotimo implementacije tabel, ki bodo vsebovale podatke, je potrebno modelirati podatke iz realnega sveta. Modeliranja se lotimo na nivoju, kjer bomo definirali strukture teh podatkov in njihove soodvisnosti oz. povezave. Temu nivoju pravimo konceptualni podatkovni model. Za načrtovanje in izdelavo podatkovnega modela smo uporabili programski paket MySQL Workbench 5.1, ki pri izdelovanju uporablja drugačen pristop. Večina paketov za modeliranje podatkov uporablja pristop, kjer najprej določimo konceptualni model, nato logični in na koncu kreiramo fizični podatkovni model. Z uporabo paketa MySQL Workbench pri kreiranju konceptualnega modela hkrati izdelujemo tudi logični podatkovni model.

6.4.1.1 Konceptualni in logični podatkovni model

Pri izdelavi konceptualnega podatkovnega modela je potrebno odkriti entitete in njihove soodvisnosti. Pri določanju soodvisnosti hkrati kreiramo logični model. S takšnim pristopom k načrtovanju lahko sproti spremljamo izgled podatkovne baze. Slika 6.3 prikazuje izdelan konceptualni oz. logični podatkovni model.

Zaradi konvencij, ki jih uporablja Cake, je bilo potrebno med generiranjem logičnega podatkovnega modela določene attribute preimenovati. Z določenjem odvisnosti “many-to-many” se v logičnem modelu generira t. i. vmesna tabela, ki vsebuje tuja ključa obeh odvisnih tabel. Konvencija narekuje, da so tuji ključ v tabelah poimenovani v ednini, zato je potrebno te avtomatsko generirane tuje ključe prepisati v ednino. Podobno se zgodi pri odvisnosti “one-to-many”.



Slika 6.3: Logični podatkovni model

6.4.1.1.1 Entiteta “groups”

Entiteta “groups” vsebuje podatke o uporabniških skupinah. Vsak uporabnik aplikacije mora pripadati natančno eni skupini.

Tabela 6.1: Relacije entitete “groups”

Relacija	Entiteta 1	Entiteta 2	Kardinalnost
groups-users	groups	users	1, n

Tabela 6.2: Atributi entitete “groups”

Atribut	Podatkovni tip	Obvezen
id	INT	Da.

name	VARCHAR(100)	Da.
created	DATETIME	Da.
modified	DATETIME	Ne.

6.4.1.1.2 Entiteta “users”

Entiteta “users” vsebuje podatke o uporabnikih in njihovo pripadnost uporabniški skupini.

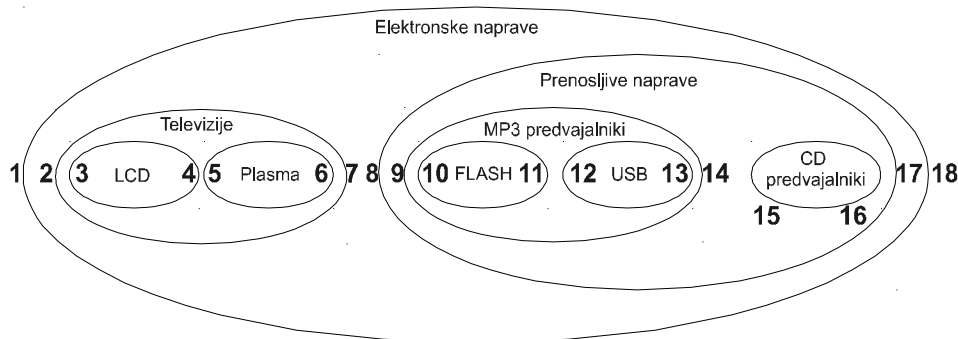
Tabela 6.3: Relacije entitete “users”

Relacija	Entiteta 1	Entiteta 2	Kardinalnost
groups-users	groups	users	1, n
users-locations	users	locations	1, n

Tabela 6.4: Atributi entitete “users”

Atribut	Podatkovni tip	Obvezen
id	INT	Da.
group_id	INT	Da.
username	VARCHAR(255)	Da.
email	VARCHAR(255)	Da.
firstname	VARCHAR(255)	Da.
lastname	VARCHAR(255)	Da.
password	CHAR(40)	Da.
created	DATETIME	Da.
modified	DATETIME	Ne.

6.4.1.1.3 Entiteta “acos”



Slika 6.4: Gnezdene množice in vrednosti atributa *lft* in *right* (primer domene elektronskih naprav)

Entiteta “acos” v sodelovanju z entiteto “aros” omogoča nadzor kontrole nad pravicami posameznih uporabnikov. Entiteta “acos” (angl. *Access Control Objects*) vsebuje zapise, ki predstavljajo objekte, nad katerimi izvajamo kontrolo dostopa. Ti objekti predstavljajo kontrolerje in njihove akcije. Objekti so v tabeli urejeni hierarhično. Hierarhija je predstavljena na dva načina. Prvi način obsega uporabo atributa *parent_id*, ki določa starševski objekt. Problem tega načina je preiskovanje glede na celotno pot skozi hierarhično strukturo. Drugi način je metoda gnezdenih množic (angl. *nested set model*), ki uporablja atributa *lft* in *right*. Pri prvi metodi hierarhijo predstavimo kot vozlišča in povezave med njimi. Druga metoda hierarhijo predstavi kot gnezdene množice, kjer atributa *lft* in *right* določata meje vsake posamezne gnezdene množice (glej sliko 6.4). Tako lahko z eno poizvedbo SQL preverimo pravico določenega uporabnika nad določenim objektom.

Tabela 6.5: Relacije entitete “acos”

Relacija	Entiteta 1	Entiteta 2	Kardinalnost
acos-acos	acos	acos	1, n
acos-aros_acos	acos	aros_acos	1, n

Tabela 6.6: Atributi entitete “acos”

Atribut	Podatkovni tip	Obvezen
id	INT	Da.
parent_id	INT	Ne.
model	VARCHAR(255)	Ne.
foreign_key	INT	Ne.
alias	VARCHAR(255)	Ne.
lft	INT	Ne.

rght	INT	Ne.
------	-----	-----

6.4.1.1.4 Entiteta “aros”

Entiteta “aros” (angl. *Access Request Objects*) vsebuje podatke o objektih, ki vršijo dostop. V našem primeru so to uporabniki oz. uporabniške skupine. Vse pravice, ki jih bodo določeni uporabniki lahko izvajali, so vezane na uporabniške skupine.

Tabela 6.7: Relacije entitete “aros”

Relacija	Entiteta 1	Entiteta 2	Kardinalnost
aros-aros	aros	aros	1, n
aros-aros_acos	aros	aros_acos	1, n

Tabela 6.8: Atributi entitete “aros”

Atribut	Podatkovni tip	Obvezen
id	INT	Da.
parent_id	INT	Ne.
model	VARCHAR(255)	Ne.
foreign_key	INT	Ne.
alias	VARCHAR(255)	Ne.
lft	INT	Ne.
rght	INT	Ne.

6.4.1.1.5 Entiteta “aros_acos”

Entiteta “aros_acos” povezuje entiteti “aros” in “acos” ter tako opredeljuje, kateri uporabniki bodo imeli kakšen dostop do akcij kontrolerjev.

Tabela 6.9: Relacije entitete “aros_acos”

Relacija	Entiteta 1	Entiteta 2	Kardinalnost
aros_acos-acos	aros_acos	acos	n, 1
aros_acos-aros	aros_acos	aros	1, n

Tabela 6.10: Atributi entitete “aros_acos”

Atribut	Podatkovni tip	Obvezen
id	INT	Da.
aro_id	INT	Da.

aco_id	INT	Da.
_create	VARCHAR(2)	Da.
_read	VARCHAR(2)	Da.
_update	VARCHAR(2)	Da.
_delete	VARCHAR(2)	Da.

6.4.1.1.6 Entiteta “locations”

Entiteta “locations” vsebuje lokacije, ki jih bodo dodajali uporabniki. Te lokacije predstavljajo spletne strani, kjer se bodo predvajali izbrani videi.

Tabela 6.11: Relacije entitete “locations”

Relacija	Entiteta 1	Entiteta 2	Kardinalnost
users-locations	users	locations	1, n
locations-selected_videos	locations	selected_videos	1, n

Tabela 6.12: Atributi entitete “locations”

Atribut	Podatkovni tip	Obvezen
id	INT	Da.
user_id	INT	Da.
name	VARCHAR(255)	Da.
url	VARCHAR(255)	Da.
image	VARCHAR(255)	Ne.
code	CHAR(40)	Da.
created	INT	Da.

6.4.1.1.7 Entiteta “categories”

Entiteta “categories” vsebuje podatke o kategorijah, katerim pripadajo videi. Kategorije so tudi hierarhično urejene, zato potrebujemo tako atribut `parent_id` kakor tudi atributa `left` in `right`.

Tabela 6.13: Relacije entitete “categories”

Relacija	Entiteta 1	Entiteta 2	Kardinalnost
categories-categories	categories	categories	1, n
categories-	categories	categories_videos	1, n

categories_videos			
-------------------	--	--	--

Tabela 6.14: Atributi entitete “categories”

Atribut	Podatkovni tip	Obvezen
id	INT	Da.
parent_id	INT	Da.
name	VARCHAR(255)	Da.
description	TEXT	Ne.
sort	INT	Da.
lft	INT	Da.
rght	INT	Da.

6.4.1.1.8 Entiteta “categories_videos”

Entiteta “categories_videos” povezuje kategorije in videe.

Tabela 6.15: Relacije entitete “categories_videos”

Relacija	Entiteta 1	Entiteta 2	Kardinalnost
categories_videos-categories	categories_videos	categories	n, 1
categories_videos-videos	categories_videos	videos	n, 1

Tabela 6.16: Atributi entitete “categories_videos”

Atribut	Podatkovni tip	Obvezen
id	INT	Da.
category_id	INT	Da.
video_id	INT	Da.
sort	INT	Da.

6.4.1.1.9 Entiteta “videos”

Entiteta “videos” vsebuje videe, med katerimi bodo uporabniki izbirali.

Tabela 6.17: Relacije entitete “videos”

Relacija	Entiteta 1	Entiteta 2	Kardinalnost
videos-categories_videos	videos	categories_videos	1, n
videos-selected_videos	videos	selected_videos	1, n

Tabela 6.18: Atributi entitete “videos”

Atribut	Podatkovni tip	Obvezen
id	INT	Da.
name	VARCHAR(255)	Da.
flv_file	VARCHAR(255)	Da.
description	TEXT	Ne.
language	VARCHAR(150)	Da.
width	INT	Da.
height	INT	Da.
duration	FLOAT	Ne.
size	INT	Ne.
metadata	TEXT	Ne.
flv_thumb	VARCHAR(255)	Ne.
code	CHAR(40)	Da.
created	DATETIME	Da.

6.4.1.1.10 Entiteta “selected_videos”

Entiteta “selected_videos” vsebuje videe, katere so si uporabniki izbrali za predvajanje na svojih spletnih straneh.

Tabela 6.19: Relacije entitete “selected_videos”

Relacija	Entiteta 1	Entiteta 2	Kardinalnost
selected_videos-videos	selected_videos	videos	n, 1
selected_videos-locations	selected_videos	locations	n, 1
selected_videos-statistics	selected_videos	statistics	1, n

Tabela 6.20: Atributi entitete “selected_videos”

Atribut	Podatkovni tip	Obvezen
id	INT	Da.
video_id	INT	Da.
location_id	INT	Da.
position	VARCHAR(5)	Da.
position_type	VARCHAR(50)	Da.
frequency	INT	Da.

valid_from	DATETIME	Da.
valid_thru	DATETIME	Da.
width	INT	Da.
height	INT	Da.
active_color	CHAR(6)	Da.
inactive_color	CHAR(6)	Da.
border_color	CHAR(6)	Da.
background_color	CHAR(6)	Da.
link_url	VARCHAR(500)	Ne.
code	CHAR(40)	Da.
sort	INT	Da.
created	DATETIME	Da.

