UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Tomaž Močnik

IZDELAVA SPLETNE APLIKACIJE ZA KOMUNIKACIJO TER SODELOVANJE MED UPORABNIKI

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Mentor: prof. dr. Saša Divjak

Ljubljana, 2011

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: 00103/2011
Datum: 04.04.2011

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

Kandidat: **TOMAŽ MOČNIK**

Naslov: **IZDELAVA SPLETNE APLIKACIJE ZA KOMUNIKACIJO TER SODELOVANJE MED UPORABNIKI**

DEVELOPMENT OF A WEB APPLICATION FOR COMMUNICATION AND COLLABORATION BETWEEN USERS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V sklopu diplomske naloge razvijte spletno aplikacijo, ki naj uporabnikom omogoča medsebojno sodelovanje ter preko spletnih vsebin, izmenjavo idej, obvestil ter podobnih informacij.
Aplikacija naj registriranim uporabnikom ob obisku poljubne spletne strani omogoča to spletno vsebino poljubno opremiti z raznimi funkcionalnostmi, kot so poudarjanje besedila, dodajanje opomb, vstavljanje objektov s pomočjo
vrivanja programske kode, brisanje objektov na spletni strani, uokvirjanje objektov itd.
Registrirani uporabniki lahko tako, po meri opremljeno, spletno stran preko elektronske pošte
posredujejo drugim, lahko pa jo le shranijo v svojem uporabniškem računu, kjer jim je kasneje na voljo za dodatno urejanje.

Mentor:

prof. dr. Saša Divjak

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:

prof. dr. Nikolaj Zimic

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Namesto te strani **vstavite** original izdane teme diplomskega dela s podpisom mentorja in dekana ter žigom fakultete, ki ga diplomant dvigne v študentskem referatu, preden odda izdelek v vezavo!

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Tomaž Močnik,

z vpisno številko 63030238,

sem avtor diplomskega dela z naslovom:

Izdelava spletne aplikacije za komunikacijo ter sodelovanje med uporabniki

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 4.7.2011 Podpis avtorja:

Zahvala

Zahvaljujem se mentorju prof. dr. Saši Divjaku za pomoč in vodenje pri opravljanju diplomske naloge. Zahvaljujem se tudi Mitju Kovačiču za vse nasvete in ideje, ki so mi pomagale pri izdelavi spletne aplikacije.

Iskreno se zahvaljujem staršem in bratu, ki so mi omogočili študij in me tudi moralno podpirali. Njihova pomoč in vzpodbuda sta pripomogli k uresničitvi zastavljenih ciljev.

Posebna zahvala pa gre tudi mojemu dekletu Gabrijeli za izkazano razumevanje ter brezpogojno podporo.

Ker čas kar prehitro mineva je to diplomsko delo tudi posvetilo žal že pokojnima očetu Tinetu ter staremu očetu Vinku.

Hvala za vse.

Kazalo

Seznam uporabljenih kratic in simbolov

AJAX - *Asynchronus JavaScript And XML (Asinhorni JavaScript in XML)*
ANSI - *American National Standards Institute (Ameriški nacionalni inštitut za standardizacijo)*
ASP - *Active Server Pages (Ogrodje za razvijanje dinamičnih spletnih strani, ki ga podpira Microsoft)*
CGI - *Common Gateway Interface (Standard za pisanje spletnih programov)*
CSS - *Cascading Style Sheets (Stilne predloge za izgled spletne strani)*
DOM - *Document Object Model (Objektni model sestave dokumenta spletne strani)*
HTML - *HyperText Markup Language (Hipertekstovni označevalni jezik)*
ISAPI - *Internet Server Application Programming Interface (Programski vmesnik za strežnik IIS)*
ISO - *International Standard Organization (Mednarodna organizacija za standardizacijo)*
ISS - *Internet Information Services (Microsoftov spletni strežnik)*
JavaScript - *Skriptni programski jezik za razvoj interaktivnih spletnih strani*
JSON - *JavaScript Object Notation (Notacija JavaScript objektov)*
JSP - *Java Server Pages (Tehnologija za hiter razvoj dinamičnih spletnih vsebin)*
MVC - *Model-View-Controller (Model-Pogled-Krmilnik)*
MySQL - *My Structured Query Language (Odprtokodni sistem za upravljanje zbirk podatkov)*
NSAPI - *Netscape Server Application Programming Interface (Programski vmesnik za Netscape-ov strežnik)*
PHP - *PHP Hypertext Preprocessor (Programski jezik za razvoj dinamičnih spletnih strani)*
SGML - *Standard Generalized Markup Language (Meta-jezik, ki služi za opis besedilnih dokumentov)*

SQL - *Structured Query Language (Standardiziran jezik, ki se ga uporablja za dostop do zbirk podatkov)*
URI - *Uniform Resource Identifier (Enotni označevalnik vira)*
URL - *Uniform Resource Locator (Enolični krajevnik vira)*
VBScript - *Visual Basic Script (Microsoftov skriptni programski jezik)*
W3C - *World Wide Web Consortium (Konzorcij svetovnega spleta)*
WWW - *World Wide Web (Svetovni splet)*
XHR - *XMLHttpRequest (Vmesnik, ki se uporablja za prenos podatkov med spletnim strežnikom in brskalnikom)*
XHTML - *HyperText Markup Language (Označevalni jezik, podoben HTML-ju, vendar usklajen s sintakso XML)*
XML - *Extensible Markup Language (Razširljiv označevalni jezik)*
XSL - *Extensible Stylesheet Language Family (skupek priporočil za transformacijo in prikaz dokumentov XML)*
XSLT - *eXtensible Stylesheet Language Transformations (jezik za pretvorbo in urejanje dokumentov XML)*