6.4.1.1.11 Entiteta “statistics”

Entiteta “statistics” hrani podatke o številu prikazov in klikov posameznega izbranega videa.

Tabela 6.21: Relacije entitete “statistics”

Relacija	Entiteta 1	Entiteta 2	Kardinalnost
statistics-selected_videos	statistics	selected_videos	n, 1

Tabela 6.22: Atributi entitete “statistics”

Atribut	Podatkovni tip	Obvezen
id	INT	Da.
selected_video_id	INT	Da.
created	DATETIME	Da.
action	CHAR(5)	Da.

6.4.1.1.12 Entiteta “pages”

Entiteta “pages” vsebuje podatke o vsebinskih straneh.

Tabela 6.23: Atributi entitete “pages”

Atribut	Podatkovni tip	Obvezen
id	INT	Da.
name	VARCHAR(255)	Da.

alias	VARCHAR(255)	Da.
content	TEXT	Da.
location	VARCHAR(150)	Da.
sort	INT	Da.
homepage	TINYINT	Da.
created	DATETIME	Da.

6.4.1.2 Fizični podatkovni model

Ko imamo izdelani konceptualni in logični podatkovni model, sledi generiranje fizičnega modela. Orodje MySQL Workbench omogoča avtomatsko generiranje fizičnega podatkovnega modela, ki je dejansko skripta SQL. Skripta vsebuje stavke SQL, ki omogočajo fizično generiranje tabel v podatkovni bazi. Primer 6.6 prikazuje stavek SQL, ki fizično ustvari tabelo v podatkovni bazi.

```
CREATE TABLE IF NOT EXISTS `videogadget`.`videos` (
  `id` INT NOT NULL AUTO_INCREMENT ,
  `name` VARCHAR(255) NULL ,
  `flv_file` VARCHAR(255) NULL ,
  `description` TEXT NULL ,
  `language` VARCHAR(150) NULL ,
  `width` INT NULL ,
  `height` INT NULL ,
  `duration` FLOAT NULL ,
  `size` INT NULL ,
  `metadata` TEXT NULL ,
  `flv_thumb` VARCHAR(255) NULL ,
  `code` CHAR(40) NULL ,
  `created` DATETIME NULL ,
  PRIMARY KEY (`id`) )
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_general_ci;
```

Primer 6.6: Stavek SQL, ki ustvari tabelo "videos" v podatkovni bazi

6.4.2 Sloj dostopa do podatkov

Dostop do podatkov izvajamo s pomočjo t. i. modelov. Model v ogrodju Cake je razred, ki podeduje attribute in metode razreda, imenovanega AppModel, ki pa podobno deduje od osnovnega razreda Model. V osnovi model predstavlja podatke, ki so vsebovani v tabeli podatkovne baze. Vendar v splošnem ni vezan samo na podatkovne baze, ampak podatke lahko črpa tudi iz datotek datotečnega sistema, datotek XML ali datotek CSV itd. Model kreiramo tako,

da ustvarimo razred, poimenovan glede na konvencije ogrodja Cake, npr. **Video**. Razred je ustvarjen v datoteki “video.php”, ki se nahaja v direktoriju “models”. Primer 6.7 predstavlja deklaracijo razreda **Video**, ki zadostuje za uporabo v aplikaciji. S tem pridobimo naslednje funkcionalnosti:

- branje podatkov,
- kreiranje novih zapisov,
- spreminjanje obstoječih podatkov,
- brisanje zapisov.

```
class Video extends AppModel {
    var $name = 'Video';
}
```

Primer 6.7: Deklaracija razreda Video, ki predstavlja tabelo “videos” v podatkovni bazi

Modeli v ogrodju Cake te funkcionalnosti razširjajo še z dvema pomembnima in izredno uporabnima dodatkom, in sicer možnostjo definiranja tipov atributov, ki jih tabela vsebuje, in možnostjo določanja relacij z ostalimi modeli, ki jih uporabljamo v aplikaciji. Tako lahko popolnoma preslikamo konceptualni podatkovni model v sloj dostopa do podatkov. Z definiranjem tipov atributov nam model omogoča avtomatsko validiranje podatkov med vnosom v tabelo (glej primer 6.8). Če definiramo relacije v modelu, potem lahko pri branju podatkov tega modela avtomatsko pridobimo tudi podatke modelov, s katerimi je ta model v relaciji. Določimo lahko naslednje relacije:

- “one-to-many” (glej primer 6.9),
- “many-to-one” (glej primer 6.10),
- “many-to-many” (glej primer 6.11).

```
var $validate = array(
    'name' => array('notempty' => array('rule' => 'notempty')),
    'flv_file' => array('notempty' => array('rule' => 'notempty')),
    'language' => array('notempty' => array('rule' => 'notempty')),
    'width' => array('numeric' => array('rule' => 'numeric')),
    'height' => array('numeric' => array('rule' => 'numeric')),
    'duration' => array('numeric' => array('rule' => 'numeric')),
```

```
'size' => array('numeric' => array('rule' => 'numeric')),
'code' => array('notempty' => array('rule' => 'notempty'))
);
```

Primer 6.8: Definicija tipov atributov za model Video

```
var $hasMany = array(
    'SelectedVideo' => array(
        'className' => 'SelectedVideo',
        'foreignKey' => 'location_id',
        'dependent' => false
    )
);
```

Primer 6.9: Deklaracija relacije “one-to-many” v modelu Video

```
var $belongsTo = array(
    'Group' => array(
        'className' => 'Group',
        'foreignKey' => 'group_id'
    )
);
```

Primer 6.10: Deklaracija relacije “many-to-one” v modelu User

```
var $hasAndBelongsToMany = array(
    'Category' => array(
        'className' => 'Category',
        'joinTable' => 'categories_videos',
        'foreignKey' => 'video_id',
        'associationForeignKey' => 'category_id',
        'unique' => true
    )
);
```

Primer 6.11: Deklaracija relacije “many-to-many” modela Video

Za potrebe naše aplikacije smo kreirali modele za vsako posamezno tabelo iz podatkovne baze. Vsakemu modelu smo določili tipe atributov in relacije z ostalimi modeli. Omeniti je še potrebno dve posebnosti pri kreiranju modelov. Model Category, ki predstavlja kategorije, mora omogočati hierarhično pridobivanje podatkov iz tabele. Za ta namen Cake omogoča uporabo dodatka, ki razširi funkcionalnosti modela. Dodatki, ki razširjajo model, se imenujejo “behaviours”. Ta specifičen dodatek omogoča pridobivanje zapisov iz tabele v obliki drevesa. Vključitev dodatka prikazuje primer 6.12. Druga posebnost pa omogoča filtriranje modelov, v okviru določenega modela, ki ima definirane relacije z ostalimi modeli. V primeru modela Video lahko pridobimo

samo podatke, ki jih določa relacija z modelom Category, ne pa podatke, ki so določeni z relacijo z modelom SelectedVideo. Vključitev tega dodatka izvedemo na podoben način kot pri primeru 6.12 z razliko, da je ta dodatek poimenovan “Containable”.

```
var $actsAs = array('Tree');
```

Primer 6.12: Vključitev dodatka “Tree” v modelu Category

6.4.3 Sloj poslovne logike

Sloj poslovne logike se v vzorcu MVC imenuje kontroler. Namen kontrolerja je povezovanje modelov in pogledov. Kontroler z uporabo modelov pridobiva podatke, jih obdela in nato prikaže preko pogledov. Kontrolerji so, podobno kot modeli, razredi, ki se nahajajo v direktoriju “controllers”. Če poimenovanje kontrolerjev izhaja iz imena modela, potem ogrodje Cake avtomatsko poveže model s tem kontrolerjem. V primeru kontrolerja VideosController bo ogrodje Cake s tem kontrolerjem povežalo model Video. Kontrolerji definirajo metode, ki predstavljajo akcije v okviru aplikacije. Uporabniki z uporabo teh akcij dosežejo želene rezultate.

Pri opisu posameznega kontrolerja se bomo omejili na tri elemente, in sicer: implementirane akcije, uporabljene kontrolerje in potrebne pomočnike (angl. *helpers*).

Komponente v ogrodju Cake so posebni razredi, ki razširjajo funkcionalnosti kontrolerjev. Isto komponento lahko uporabimo v večih kontrolerjih. Komponente poimenujemo poljubno, vendar se je potrebno držati konvecij poimenovanja razredov. Na koncu imena je potrebno dodati besedo “Component”. Komponente shranjujemo v direktoriju “controllers/components”.

Pomočniki so podobno posebni razredi, ki pa razširjajo funkcionalnosti pogledov v ogrodju Cake. V osnovi Cake vsebuje pomočnike, ki olajšajo prikaz določenih elementov dokumenta XHTML, obrazcev, vključitev kode JavaScript in druge. Poimenujejo se v skladu s konvencijo, na koncu pa je potrebno dodati besedo “Helper”. Pomočnike shranjujemo v direktoriju “views/helpers”.

6.4.3.1 Komponente

6.4.3.1.1 Avtentikacija in avtorizacija

Za avtentikacijo in avtorizacijo uporabnikov bomo uporabili dodatne elemente ogrodja Cake, ki te funkcionalnosti omogočajo. V osnovi potrebujemo dve komponenti. Prva, imenovana AuthComponent, omogoča avtentikacijo uporabnikov. Torej, uporabniki se lahko prijavijo in odjavijo iz aplikacije. Druga, imenovana AclComponent, pa omogoča preverjanje pravic za

dostop z uporabo t. i. seznama dostopnih pravic (angl. *Access Control List*). ACL upravlja z dvema tipoma objektov, in sicer tipom objektov, ki zahtevajo dostop ter tipom objektov, do katerih se zahteva dostop. Objekti, ki zahtevajo dostop, so v večini primerov uporabniki aplikacije, tako je tudi v primeru naše aplikacije. Po drugi strani pa so objekti, do katerih se zahteva dostop, lahko katerekoli entitete, katerim želimo omejiti dostop oz. ta dostop nadzorovati. Naša aplikacija bo nadzorovala dostop do akcij kontrolerjev. Seznam ARO in ACO sta implementirana v obliki drevesa. Tako lahko pravice za dostop enostavno grupiramo glede na objekte ARO ali objekte ACO, zato tudi tabeli “aros” in “acos” vsebujeta attribute, ki omogočajo enostavno poizvedovanje po drevesni strukturi (atributi `parent_id`, `lft` in `right`). Tabela “aros_acos”, ki povezuje tabeli “aros” in “acos”, definira pravice, ki jih ima določen objekt ARO nad objektom ACO. Na razpolago imamo 4 vrste pravic, in sicer: kreiranje, branje, spreminjanje in brisanje [12]. Pravice uporabnikov naše aplikacije bomo dodelili glede na skupino, kateri pripadajo. Uporabniki v skupini Administratorji imajo pravice uporabljati katerokoli akcijo. Uporabnikom v skupini Uporabniki na začetku onemogočimo uporabo vseh akcij, nato pa jim dovolimo uporabo samo tistih akcij, katere potrebujejo, da lahko uporabljajo aplikacijo.

V osnovi naša aplikacija loči med dvema tipoma uporabnikov. Prvi so uredniki, ki imajo popoln nadzor nad aplikacijo. Drugi pa so uporabniki, ki aplikacijo uporabljajo za svoje potrebe. Ogradje Cake omogoča enostavno ločevanje teh dveh skupin uporabnikov, in sicer tako, da ustvari dva segmenta aplikacije. Segmenta ločimo glede na strukturo naslova URL. Do akcij kontrolerjev, ki so namenjene samo urednikom, dostopamo preko naslovov URL, ki vsebujejo fragment “admin”. Na primer obe skupini uporabnikov lahko urejajo podatke uporabnikov. Uredniki lahko spreminjajo podatke kateregakoli uporabnika v aplikaciji, uporabnik pa lahko spreminja samo svoje podatke. Tako urednik uporabi naslednji naslov:

`http://<domena>/admin/users/edit/<id_uporabnika>`,

uporabnik pa naslov:

`http://<domena>/users/edit`.

Fragment “admin” v naslovu URL bo določil klic metode “admin_edit” v kontrolerju `UsersController`, medtem ko bo uporabniški naslov sprožil klic metode “edit” v istem kontrolerju.

6.4.3.1.2 FLV Metadata

Glede na primer uporabe 5.3.31 je potrebno ob kreiranju novega videa določiti tudi meta podatke, ki opisujejo ta video. Za branje teh podatkov je potrebno razumeti zgradbo datoteke FLV, ki

podatke vsebuje. Komponenta FlvMetadata upošteva zgradbo datoteke FLV in omogoča branje meta podatkov iz nje.

Tabela 6.24: Atributi komponente “FlvMetadata”

Atribut	Opis
VCidMap	Definira preslikavo med šifro video kodeka in njegovim imenom
ACidMap	Definira preslikavo med šifro avdio kodeka in njegovim imenom.

Tabela 6.25: Metode komponente “FlvMetadata”

Metoda	Parametri	Opis
getMetaData	string, string	Preveri tip datoteke in prebere meta podatke.
bin2Double	string	Pretvori binarni zapis števila v dvojni natančnosti (IEEE 754) v število v plavajoči vejici (float).

6.4.3.1.3 FLV Upload

Ogrodje Cake ne vključuje elementov, ki bi omogočili nalaganje datotek na strežnik, zato je bilo potrebno to razviti od začetka. Komponenta FlvUpload omogoča upravljanje datotek FLV na strežniku.

Tabela 6.26: Metode komponente “FlvUpload”

Metoda	Parametri	Opis
add	array, Model, string, string	Omogoča nalaganje novega videa na strežnik. Pri tem preveri tip datoteke, ki jo nato shrani na strežnik. V primeru, da datoteka z istim imenom že obstaja, novo datotekko preimenuje.
edit	array, Model, string, string	Pri urejanju obstoječega videa omogoča nalaganje novega videa, tako da prejšnega pobriše.
delete	string, string	Pobriše datoteko iz strežnika.
getFileExtension	string	Iz imena datoteke izlušči tip datoteke glede na končnico.

6.4.3.1.4 Ostranjevanje

Pri prikazu vseh seznamov v aplikacij je potrebno uporabiti ostranjevanje. V primeru, da seznam vsebuje več kot 20 elementov, je potrebno te elemente prikazati na večih straneh. Na primer, če seznam vsebuje 90 elementov, bomo prikaz celotnega seznama razbili na 5 strani, kjer bo vsaka stran vsebovala največ 20 elementov. Za ta namen ogrodje Cake vsebuje posebno komponento, ki omogoča takšno razbitje. Uporaba je izredno enostavna, v večini primerov je potrebno samo izvesti klic metode “paginate”, ki nato poskrbi za celotno ostranjevanje.

6.4.3.1.5 Email

Naša aplikacija mora biti sposobna pošiljanja email sporočil uporabnikom. Na primer, pri registraciji uporabnika je potrebno novemu uporabniku poslati sporočilo o uspešni registraciji. To omogoča komponenta Email, ki je del ogrodja Cake.

6.4.3.2.6 Cookie

Ogrodje Cake vsebuje komponento Cookie, ki omogoča upravljanje s piškotki. Piškotki so posebne datoteke, ki so shranjene na strani klienta. Aplikacija uporablja piškotke za shranjevanje podatkov, ki so vezani na klineta. Na primer zaporedno število predvajanja za določenega obiskovalca. Komponenta omogoča branje in kreiranje piškotkov.

6.4.3.2 Kontrolerji

6.4.3.2.1 Groups

Za potrebe upravljanja skupin smo kreirali kontroler GroupsController. Kontroler deklarira pomočnika “Format”.

Tabela 6.27: Metode kontrolerja “GroupsController”

Metoda	Parametri	Opis
admin_index		Omogoča prikaz seznama skupin z uporabo ostranjevanja.
admin_view	int	Omogoča prikaz atributov posamezne skupine.
admin_edit	int	Omogoča urejanje atributov skupine.

6.4.3.2.2 Users

Kontroler UsersController omogoča upravljanje z uporabniki ter izvaja avtentikacijo in avtorizacijo. Za delovanje uporablja komponento "Email" in deklarira pomočnika "Format".

Tabela 6.28: Metode kontrolerja "UserController"

Metoda	Parametri	Opis
admin_login		Obvesti urednika o uspešni prijavi.
admin_logout		Obvesti urednika o uspešni odjavi in ga preusmeri na domačo stran.
admin_index		Omogoča prikaz seznama uporabnikov z uporabo odstranjevanja.
admin_view	int	Prikaže vrednosti atributov posameznega uporabnika.
admin_add		Omogoča dodajanje novega uporabnika.
admin_edit	int	Omogoča spreminjanje atributov posameznega uporabnika.
admin_delete	int	Pobriše posameznega uporabnika.
login		Obvesti trenutnega uporabnika o uspešni prijavi.
logout		Obvesti trenutnega uporabnika o uspešni odjavi in ga preusmeri na domačo stran.
index		Prikaže nadzorno ploščo trenutnega uporabnika.
edit		Omogoča spreminjanje atributov trenutnega uporabnika.
add		Omogoča registracijo novega uporabnika.

Pri registraciji novega uporabnika (metoda "add") je potrebno inkriptirati geslo, ki ga posreduje uporabnik. Inkripcijo gesla izvede metoda "password" komponente AuthComponent. Za inkripcijo uporabljamo zgoščevalno funkcijo SHA-1. Ko novega uporabnika shranimo v podatkovno bazo, mu je potrebno poslati email, ki vsebuje pozdravno sporočilo. Email pošljemo z uporabo komponente EmailComponent. Definirati je potrebno prejemnika, pošiljatelja, naslov sporočila in vsebino. Vsebino določa posebna predloga, ki se nahaja v direktoriju "views/elements/email".

6.4.3.2.3 Categories

Kontroler `CategoriesController` omogoča upravljanje s kategorijami. V uporabniškem delu aplikacije skrbi za prikaz hierarhije kategorij. Deklarira pomočnika "Format".

Tabela 6.29: Metode kontrolerja "CategoriesController"

Metoda	Parametri	Opis
<code>view</code>		Prikaže posamezno kategorijo in vse videe, ki pripadajo tej kategoriji.
<code>treemenu</code>		Prikaže hierarhično navigacijo kategorij.
<code>admin_index</code>		Omogoča prikaz seznama kategorij z uporabo odstranjevanja.
<code>admin_view</code>	<code>int</code>	Prikaže vrednosti atributov posamezne kategorije.
<code>admin_add</code>		Omogoča dodajanje nove kategorije.
<code>admin_edit</code>	<code>int</code>	Omogoča spreminjanje atributov posamezne kategorije.
<code>admin_delete</code>	<code>int</code>	Pobriše posamezno kategorijo.
<code>admin_updatesort</code>		Spremeni vrstni red prikaza kategorij.

Pri prikazu hierarhije kategorij je potrebno uporabiti "behaviour" modela, imenovanega "Tree". S klicem metode "generatetreelist" v dodatku "Tree" pridobimo hierarhično strukturiran seznam vseh kategorij. Na seznamu je globina posamezne kategorije predstavljena s toliko podčrtaji, kolikor je njena globina. Podčrtaji niso primerni za prikaz na spletni strani, zato jih je potrebno zamenjati s presledki. Tu si pomagamo z regularnim izrazom.

6.4.3.2.4 Videos

Kontroler `VideosController` omogoča upravljanje z videi. V uporabniškem delu aplikacije skrbi za prikaz videa. Deklarira pomočnike "Format", "FlvUpload" in "Javascript". Uporablja komponente "FlvUpload" in "FlvMetadata".

Tabela 6.30: Metode kontrolerja "VideosController"

Metoda	Parametri	Opis
<code>view</code>	<code>int</code>	Prikaže posamezni video in omogoča predvajanje tega videa.
<code>index</code>		Prikaže seznam zadnjih 6 dodanih videov.

frontplay	string	Omogoča prenos video datoteke za predvajanje na naši spletni strani.
play	string	Omogoča prenos video datoteke za predvajanje na gostujoči spletni strani in zabeleži predvajanje v statistiko.
click	string	Beleži klik na video v gostujoči spletni strani.
admin_index		Omogoča prikaz seznama videov z uporabo odstranjevanja.
admin_view	int	Prikaže vrednosti atributov posameznega videa.
admin_add		Omogoča dodajanje novega videa.
admin_edit	int	Omogoča spreminjanje atributov posameznega videa.
admin_delete	int	Pobriše posamezni video.
admin_createthumb	int	Prikaže obrazec za določanje sličice videa.
admin_savethumb	int	Shrani sličico na strežniku.

Komponenti FlvMetadata in FlvUpload uporabimo v metodi “admin_add” in “admin_edit”. Z uporabo metode “add” oz. “edit” komponente FlvUpload naložimo datoteko na strežnik. Metoda nam vrne ime datoteke na strežniku. Vse video datoteke shranjujemo v direktoriju “uploads/videos”. Tako dosežemo, da te datoteke niso prosto dostopne, ampak jih lahko edino prenašamo s posredovanjem aplikacije. Ko smo datoteko naložili, preberemo meta podatke s komponento FlvMetadata.

Metode “admin_play”, “frontplay” in “play” prenašajo datoteke iz strežnika in posledično omogočajo njihovo predvajanje. Metoda “play” pred prenosom zabeleži predvajanje v statistiko za ta posamezni video. Primer 6.13 prikazuje programsko kodo, ki omogoča prenos datoteke iz strežnika. Najprej nastavimo atribut v glavi HTTP odziva, ki določa vrsto datoteke, ki se bo prenesla. Atribut “Content-type” nastavimo na vrednost “video/x-flv”. Ta vrednost je določena v okviru standarda MIME (angl. *Multipurpose Internet Mail Extensions*). Standard določa način opisa vsebine, ki se prenaša preko HTTP protokola. Naslednji korak predstavlja branje video datoteke in izpis prebrane vsebine na izhod. Konstanta pri klicu funkcije “fread” predstavlja dolžino prebrane vsebine v bajtih. V vsakem koraku preberemo 8192 bajtov iz datoteke in jih zapišemo na izhod oz. pošljemo uporabniku.

```
header('Content-type: video/x-flv');
$fh = fopen(Configure::read('APP_UPLOAD_DIR').'videos/');
```

```

        $video['Video']['flv_file'],'rb');
fseek($fh, 0);
while (!feof($fh)){
    print(fread($fh, 8192));
}
fclose($fh);

```

Primer 6.13: Prenos video datoteke iz strežnika

Sličico, ki predstavlja video, ustvarimo v posebej za ta namem pripravljenem predvajalniku. Urednik predvaja video in, ko je zadovoljen s trenutno prikazano sliko, pritisne na tipko “S”. Predvajalnik vzame trenutno prikazano sliko, ki jo nato zakodira v formatu BASE64 in pošlje aplikaciji (s klicem metode “admin_savethumb”). Aplikacija prejeto sličico dekodira in jo shrani na strežniku. BASE64 je posebna kodirna shema, ki je bila zasnovana za kodiranje binarnih podatkov in prenos teh podatkov preko kanala, ki je v osnovi namenjen prenosu tekstualnih podatkov.

6.4.3.2.5 CategoriesVideos

Kontroler CategoriesVideosController poskrbi za povezavo med kategorijami in videi.

Tabela 6.31: Metode kontrolerja “CategoriesVideosController”

Metoda	Parametri	Opis
admin_updatesort		Omogoča določanje vrstnega reda videov v okviru posamezne kategorije.

6.4.3.2.6 Locations

Kontroler LocationsController omogoča upravljanje z lokacijami. Deklarira pomočnika “Format” in “Javascript”. Uporablja komponento “Cookie”.

Tabela 6.32: Metode kontrolerja “LocationsController”

Metoda	Parametri	Opis
play	string	Omogoča predvajanje videov, ki so določeni v posamezni lokaciji. Pri tem upošteva parametre posameznega videa in vrstni red videov v okviru lokacije.
index		Prikaže seznam lokacij z uporabo odstranjevanja.
view	int	Prikaže posamezno lokacijo in seznam

		videov, ki pripadajo tej lokaciji.
add		Omogoča dodajanje nove lokacije.
edit	int	Omogoča spreminjanje obstoječe lokacije.
delete	int	Pobriše posamezno lokacijo.

Metoda “play” preveri vrednosti vseh parametrov, preden določi, kateri izbrani video se bo predvajal. Vrednosti parametrov pridobi iz piškotka, ki jih nato primerja z vrednostmi, ki jih je določil uporabnik. Če vrednosti parametrov omogočajo predvajanje izbranega videa za to lokacijo, potem je naslednji korak sestava skripte, ki bo omogočila predvajanje videa na gostujoči spletni strani.

Metoda “add” kreira novo lokacijo, pri tem pa določi vrednost atributa “code”. Ta atribut uporabljamo za generiranje kode (glej primer 6.14), ki jo uporabnik doda v dokument XHTML svoje spletne strani. Vrednost atributa generiramo z uporabo zgoščevalne funkcije SHA-1, kateri kot parameter posredujemo vrednost primarnega ključa.

```
<script type="text/javascript"
  src="http://<domena>/selected_videos/play/
  0ade7c2cf97f75d009975f4d720d1fa6c19f4897"></script>
```

Primer 6.14: Koda, ki jo uporabnik vključi v svojo spletno stran

6.4.3.2.7 SelectedVideos

Kontroler SelectedVideosController omogoča upravljanje z izbranimi videi. Deklarira pomočnike “Format”, “Javascript”, “ColorPicker” in “DatetimePicker”. Uporablja komponento “Cookie”.

Tabela 6.33: Metode kontrolerja “SelectedVideosController”

Metoda	Parametri	Opis
play	string	Omogoča predvajanje izbranega videa. Pri tem upošteva parametre posameznega videa. Hkrati zabeleži predvajanje v statistiko.
preview	int	Omogoča predogled izbranega videa v gostujoči spletni strani.
index		Prikaže seznam izbranih videov z uporabo odstranjevanja.
view	int	Prikaže posamezni izbrani video.
add		Omogoča dodajanje novega izbranega videa.
edit	int	Omogoča spreminjanje obstoječega izbranega videa.

delete	int	Pobriše posamezni izbrani video.
--------	-----	----------------------------------

Metoda “play” generira skripto, ki omogoča vključevanje predvajalnika v gostujočo spletno stran. Preden začne generirati skripto, preveri vrednosti parametrov iz piškotka in tiste vrednosti, ki jih je določil uporabnik.

Metoda “preview” omogoča predogled vključenega predvajalnika v gostujočo spletno stran. Za ta namen je potrebno spremeniti strukturo dokumenta HTML, ki predstavlja gostujočo spletno stran. Pred zaključkom elementa `<body>` je potrebno dodati kodo JavaScript, ki bo omogočala vključitev predvajalnika. Pomagamo si z razredom `DomDocument`, ki je del osnovne knjižnice PHP. Razred naloži gostujočo stran in zgradi model dokumentnega objekta. V generirano strukturo dodamo nov element, imenovan “script”. Primer 6.15 prikazuje rezultat dodajanja. Ker bomo gostujočo spletno stran prenesli iz našega strežnika, je potrebno dodati tudi element “base”. Ta element bo omogočil prikaz vseh zunanjih elementov gostujoče strani, npr. slike.

```
<script type="text/javascript"
  src="http://<domena>/selected_videos/play/
  0ade7c2cf97f75d009975f4d720d1fa6c19f4897/demo"></script>
```

Primer 6.15: Vključen element “script” v gostujoči spletni strani



Slika 6.5: Kontrole v predvajalniku



Slika 6.6: Prikazan gumb, če vključitev ni možna

Metoda “add” omogoča dodajanje novega izbranega videa. Pri tem uporabnik določi barve uporabniškega vmesnika za predvajalnik. V bistvu določi barve treh gumbov predvajalnika (glej sliko 6.5) in enega gumba, ki je prikazan, ko parametri ne dovoljujejo vključitev predvajalnika (glej sliko 6.6). Za generiranje tega gumba uporabimo posebne funkcije PHP iz knjižnice GD. Knjižnica GD vsebuje določene funkcije, ki omogočajo delo s slikami. Ko je gumb kreiran, ga shranimo na strežniku, zato da ga ni potrebno ponovno generirati, ko je zahtevan s strani klienta.

Metoda “add” tudi določi vrednost atributa “code”. Ta atribut uporabljamo za generiranje kode, ki jo uporabnik doda v dokument XHTML svoje spletne strani. Vrednost atributa generiramo z uporabo zgoščevalne funkcije SHA-1, kateri kot parameter posredujemo vrednost primarnega ključa.

6.4.3.2.8 Statistics

Kontroler `StatisticsController` skrbi za prikaz statistike. Deklarira pomočnike “Format”, “Javascript” in “DatetimePicker”.

Tabela 6.34: Metode kontrolerja “StatisticsController”

Metoda	Parametri	Opis
view	int	Prikaže obrazec za določanje prikaza statistike. Prikaže statistiko.
spreadsheet		Omogoča izvoz statistike v preglednico, ki je primerna za uporabo v programu Microsoft Excel in OpenOffice Spreadsheets.
svg		Omogoča izvoz grafa statistike v obliki slike (format SVG).

Metoda “spreadsheet” omogoča prenos datoteke, ki vsebuje statistične podatke v obliki preglednice. Za ta namem je potrebno nastaviti dva parametra v glavi HTTP odziva. Vrednost parametra “Content-type” mora biti nastavljena na “application/vnd.ms-excel”, vrednost parametra “Content-Disposition” pa “attachment; filename=statistics.xls” (glej primer 6.16).

```
header('Content-type: application/vnd.ms-excel');
header('Content-Disposition:attachment;filename=statstics.xls');
```

Primer 6.16: Vrednosti dveh parametrov, ki omogočajo prenos preglednice

Podobno velja za metod “svg”, ki omogoča prenos grafa statistike v obliki slike. Format datoteke, ki vsebuje grafične elemente je SVG. Prime 6.17 prikazuje parametre, ki omogočajo prenos datoteke v formatu SVG.

```
header("Content-type: image/svg+xml");
header('Content-Disposition:attachment;filename=statstics.svg');
```

Primer 6.17: Vrednosti parametrov, ki omogočajo prenos slike v formatu SVG.

6.4.3.2.9 Pages

Kontroler `StatisticsController` skrbi za prikaz statistike. Deklarira pomočnike “Format”,

“Javascript” in “DatetimePicker”.

Tabela 6.35: Metode kontrolerja “PagesController”

Metoda	Parametri	Opis
topmenu		Prikaže navigacijo, ki se uporablja v glavi spletne strani.
footer		Prikaže navigacijo, ki se uporablja v nogi spletne strani.
display	string	Omogoča prikaz vsebine posamezne strani.
admin_sethomepage	int	Izbrano stran nastavi kot domačo stran.
admin_index		Prikaže seznam vsebinskih strani z uporabo odstranjevanja.
admin_view	int	Prikaže attribute posamezne vsebinske strani.
admin_add		Omogoča dodajanje nove vsebinske strani.
admin_edit	int	Omogoča urejanje atributov posamezne vsebinske strani.
admin_delete	int	Pobriše obstoječo vsebinsko stran.

6.5 Sloj uporabniškega vmesnika

Sloj uporabniškega vmesnika je edini in najvplivnejši del aplikacije, s katerim obiskovalci in uporabniki interagirajo. Zato je potrebno veliko pozornosti posvetiti izgradnji uporabniškega vmesnika. Vmesnik mora omogočati vse funkcionalnosti, ki so zahtevane s strani aplikacije, ob tem pa mora biti prijazen do obiskovalca oz. uporabnika. To pomeni, da morajo biti funkcionalnosti implementirane tako, da je izvajanje teh funkcionalnosti čim lažje in učinkovitejše.

V ogrodju Cake ima vsaka posamezna akcija v kontrolerju določen pogled, ki definira prikaz podatkov iz tega kontrolerja. Med izvajanjem akcije kontrolerja določimo vrednosti spremenljivk, ki jih nato uporabimo med generiranjem prikaza. Te spremenljivke vsebujejo podatke, ki jih je kontroler pridobil iz modela in dodatno obdelal. Poglede poimenujemo glede na ime akcije, kateri ustrezajo, pri tem upoštevamo konvencije poimenovanja dodatotek. Pogledi vseh akcij določenega kontrolerja so shranjeni v istem direktoriju, ki ustreza temu kontrolerju. Če kontroler VideosController vsebuje akcije `admin_view`, `index` in `editVideo`, potem se v direktoriju “`views/videos`” nahajajo pogledi `admin_view.php`, `index.php` in `edit_video.php`.

6.5.1 Elementi

Velikokrat je potrebno določene elemente spletne strani uporabiti večkrat. Za ta namen ima ogrodje Cake izdelan mehanizem, ki se imenuje “elements”. Elemente definiramo v direktoriju “views/elements” in jih lahko uporabljamo v več pogledih. Naša aplikacija uporablja dve različni postavitvi za uporabniški del aplikacije. Prva določa postavitev za obiskovalce, druga pa za uporabnike. Da ne bi podvajali istih elementov, uporabimo mehanizem “elements”. Definiramo dva elementa, in sicer “topmenu” ter “footer”. Prvi definira navigacijo v glavi strani, drugi pa v nogi. Primer 6.18 prikazuje uporabo in vključitev elementa v pogled.

```
print $this->element('footer');
```

Primer 6.18: Vključitev elementa “footer”

6.5.2 Pomočniki

Mehanizem elementov je zelo uporaben, vendar včasih potrebujemo močnejše orodje. Elementi definirajo posebne fragmente, ki jih nato vključujemo v poglede in postavitve. Ti fragmenti so deli kode XHTML. Velikokrat pa ta mehanizem ne zadostuje, kajti ne omogoča kompleksnejših obdelav podatkov. Ogrodje Cake ima implementiran mehanizem pomočnikov (angl. *helpers*), ki razširjajo poglede in omogočajo zahtevnejšo obdelavo podatkov. Pomočnik je definiran kot posebni razred, ki deduje od razreda AppHelper in je shranjen v direktoriju “views/helpers”. Če želimo določenega pomočnika uporabiti v pogledu, ga je potrebno deklarirati v kontrolerju, kjer se nahaja akcija.

6.5.2.1 Format

Pomočnik “Format” je namenjen oblikovanju posebnih vrednosti atributov.

Tabela 6.36: Metode pomočnika “Format”

Metoda	Parametri	Opis
date	string, string	Omogoča oblikovanje izpisa za vrednosti, ki predstavljajo datum (brez časovne komponente).
datetime	string, string	Omogoča oblikovanje izpisa za vrednosti, ki predstavljajo datum in čas.
filesize	int, string, bool	Omogoča oblikovanje izpisa vrednosti, ki predstavlja velikost datoteke v bajtih.
playtime	float, string, bool	Omogoča oblikovanje izpisa vrednosti, ki predstavlja dolžino trajanja videa (v

		sekundah).
--	--	------------

6.5.2.2 FLV Upload

Pomočnik “FlvUpload” skrbi za pravilni prikaz atributa, namenjenega video datoteki.

Tabela 6.37: Metode pomočnika “FlvUpload”

Metoda	Parametri	Opis
view	string, Model	Namenjena je oblikovanju prikaza v pogledu, ki določa pregled posameznega videa. Prikaže povezavo na datoteko in omogoči zagon predvajanja.
add	string	Oblikuje prikaz atributa v obrazcu za dodajanje novega videa. Prikaže samo polje obrazca, ki omogoča nalaganje datoteke na strežnik.
edit	string, Model	Oblikuje prikaz atributa v obrazcu za urejanje videa. Prikaže polje, ki omogoča brisanje obstoječe dateke. Prikaže tudi polje za nalaganje nove datoteke in povezavo do obstoječe datoteke.

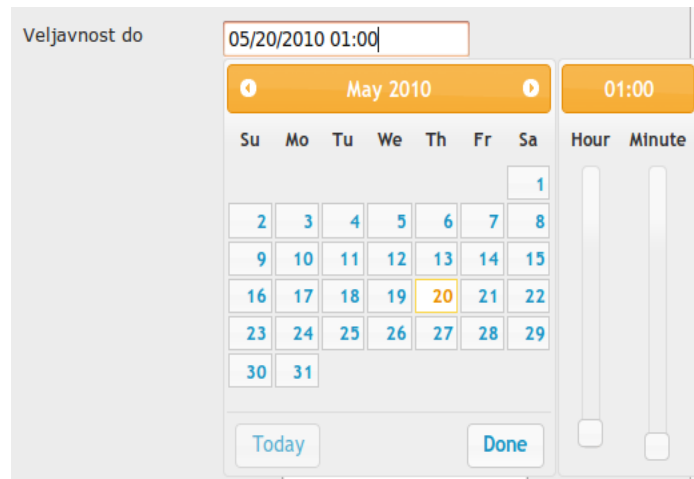
Povezava do obstoječe datoteke tudi omogoča predvajanje videa. Ko urednik klikne na to povezavo, se video prične predvajati v plasti nad stranjo. To dosežemo z uporabo posebne knjižnice jQuery in dodatka k tej knjižnici, imenovanega FancyBox.

6.5.2.3 DatetimePicker

Pomočnik “DatetimePicker” je namenjen oblikovanju atributov, ki so namenjeni določanju datumov.

Tabela 6.38: Metoda pomočnika “DatetimePicker”

Metoda	Parametri	Opis
input	string, Form, array, string	Urednikom in uporabnikom omogoča določanje datuma z uporabo koledarja. Prikaže se koledar, kjer uporabniki izberejo želen datum in uro.



Slika 6.7: Uporaba pomočnika "DatetimePicker"

Z uporabo takšnega pristopa uporabnikom olajšamo izbiro datuma in ure, hkrati pa sta format in vrednost datuma pravilna. Slika 6.7 prikazuje uporabo tega pomočnika. Za implementacijo tega pomočnika potrebujemo knjižnico jQuery in dodatek TimePicker.

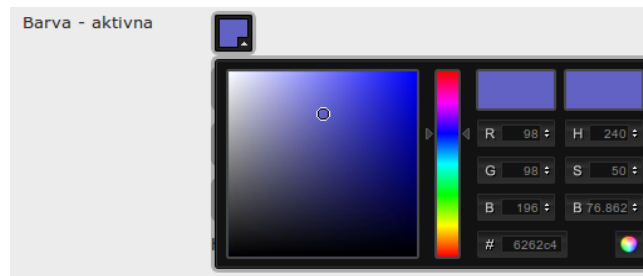
6.5.2.4 ColorPicker

Pomočnik "ColorPicker" je namenjen oblikovanju atributov, ki so namenjeni določanju barv.

Tabela 6.39: Metoda pomočnika "ColorPicker"

Metoda	Parametri	Opis
input	string, Form, array, string	Uporabnikom omogoča določanje barv.

Z uporabo takšnega pristopa uporabnikom olajšamo izbiro barv, hkrati pa sta format in vrednost barve pravilna. Slika 6.8 prikazuje uporabo tega pomočnika. Brez uporabe tega pomočnika bi morali uporabniki vpisovati vrednosti barv v formatu RGB (angl. *Red-Green-Blue*). Za implementacijo tega pomočnika potrebujemo knjižnico jQuery in dodatek ColorPicker.



Slika 6.8: Uporaba pomočnika "ColorPicker"

6.5.2.5 Form

Kreiranje obrazcev je časovno zelo potratno delo. Ogrodje Cake nam priskoči na pomoč pri ustvarjanju obrazcev s pomočnikom, imenovanim Form. Za večino navadnih obrazcev zadostuje uporaba posebne metode pomočnika, kjer samo definiramo ime atributa, za vse ostalo poskrbi metoda. Primer 6.19 prikazuje uporabo metode "input" za kreiranje obrazca, ki omogoča dodajanje nove lokacije. Metoda sprejema tudi dodatne parametre, s katerimi lahko specializiramo prikaz atributa v obrazcu. Tako lahko določimo drugačen naziv atributa (angl. *label*), sporočilo v primeru napake (angl. *error*) itd.

Tabela 6.40: Najpomembnejše metode pomočnika "Form"

Metoda	Opis
input	Kreira prikaz atributov glede na tip, pridobljen iz tabele v podatkovni bazi.
create	Pripravi začetek obrazca. Določi tip MIME za podatke, ki jih bo obrazec posredoval in naslov URL, kamor se bodo podatki poslali.
end	Zaključi obrazec. Po potrebi prikaže gumb, s katerim uporabnik potrdi obrazec.
submit	Prikaže gumb za potrditev obrazca.
button	Prikaže poljuben gumb.

```
<div class="locations form">
<?php echo $this->Form->create('Location');?>
    <?php
        echo $this->Form->input('name', array(
            'div' => array('id' => 'LocationName'),
            'label' => __('Ime', true),
            'error' => array(
                'notempty' => __('To polje je obvezno.', true)));
        echo $this->Form->input('url', array(
```

```

        'label' => __('URL', true),
        'between' =>
            '<span class="before-input">http://</span>',
        'error' => array('isUrl' => __('Vnesite veljaven URL.',
true)))));
?>
<?php echo $this->Form->end(__('Pošlji', true));?>
</div>

```

Primer 6.19: Postavitev obrazca za dodajanje nove lokacije

6.5.2.6 Html

Pomočnik Html nam olajša kreiranje določenih elementov XHTML.

Tabela 6.41: Najpomembnejše metode pomočnika "Html"

Metoda	Opis
image	Kreira element, ki določa sliko.
link	Kreira element, ki določa povezavo.
url	Kreira naslov URL.
css	Omogoča dodajanje reference za zunanjo datoteko s stili.

6.5.2.7 Javascript

Pomočnik Javascript nam olajša vključevanje in dodajanje kode JavaScript v dokument XHTML.

Tabela 6.42: Najpomembnejše metode pomočnika "Javascript"

Metoda	Opis
codeBlock	Omogoča eksplicitno dodajanje kode JavaScript.
link	Omogoča implicitno dodajanje kode JavaScript, ki se nahaja v zunanji datoteki.

6.5.3 Pogledi

Izdelovanje pogledov je relativno enostavna operacija. Potrebno je določiti elemente XHTML, s katerimi predstavimo in oblikujemo podatke. Podatki so shranjeni v spremenljivkah, do katerih lahko dostopamo. Pogled je sestavljen iz dveh različnih programskih oz. skriptnih jezikov. Za določanje postavitve uporabljamo elemente jezika XHTML, za izpis vrednosti spremenljivk pa skriptni jezik PHP. Skupek obeh jezikov predstavlja močno orodje za generiranje pogledov.

Večina pogledov je skoraj enakih, zato se jih ne splača opisovati. Predstavili bomo samo tiste poglede, ki dejansko izstopajo. Ti so:

- dodajanje in urejanje izbranega videa,
- pregled lokacije,
- pregled statistike.

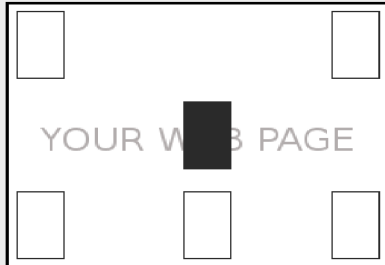
6.5.3.1 Dodajanje in urejanje izbranega videa

Izdelava obrazca za dodajanje oz. urejanje izbranega videa je bila relativno zahtevna. Potrebno je bilo omogočiti izbiro obstoječe lokacije ali dodajanje nove. Določanje pozicije je bilo potrebno implementirati na način, da bo uporabnik vizualno določil, kje želi predvajati video. Za določanje veljavnosti smo uporabili pomočnika “DatetimePicker”. Podobno smo za določanje barv uporabili pomočnika “ColorPicker”. Slika 6.9 prikazuje izdelan obrazec za urejanje izbranega videa.

Uredi izbran video: Umetnost



Lokacija
 Test1 - www.ebay.com
 Dodaj novo lokacijo

Pozicija


☒ Plavajoč ☐ Fiksiran

Podrobnosti

Jezik	Angleščina
Trajanje	14,83 s
Širina	300 px
Višina	320 px
Velikost	481,26 kB
Kategorije	Umetnost

Frekvenca
 3

Veljavnost od: 04/12/2010 01:00
 Veljavnost do: 05/20/2010 01:00

Širina: 300 px
 Višina: 320 px

Barva - aktivna: 
 Barva - neaktivna: 
 Barva - rob: 
 Barva - ozadje: 

URL za povezavo: http://

Pošlji

Slika 6.9: Obrazec za urejanje izbranega videa

6.5.3.2 Pregled lokacije

Posebnost pri izdelavi pregleda posmezne lokacije je bila implementacija funkcionalnosti, ki omogoča določanje vrstnega reda videov. Za ta namem smo uporabili knjižnico jQuery in dodatek TableDnD. Dodatek TableDnD omogoča spreminjanje vrstnega reda vrstic v tabeli z uporabo mehanizma “povleci-in-spusti” (angl. *drag-and-drop*). Slika 6.10 prikazuje uporabo dodatka za spreminjanje vrstnega reda.

Povezani videoi

ID	Video	Frekvenca	Veljavnost od / do	URL za povezavo	Akcije
↑ ↓ 1	Umetnost	3	12.04.2010 01:00:00 20.05.2010 01:00:00		Pregled Uredi Briši Statistika
↑ ↓ 2	Zabava	2	12.04.2010 00:25:00 12.05.2010 00:25:00	www.studioventura.si	Pregled Uredi Briši Statistika

Slika 6.10: Prikaz spreminjanje vrstnega reda videov za določeno lokacijo

6.5.3.3 Pregled statistike



Slika 6.11: Prikaz statistike (graf je bil generiran s knjižnico Raphael)

Posebnost pri izdelavi pogleda za pregled statistike je prikaz grafa. To smo izdelali z uporabo posebne knjižnice Raphael. Ta knjižnica omogoča generiranje prikaza grafa v sliki, ki je določena s formatom SVG. Podatke, ki jih želimo prikazati v grafu, izpišemo v tabeli XHTML, knjižnica pa nato poskrbi za generiranje slike. Graf, ki ga kreiramo s to knjižnico, je prikazan v sliki 6.11.

6.6 Izdelava distribucijskega kanala

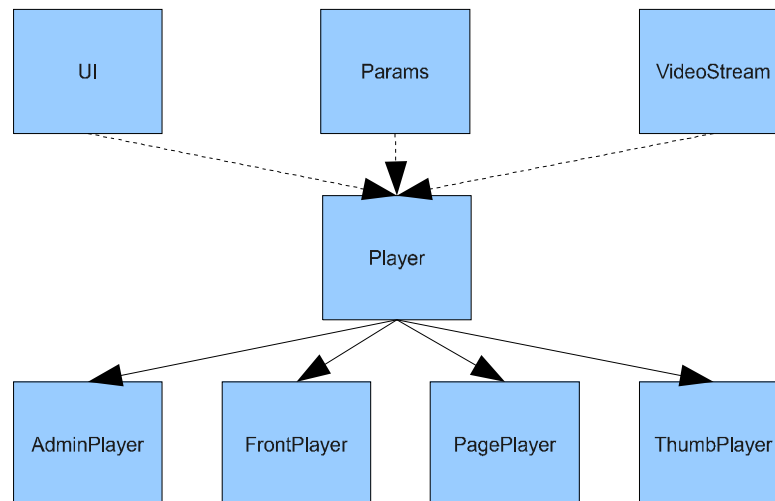
Distribucijski kanal je v osnovi razdeljen na dva segmenta. Prvi segment predstavlja predvajalnik videov. To je aplikacija v okolju Flash, ki služi predvajanju videov. Drugi segment pa vključuje skripto, ki omogoča vključevanje predvajalnika v gostujočo spletno stran. Ta skripta je izdelana z uporabo skriptnega jezika JavaScript. Skupaj pa omogočata distribucijo (vključevanje in predvajanje) videov.

6.6.1 Predvajalnik videov

Predvajalnik je zgrajen iz treh osnovnih razredov, in sicer:

- razreda za izris uporabniškega vmesnika (UI),
- razreda za branje parametrov (Params),
- razreda za predvajanje videa (VideoStream).

Razred, ki dejansko implementira predvajalnik (Player), uporablja vse zgoraj naštete razrede za svoje delovanje. Slika 6.12 prikazuje strukturo razredov in njihovo soodvisnost. Črtna puščica pomeni, da razred, kamor puščica kaže, uporablja razred, iz katerega puščica izhaja. Polna puščica pomeni, da razred, kamor puščica kaže, deduje od razreda, od koder puščica izhaja.



Slika 6.12: Razredi in njihova soodvisnost

Iz slike 6.12 je razvidno, da smo dejansko izdelali štiri predvajalnike. Ti so:

- AdminPlayer (predvajanje videov v administrativnem delu aplikacije),

- FrontPlayer (predvajanje videov v uporabniškem delu aplikacije),
- PagePlayer (predvajanje videov v gostujočih straneh),
- ThumbPlayer (omogoča kreiranje sličic).

AdminPlayer in FrontPlayer sta skoraj enaka. Uporabljata samo en gumb za nadzor predvajanja. Ta gumb omogoča prekinitev in nadaljevanje predvajanja. Razlikujeta se samo glede na akcijo ob koncu predvajanja. AdminPlayer se ob koncu predvajanja ugasne, FrontPlayer pa omogoča ponovno predvajanje.

PagePlayer vsebuje nabor treh gumbov za kontrolo predvajanja. Obiskovalec lahko začasno prekine predvajanje, utiša zvok ali popolnoma ugasne predvajalnik. Ob koncu predvajanja se PagePlayer tudi ugasne. PagePlayer omogoča tudi beleženje statistike klikov na video.

ThumbPlayer smo izdelali za kreiranje sličice, ki predstavlja video. Urednik lahko med pregledovanjem videa pritisne na tipko "S" in tako ustvari novo sličico, ki bo predstavljala trenutno predvajani video. S posebnim ukazom zajamemo trenutno predvajano sliko, ki jo nato zakodiramo v formatu BASE64. Tako zakodirano sliko pošljemo aplikaciji, ki jo shrani na strežniku.

6.6.1.1 Params

Razred "Params" skrbi za branje parametrov, ki jih predvajalnik pridobi preko dokumenta XHTML.

Tabela 6.43: Metode razreda "Params"

Metoda	Parametri	Opis
Params	Object, Player	To je konstruktor. Iz parametra tipa Object izluščimo parametre.
videoClick	MouseEvent	Po kliku na video preusmeri uporabnika na določeno spletno stran. Ob tem zabeleži klik v aplikacijo.
readParams		Vrne vse prebrane parametre.
getParam	String	Vrne posamezni parameter.
hexStringToNumber	String	Pretvori vrednost barve v formatu RGB v heksadecimalno vrednost.

6.6.1.2 UI

Razred “UI” izriše uporabniški vmesnik in se odziva na uporabniško interakcijo s predvajalnikom.

Tabela 6.44: Metode razreda “UI”

Metoda	Parametri	Opis
UI	Array, Player	Parameter tipa Array vsebuje barve, ki jih uporabimo za izris uporabniškega vmesnika.
play		Prikaže gumb za trenutno prekinitev predvajanja.
showOnlyPlayButton		Prikaže samo gumb za predvajanje. To potrebujemo v predvajalniku AdminPlayer in FrontPlayer.
noClose		Nastavi vrednost spremenljivke, ki ob koncu predvajanja prepreči zapiranje predvajalnika.
drawPlayButton		Izriše gumb za predvajanje.
drawSoundButton		Izriše gumb za utišanje zvoka.
drawSoundMark		Izriše gumb, ki omogoča ponovni vklop zvoka.
drawCloseButton		Izriše gumb za zapiranje predvajalnika.
drawLoading		Izriše animacijo, ki se pokaže v primeru, ko hitrost nalaganja datoteke iz strežnika ne sledi hitrosti predvajanja.
closeClick	MouseEvent	Odzove se na klik na gumb za zapiranje predvajalnika.
playClick	MouseEvent	Odzove se na klik na gumb za prekinitev ali nadaljevanje predvajanja.
pause		Prikaže gumb za nadaljevanje predvajanja.
soundClick	MouseEvent	Odzove se na klik na gumb za utišanje oz. ponovni vklop zvoka.
videoMouseMove	MouseEvent	Odzove se na premik miškega kazalca. Gumb za kontrolo predvajanja skrijemo, če uporabnik ne interaktira s predvajalnikom. Ko se pokliče ta metoda, gumb ponovno prikažemo.
hideControls	TimeEvent	Ko se pokliče ta metoda, skrijemo gumbe. Metoda se pokliče po poteku časovnika. To pomeni, da uporabnik določen čas ni interaktiral s predvajalnikom.
showControls		Ponovno prikaže gumbe.
showLoading		Prikaže animacijo med tem, ko čakamo na prenos datoteke.
moveLoading	TimeEvent	Izrisuje premikajoči se del animacije med čakanjem na

		prenos datoteke.
hideLoading		Skrije animacijo.
showPlayMark		Prikaže gumb za nadaljevanje predvajanja.
hidePlayMark		Skrije gumb za nadaljevanje predvajanja.

6.6.1.3 VideoStream

Razred “VideoStream” skrbi za nalaganje video datotek in omogoča predvajanje videa.

Tabela 6.45: Metode razreda “VideoStream”

Metoda	Parametri	Opis
VideoStream	String, Player	To je konstruktor. Parameter tipa String vsebuje naslov URL, kjer se nahaja video datoteka.
initVideo		Metoda inicializira povezavo s strežnikom, kjer se nahaja datoteka in zahteva prenos video datoteke.
play		Prične predvajanje videa.
pause		Trenutno prekine predvajanje videa.
resume		Nadaljuje predvajanje trenutno prekinjenega videa.
stop		Prekine predvajanje.
mute		Utiša zvok.
unMute		Ponovno vklopi zvok.
noClose		Ob koncu predvajanja prepreči zapiranje predvajalnika.
netStatusHandler	NetStatusEvent	Odziva se na dogodke med prenosom in predvajanjem videa.

Actionscript 3 vsebuje knjižnice, ki omogočajo relativno enostavno implementacijo predvajanja videa. Primer 6.20 prikazuje metodo “initVideo”, ki vzpostavi povezavo s strežnikom, zahteva prenos datoteke in postavi video v predvajalnik.

```
public function initVideo():void {
    this.ncConnection = new NetConnection();
    this.ncConnection.addEventListener(
        NetStatusEvent.NET_STATUS,
        netStatusHandler);
    this.ncConnection.connect(null);

    this.nsStream = new NetStream(this.ncConnection);
    this.nsStream.addEventListener(
        AsyncErrorEvent.ASYNC_ERROR,
```

```

        asyncErrorHandler);
    this.nsStream.addListener(
        NetStatusEvent.NET_STATUS,
        netStatusHandler);
    this.nsStream.client = this;
    this.nsStream.bufferTime = BUFFER_TIME;

    this.vidDisplay = new Video(this.player.stage.stageWidth,

    this.player.stage.stageHeight);
    this.vidDisplay.attachNetStream(this.nsStream);
    this.vidDisplay.smoothing = SMOOTHING;
    this.player.addChild(this.vidDisplay);
}

```

Primer 6.20: Metoda “initVideo”

Med prenosom in predvajanjem videa se lahko zgodi množica dogodkov. Vsak dogodek določa spremembo stanja prenosa ali predvajanja. Metoda “netStatusHandler”, ki je prikazana v primeru 6.21, se odziva na te dogodke. Naš predvajalnik se bo odzival na sledeče dogodke:

- “NetStream.Play.StreamNotFound” (zahtevana datoteka na strežniku ne obstaja),
- “NetStream.Play.Start” (predvajanje videa se je pričelo),
- “NetStream.Play.Stop” (video se je predvajal v celoti),
- “NetStream.Buffer.Flush” (datoteka se je prenesla v celoti),
- “NetStream.Buffer.Empty” (prenos datoteke je prepočasen, kajti predpomnilnik se je izpraznil),
- “NetStream.Buffer.Full” (predpomnilnik je poln in predvajanje se bo pričelo) [13].

```

private function netStatusHandler(event:NetStatusEvent):void {
    switch (event.info.code) {
        case "NetStream.Play.StreamNotFound":
            this.player.close();
            break;
        case "NetStream.Play.Start":
            this.resumeFromStart = true;
            break;
        case "NetStream.Play.Stop":
            if (this.noClosure) {

```

```

        this.stop(BUFFER_TIME);
        this.resumeFromStart = true;
        this.player.ui.pause();
        this.player.ui.showControls();
    } else {
        this.player.ui.pause();
        this.player.ui.visible = false;
        this.stop();
        this.player.close();
    }
    break;
case "NetStream.Buffer.Flush":
    this.isFlushed = true;
    break;
case "NetStream.Buffer.Empty":
    if (!this.isFlushed)
        this.player.ui.showLoading();
    break;
case "NetStream.Buffer.Full":
    this.player.ui.hideLoading();
    break;
}
}

```

Primer 6.21: Metoda “netStatusHandler”

6.6.1.4 Player

Razred “Player” uporablja razrede Params, UI in VideoStreama za svoje delovanje. Predstavlja vezni člen med posameznimi razredi.

Tabela 6.46: Metode razreda “Player”

Metoda	Parametri	Opis
Player		To je konstruktor. Inicializira objekte razredov Params, UI in VideoStream.
play		Zažene predvajanje videa.
close		Zapre predvajalnik.

Metoda “close” omogoča komunikacijo predvajalnika s spletno stranjo. Ko želimo ugasniti oz. zapreti predvajalnik, lahko to izvedemo le s pomočjo kode JavaScript. Obstaja posebna knjižnica v jeziku Actionscript 3, ki omogoča komunikacijo aplikacije Flash s spletnim mestom. Komunikacija poteka z uporabo kode JavaScript. Primer 6.22 prikazuje klic funkcije JavaScript, ki se nahaja v spletni strani.

```
ExternalInterface.call("shutDown()");
```

Primer 6.22: Klic funkcije JavaScript iz aplikacije Flash

6.6.2 Vključitev predvajalnika v spletno stran

Vključitev predvajalnika v gostujočo spletno stran izvedemo v treh fazah. V prvi fazi preverimo ustreznost brskalnika, v drugi fazi ustreznost predvajalnika Flash in v tretji fazi izvedemo vključitev. Ustreznost brskalnika je potrebno preveriti zaradi tega, ker vključujemo predvajalnik v spletno stran, ki ni naša, zato ne smemo dovoliti, da bi vključitev pokvarila izgled gostujoče spletne strani. Do tega lahko pride zaradi mnogih različnih brskalnikov, ki obstajajo na trgu. Določeni brskalniki niso implementirali zahtevanih standardov v celoti ali pa so del standarda implementirali površno oz. drugače, kot je navedeno v standardu. Določene verzije brskalnikov, ki so v uporabi, ne podpirajo vseh funkcionalnosti, ki jih naša aplikacija zahteva. Preveriti je potrebno tudi ustreznost verzije predvajalnika Flash. Določene verzije, predvsem starejše, ne podpirajo funkcionalnosti, ki jih potrebujemo za uspešno predvajanje videa. Vse tri faze izvedemo s pomočjo skriptnega jezika JavaScript.

6.6.2.1 Preverjanje ustreznosti brskalnika

Najprej je potrebno določiti tipe in verzije brskalnikov, v katerih bo naša aplikacija delovala brez problemov. Delovanje aplikacije je potrebno omogočiti v naslednjih brskalnikih:

- Firefox 3+,
- Microsoft Internet Explorer 7+,
- Chrome 4+,
- Safari 3+,
- Opera 8+.

Skriptni jezik JavaScript omogoča pridobivanje informacij o proizvajalcu in verziji brskalnika z branjem posebne spremenljivke. V primeru brskalnika Firefox, Internet Explorer, Chrome in Opera informacijo o proizvajalcu in verziji pridobimo z branjem spremenljivke "navigator.userAgent". V brskalniku Safari to informacijo dobimo v spremenljivki "navigator.vendor". Ko pridobimo te informacije za trenutni brskalnik, je potrebno samo še preveriti skladnost z informacijami v skripti.

6.6.2.2 Preverjanje ustreznosti predvajalnika Flash

Tudi v tem primeru si pomagamo s skriptnim jezikom JavaScript. Predvajalnik Flash je lahko v brskalniku vključen na dva načina. Internet Explorer uporablja predvajalnik Flash kot ActiveX kontrolo, medtem ko vsi ostali, zgoraj naštetih brskalniki, uporabljajo predvajalnik Flash kot dodatek. Zato je potrebno uporabiti dve različni metodi detekcije. Za ta namen je podjetje Adobe pripravilo prostodostopno skripto, ki omogoča detekcijo predvajalnika Flash. S pomočjo te skripte smo razvili lastno skripto, ki preveri, ali je predvajalnik Flash ustrezen. Podobno kot pri detekciji verzije brskalnika tudi tu uporabimo posebno spremenljivko "navigator.plugins", kjer brskalnik shrani imena in verzije dodatkov, ki jih uporablja. Vendar ta način je uporaben samo v brskalnikih, ki predvajalnik Flash implementirajo kot dodatek. V primeru brskalnika Internet Explorer uporabimo drugačno metodo. Potrebno je kreirati nov objekt tipa ActiveXObject, ki vsebuje potrebne informacije o verziji.

6.6.2.3 Vključitev

Ko smo preverili ustreznost brskalnika in predvajalnika Flash, lahko izvedemo vključitev. Kakor v prejšnjih dveh fazah je potrebno tudi v tej fazi upoštevati različne implementacije standarda XHTML. Obstajata dva načina dodajanja oz. vključevanja dodatkov v dokument XHTML.

Prvi način uporablja element OBJECT, ki omogoča vključevanje dodatkov za brskalnik Internet Explorer na platformi Windows. Primer 6.23 prikazuje element OBJECT in njegove parametre. Atribut CLASSID določa kontrolo ActiveX, katero naj brskalnik uporabi za prikaz vsebine. V tem primeru je to predvajalnik Flash. CODEBASE določa lokacijo, kjer se nahaja kontrola ActiveX. Brskalnik lahko s te lokacije prenese predvajalnik Flash in ga inštalira na klientovo platformo. Atributa WIDTH in HEIGHT določata širino in višino vključene vsebine, torej predvajalnika Flash. Parameter "allowScriptAccess" ščiti dokument XHTML pred potencialno zlonamerno aplikacijo Flash, tako da nadzoruje zmožnost klika kode JavaScript iz aplikacije. Parameter lahko določa tri možne vrednosti: "always", "never" in "sameDomain". Vrednost "always" ali "never" brezpogojno omogoča ali onemogoča dostop do kode JavaScript. Vrednost "sameDomain" omogoči dostop do kode JavaScript samo, če je aplikacija Flash prenešena z iste lokacije kot dokument XHTML. Parameter "movie" določa lokacijo aplikacije Flash. Parameter "quality" določa kakovost prikazane vsebine v predvajalniku. Parameter določa 6 različnih stopenj kvalitete. Za našo aplikacijo bomo uporabili predzadnjo stopnjo kvalitete, in sicer "high". Ta stopnja daje prednost kvaliteti prikaza pred hitrostjo predvajanja in za izrisovanje vedno uporablja metodo "anti-aliasing". Določitev vrednosti "transparent" parametra "wmode" dejansko omogoča predvajanje transparentnega videa. Parameter lahko vsebuje še dve vrednosti, in sicer "window", ki predvaja aplikacijo kot del dokumenta XHTML, ter vrednost "opaque", ki

skrije celotno spletno stran razen aplikacije Flash. Parameter "FlashVars" določa spremenljivke in njihove vrednosti, ki jih lahko uporabimo v aplikaciji. Imena spremenljivk in njihove vrednosti zapišemo v obliki para "name=value", kjer "name" določa ime spremenljivke in "value" določa vrednost spremenljivke. Če želimo vključiti več spremenljivk, potem je potrebno pare ločiti z znakom "&". Primer 6.24 določa spremenljivke in njihove vrednosti, ki jih pošljemo aplikaciji [14].

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
        codebase="http://fpdownload.macromedia.com/pub/shockwave/
                cabs/flash/swflash.cab#version=9,0,0,0"
        width="300"
        height="320">
    <param name="allowScriptAccess" value="always" />
    <param name="movie"
        value="http://<domena>/flash/PagePlayer.swf"/>
    <param name="quality" value="high" />
    <param name="wmode" value="transparent" />
    <param name="FlashVars" value="<flashvars>" />
</object>
```

Primer 6.23: Uporaba elementa OBJECT

```
videoURL=http://<domena>/videos/play/
                wsedrf43edfrtg56zhgt54edfr5tgbnj889ikjhg&
linkURL=http://www.google.com&
pageURL=http://<domena>&
selectedCode=wsedrf43edfrtg56zhgt54edfr5tgbnj889ikjhg&
activeColor=2147AB&
inactiveColor=333333&
borderColor=000000&
backgroundColor=AAAAAA
```

Primer 6.24: Pari spremenljivk in njihovih vrednosti (parameter FlashVars)

Drugi način vključitve dodatka v spletno stran omogoča uporaba elementa EMBED, ki je uporabljen v brskalnih Firefox, Chrome, Safari in Opera. Primer 6.25 prikazuje uporabo elementa EMBED, ki omogoča vključitev aplikacije v spletno stran. Atribut "src" določa lokacijo aplikacije, ki jo želimo uporabiti. Atribut "pluginspage" določa lokacijo predvajalnika Flash. Brskalnik lahko z lokacije prenese predvajalnik Flash in ga inštalira na klientovo platformo. Atribut "type" pove brskalniku, kateri dodatek naj uporabi za prikaz te vsebine. Vrednost "application/x-shockwave-flash" predstavlja vrednost tipa MIME, ki opredeljuje vsebino Flash. Atributa "width" in "height" sta ekvivalentna atributoma elementa OBJECT in določata širino in višino vsebine. Podobno so vsi ostali atributi ekvivalentni parametrom elementa OBJECT [14].

```
<embed src="http://<domena>/flash/PagePlayer.swf"
  pluginspage="http://www.macromedia.com/go/getflashplayer"
  type="application/x-shockwave-flash"
  width="300"
  height="320"
  quality="high"
  wmode="transparent"
  FlashVars="'<flashvars>"
  allowScriptAccess="always" />
```

Primer 6.25: Uporaba elementa EMBED

Obe zgoraj opisani metodi lahko združimo in tako omogočimo vključitev vsebine v večini brskalnikov. Takšen pristop svetuje tudi podjetje Adobe. Primer 6.26 prikazuje združitev obeh metod.

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
  codebase="http://fpdownload.macromedia.com/pub/shockwave/
    cabs/flash/swflash.cab#version=9,0,0,0"
  width="300"
  height="320">
  <param name="allowScriptAccess" value="always" />
  <param name="movie"
    value="http://<domena>/flash/PagePlayer.swf" />
  <param name="quality" value="high" />
  <param name="wmode" value="transparent" />
  <param name="FlashVars" value="<flashvars>" />
  <embed src="http://<domena>/flash/PagePlayer.swf"
    pluginspage="http://www.macromedia.com/go/getflashplayer"
    type="application/x-shockwave-flash"
    width="300"
    height="320"
    quality="high"
    wmode="transparent"
    FlashVars="'<flashvars>"
    allowScriptAccess="always" />
</object>
```

Primer 6.26: Združena elementa OBJECT in EMBED

Posebno pozornost je potrebno nameniti pozicioniranju predvajalnika v gostujoči spletni strani. Za ta namen določimo 5 stilov CSS, in sicer “left”, “top”, “right”, “bottom” ter “position”. Stile posameznega elementa v dokumentu XHTML lahko nastavljamo z uporabo skriptnega jezika JavaScript. Primer 6.27 prikazuje določanje stilov za pozicioniranje predvajalnika v levi zgornji kot spletne strani. Prva vrstica v primeru določa vrsto pozicioniranja. V tem primeru smo predvajalnik pozicionirali plavajoče.

```

placeholder.style.position = 'fixed';
placeholder.style.left = 0;
placeholder.style.top = 0;

```

Primer 6.27: Pozicioniranje predvajalnika v gostujoči spletni strani

Slika 6.13 prikazuje končni rezultat vključitve predvajalnika v gostujočo spletno stran. Predvajalnik je vključen na sredini spodnje stranice spletne strani. Določeno ima fiksno pozicioniranje.



Slika 6.13: Vključen predvajalnik v spletno stran www.progmbh.com

6.7 Produciranje videov

Vsebinsko naše aplikacije predstavljajo videi, ki so posneti in obdelani na poseben način. Proces produkcije videov je izredno pomemben, kajti brez izdelanega procesa ne bi mogli kreirati videov in aplikacija bi bila nesmiselna. Proces produkcije je razdeljen na tri faze:

1. snemanje posnetkov v posebnem studiu,

2. določanje transparentne plasti v videu,
3. kodiranje videa v zahtevanem formatu.

6.7.1 Snemanje

Transparenta plast je poglavitna lastnost videov, ki jih produciramo. Da lahko izdelamo transparentne videe, je potrebno proces snemanja temu prilagoditi. Najpomembnejši element procesa je uporaba posebnega studia, kjer videe snemamo pred ozadjem konstantne barve. Za ta namen smo uporabili lastni studio, kjer uporabljamo zeleno barvo ozadja. Fotografiji v sliki 6.14 prikazujeta naš studio. Zelena barva je najprimernejša za produciranje digitalnih videov s transparentno plastjo, kajti zelena barva uporablja največjo stopnjo vzorčenja v digitalnem videu. Pozornost je tudi potrebno nameniti sceni in igralcem, ki jih uporabljamo pri snemanju. Tako scena kakor tudi igralci ne smejo vsebovati zelene barve. Potrebno se je izogibati uporabi bele barve in svetlih predmetov, ker lahko reflektirajo barvo ozadja.



Slika 6.14: Fotografiji studia



Izrednega pomena je osvetljenost igralcev, scene in ozadja. Ozadje mora biti osvetljeno enakomerno, kar dosežemo z uporabo treh luči. Da ne bi bilo nepotrebnih senc na ozadju, je potrebno luči postaviti za igralcem. Po eno luč postavimo na vsako stran igralca. Tretjo luč pa postavimo nad igralca in jo usmerimo navzdol. Posebej je potrebno osvetliti igralca in sceno. Za to potrebujemo dodatne luči, ki jih usmerimo v igralca in sceno. Pri tem je potrebno biti pazljiv, da tako osvetljeni igralec ne meče sence na ozadje.

6.7.2 Transparentnost videa

Ko je video posnet, sledi naslednja faza. Iz posnetega videa je potrebno odstraniti zeleno barvo ozadja (angl. *keying*). Pri tem si pomagamo s posebnim programskim paketom Adobe After

Effects, ki služi urejanju video posnetkov. Program vsebuje poseben dodatek, imenovan Linear Color Key, ki nam omogoča odstranjevanje določenih barv. Z nastavitvijo posebnih parametrov lahko odstranimo zeleno barvo ozadja. Rezultat je video, ki ima transparentno plast.

6.7.3 Kodiranje videa

Zadnji korak kreiranja video posnetkov predstavlja kodiranje videa. Video je potrebno zakodirati v formatu FLV, ki bo omogočal predvajanje videa s transparentnim ozadjem. Za predstavitev videa v formatu FLV bomo uporabili kodek VP6 z alfa kanalom. Zvok pa bo predstavljen s kodekom MP3. Za ta namen uporabljamo programski paket On2 Flix Pro. Pri tem je potrebno nastaviti vrednosti določenih parametrov. Najprej je potrebno določiti širino in višino videa. Za potrebe naše aplikacije smo določili standardno velikost, ki znaša 300 pikslov v širino in 320 pikslov v višino. Potrebno je določiti t. i. "data rate" oz. "bitrate", ki določa potrebno količino prenešenih podatkov za predvajanje videa. To vrednost smo nastavili na 250 kb/s. Za zvok je potrebno nastaviti isti parameter. Vrednost tega smo nastavili na 48 kb/s. Zvok je vzorčen z 44000 vzorci na sekundo. Video se predvaja s 30 sličicami na sekundo (angl. *framerate*). Ker se bo video preko omrežja prenašal z metodo progresivnega nalaganja, smo uporabili metodo "variable bit rate". Z uporabo te metode se potrebna količina prenešenih podatkov za predvajanje videa spreminja glede na kompleksnost posamezne sličice (angl. *frame*). Video je sestavljen iz ključnih sličic (angl. *keyframes*) in t. i. delta sličic (angl. *delta frames*). Ključne sličice vsebujejo popolno informacijo o sličici in so enakomerno razporejene skozi celoten video. Med ključnimi sličicami se nahajajo delta sličice, ki vsebujejo informacije o razlikah med sosednjimi sličicami. Razdalja med dvema ključnima sličicama določa kvaliteto kompresije. Če je razdalja večja, je velikost zakodirane datoteke manjša, hkrati pa se tudi poslabša kvaliteta predvajanega videa. To razdaljo smo nastavili na vrednost 60 sličic. Kodirnik bo ključno sličico vstavil vsakih 60 sličic. Video zakodiramo z uporabo metode dvojnega prehoda. Ob prvem prehodu kodirnik samo analizira video. Pri tem si zapomni kompleksnost posamezne sličice. Ob drugem prehodu uporabi te informacije in poskuša enakomerno razporediti "bitrate". Rezultat te metode je enakomernejša razporeditev "bitrate"-a in posledično boljša kvaliteta predvajanega videa.

6.8 Težave pri razvoju

Med razvojem aplikacije nismo naleteli na veliko težav. Kar definitivno pomeni, da je bilo načrtovanje aplikacije izvedeno relativno uspešno. Kljub temu smo morali v treh primerih zavihati rokave.

Prva težava se je pojavila pri ustvarjanju sličice. Z uporabo skriptnega jezika Actionscript 3 smo lahko pridobili trenutno predvajano sliko. Problem je predstavljalo shranjevanje slike na

strežniku. Aplikacija Flash teče na strani klienta, zato je potrebno za shranjevanje sličice uporabiti skripto, ki teče na strežniku. Sličico je potrebno prenesti na strežnik, kjer jo bo skripta shranila v datotečni sistem strežnika. Actionsript 3 je pridobljeno sliko shranil v spremenljivko tipa ByteArray, ki predstavlja polje bajtov. To polje je potrebno prenesti na strežnik. Težavo smo rešili tako, da smo polje ByteArray zakodirali s shemo BASE64. Tako zakodirano sliko smo nato lahko poslali na strežnik z uporabo metode POST protokola HTTP. Na strežniku smo sliko dekodirali in jo shranili v datotečni sistem strežnika.

Naslednja težava se je pojavila pri kreiranju gumba, ki omogoča ponovni zagon predvajalnika. Gumb je prikazan v sliki 6.6. Knjižnica GD, ki smo jo uporabili za generiranje slike, ni vsebovala funkcij, ki bi omogočale uporabo metode “anti-aliasing”. Če ne bi uporabili te metode, bi gumb imel nazobčene robove. Težavo smo rešili z uporabo metode nadvzorčenje (angl. *supersampling*). Generiramo gumb, ki je desetkrat večji kot je potrebno. Nato ta gumb zmanjšamo na želeno velikost. Med zmanjšanjem se izvede ponovno vzorčenje (angl. *resampling*), ki uporablja gladko interpolacijo pikslov. Rezultat je gumb brez nazobčenih robov.

Za prikaz grafa statistike za posamezni izbrani video smo uporabili knjižnico Raphael, ki generira graf iz tabele v dokumentu XHTML. Težava se je pojavila, ko smo želeli izdelati možnost, ki bi omogočala shranjevanje prikazanega grafa. Graf je izdelan v formatu SVG, ki je dejansko datoteka XML. Če želimo omogočiti prenos grafa, je potrebno nastaviti določene parametre v glavi HTTP odziva. To lahko storimo z uporabo skripte PHP na strežniku. Graf se generira na strani klienta, kar predstavlja problem. Graf je potrebno poslati na strežnik, kjer skripta nastavi zahtevane parametre in omogoči prenos. S pomočjo knjižnice jQuery smo izdelali funkcijo JavaScript, ki omogoča prenos grafa. Primer 6.28 prikazuje to funkcijo. Funkcija ustvari trenutni obrazec v dokumentu XHTML, s pomočjo katerega pošlje graf na strežnik.

```
jQuery.download = function(url, data, method){
if( url && data ){
    inputs = '<textarea name="data">' + data + '</textarea>';
    jQuery('<form action="' + url +
        '" method="' + (method || 'post') + '">' +
        inputs + '</form>')
        .appendTo('body').submit().remove();
};
};
```

Primer 6.28: Funkcija za prenos datoteke

7 Sklepne ugotovitve

Zastavljeni cilj, izdelati aplikacijo za distribucijo transparentnih videov v spletne strani, je po našem mnenju uspešno dosežen. Aplikacija omogoča vključevanje vnaprej posnetih videov v gostujoče spletne strani. Z uporabo aplikacije lahko uporabniki povečajo prepoznavnost vsebine in čas, ki ga obiskovalci preživijo na njihovem spletnem mestu.

Izbira ogrodja CakePHP za izdelavo aplikacije se je izkazala za zelo uspešno. Ogrodje vključuje množico uporabnih orodij, ki skrajšajo čas razvoja in povečajo kvaliteto izdelane aplikacije. Odlično strukturirana programska koda, ki je posledica uporabe vzorca MVC, omogoča relativno enostavni nadaljnji razvoj. Spodaj so predstavljeni trije predlogi nadgradnje aplikacije in predvajalnika video vsebin.

Pri določanju pozicije izbranega videa bi lahko uporabili sliko, ki bi predstavljala spletno stran, na kateri se bo video predvajal. To sliko bi prikazali v ozadju polja, kjer se določa pozicija. Za ta namen bi lahko uporabili eno izmed številnih spletnih storitev, ki to omogočajo.

Predvajanje videov bi lahko nadgradili z možnostjo prikaza podnapisov ali izbire jezika videa. Za implementacijo podnapisov bi morali nadgraditi predvajalnik in izdelati prevode za videe. Če bi želeli imeti možnost izbire jezika v videu, bi morali ponovno posneti vsa besedila v različnih jezikih. Nadgraditi bi bilo potrebno predvajalnik z možnostjo zamenjave oz. izbire avdio posnetka.

Najbolj daljnosežna nadgradnja bi bila možnost, ki bi uporabnikom aplikacije omogočala dodajanje lastnih videov. Izdelati bi bilo potrebno aplikacijo Flash, ki bi omogočala snemanje video posnetkov preko spletne strani. Te videe bi lahko obdelali na strežniku, kjer bi določili transparentno plast. Za ta namem bi lahko uporabili odprtokodni programski paket FFmpeg.

Literatura

- [1] R. W. Sebesta, *Programming the World Wide Web 2009: Fifth Edition*, New Jersey: Pearson Addison-Wesley, 2010, pogl. 1, 2, 3, 4, 7.
- [2] K. Coar, R. Bowen, *Apache Cookbook*, Sebastopol: O'Reilly, 2008, pogl. 3, 4.
- [3] R. Nixon, *Learning PHP, MySQL & JavaScript*, Sebastopol: O'Reilly, 2009, pogl. 8, 16.
- [4] D. Golding, *Beginning CakePHP: From Novice to Professional*, Berkeley: Apress, 2008, pogl. 1.
- [5] K. Chain, J. Omokore, R. K. Miller, *Practical CakePHP Projects*, Berkeley: Apress, 2009, pogl. 1.
- [6] E. Rosebrock, E. Filson, *Setting Up LAMP: Getting Linux, Apache, MySQL, and PHP Working Together*, Alameda: Sybex, 2004, pogl. 1.
- [7] B. Bibeault, Y. Katz, *jQuery in Action*, Greenwich: Manning, 2008, pogl. 1.
- [8] (2010) *Adobe - Flash Player Statistics*. Dostopno na:
http://www.adobe.com/products/player_census/flashplayer/
- [9] (2010) *Video File Format Specification*. Dostopno na:
http://www.adobe.com/devnet/flv/pdf/video_file_format_spec_v10.pdf
- [10] F. Solina, *Projektno vodenje razvoja programske opreme*, Ljubljana: Založba FE in FRI, 1997, pogl. 3, 5.
- [11] B. Collins-Sussman, B. W. Fitzpatrick, C. M. Pilato, *Version Control with Subversion*, Sebastopol: O'Reilly, 2004, pogl. 2, 3.
- [12] (2010) *Understanding how ACL works*. Dostopno na:
<http://book.cakephp.org/view/465/Understanding-How-ACL-Works>
- [13] (2010) *NetStatusEvent - ActionScript 3.0 Language and Components Reference*. Dostopno na:
<http://www.adobe.com/livedocs/flash/9.0/ActionScriptLangRefV3/flash/events/NetStatusEvent.html#info>
- [14] (2010) *Flash OBJECT and EMBED tag attributes*. Dostopno na:
http://kb2.adobe.com/cps/127/tn_12701.html