Povzetek

Svetovni splet je bil kot storitev širši javnosti na voljo v začetku devetdesetih let prejšnjega stoletja. Prvo generacijo spleta smo uporabljali zgolj kot vir informacij, v poznih devetdesetih pa so se začele pojavljati spletne strani, ki so omogočale več. Splet, kakršnega poznamo danes, imenujemo splet druge generacije. Ena izmed glavnih lastnosti spleta druge generacije je zmožnost komunikacije ter sodelovanja med uporabniki. V diplomski nalogi je predstavljen razvoj sodobne spletne aplikacije, ki izstopa prav v tej lastnosti - komunikaciji ter sodelovanju med uporabniki.

V nalogi smo si torej prizadevali realizirati rešitev problema, ki nastane pri komunikaciji ter sodelovanju med uporabniki svetovnega spleta. Morda se vam je že zgodilo, da ste želeli nekomu na hiter ter enostaven način posredovati povezavo do neke spletne strani, hkrati pa na tej strani poudariti določeno vsebino ter tej vsebini dodati še nekaj svojih misli, pa niste našli ustreznega načina za izmenjavo teh informacij. Zato je nastala ideja za izdelavo spletne aplikacije, opisane v tej nalogi, ki nam le-te želje lahko izpolni. Je pa to le eden izmed primerov uporabe, ki jih naša aplikacija omogoča.

Diplomska naloga je v grobem razdeljena na dva dela. V prvem delu je na nekoliko bolj teoretičen način predstavljenih večji del orodij, pristopov in tehnologij, ki so bili uporabljeni pri razvoju naše spletne aplikacije. Drugi del naloge pa je namenjen prikazu samega razvoja aplikacije. Torej, prikazan je problem, katerega smo želeli v nalogi rešiti ter sodobna spletna aplikacija kot rešitev problema.

Ključne besede:

spletne aplikacije, spletne tehnologije, socialni splet, splet 2.0, PHP, Ajax, MySQL

Abstract

The World Wide Web has been known as service to wider community since the early 1990's. The very first generation of World Wide Web was used only as source of information but in the late 1990's new web pages started to appear, which offered us much more than before. The World Wide Web as we know today is called Web 2.0. One of the most important characteristic of Web 2.0 is the ability to communicate and collaborate with other users. Development of web application that stands out with this characteristic is demonstrated in this thesis – the communication and collaboration between users.

In this thesis we tried to realize solution of the problem which occurs with communication and collaboration between users of the World Wide Web. It may have happened to you, that you tried to forward somebody a hyperlink on a web page in a fast and easy way, at the same time highlight certain content and add this content a few of your own ideas, but you didn't find the right way to exchange this information. For that reason an idea of making a web application that offers us all that and more, has been developed and described in this thesis.

This thesis is briefly divided into two parts. In the first part, there is theoretically represented majority of tools, approaches and technologies, which were used in developing our web application. Therefore a problem that we wanted to solve and a modern web application as a solution to this problem are discussed in this thesis.

Key words:

web applications, web technologies, social web, web 2.0, PHP, Ajax, MySQL

Poglavje 1

Uvod

Živimo v času, ko se namizne aplikacije počasi, vendar vztrajno selijo v splet oz. spletne brskalnike. Spletne aplikacije postajajo vse bolj napredne in uporabnikom nudijo vse več funkcionalnosti, hkrati pa je delo z njimi lažje. Za njihovo uporabo praktično ne potrebujemo drugega kot povezavo v internet ter spletni brskalnik, torej ni potrebe po nameščanju programske opreme na odjemalčevi strani. Tako je tudi vzdrževanje lažje, saj je tako potrebno vzdrževati le aplikacijo na strežniški strani in ne pri vsakem odjemalcu posebej. To je le nekaj razlogov, zakaj postajajo spletne aplikacije čedalje bolj priljubljene.

Danes si marsikdo ne predstavlja dneva brez dostopa do interneta. To pa niti ni tako presenetljivo, če upoštevamo, da vsak dan večje število uporabnikov interneta z uporabo spletnih aplikacij na spletnih strežnikih hrani pomembne podatke, kateri pa jim brez dostopa do interneta nikakor niso dostopni. Za primer lahko podamo uporabo spletnih servisov podjetja Google (elektronska pošta - Gmail, spletni albumi - Picasa, zemljevidi - Google Maps, dokumenti - Google Docs, idr.). Skoraj vsakdo izmed nas uporablja vsaj enega izmed naštetih spletnih servisov, kjer pa tudi hrani pomembne podatke.

Z naraščanjem uporabe interneta pa narašča tudi komuniciranje ter sodelovanje med uporabniki, kar pa je značilno za splet druge generacije. Uporabniki lahko medsebojno komunicirajo ter sodelujejo bodisi preko elektronske pošte, bodisi preko socialnih omrežij ali po neki tretji poti. Izbire načina komunikacije ter sodelovanja je veliko, ampak vseeno včasih naletimo na dilemo, kateri način izbrati. Morda se je tudi vam že zgodilo, da ste želeli nekomu na hiter ter enostaven način nekaj sporočiti po elektronski poti, pa za to niste našli ustrezne metode. No, meni in kolegu se je zgodilo ravno to. Namreč, hotela

sva si izmenjati spletno povezavo do neke spletne strani, hkrati pa na tej strani poudariti določeno vsebino ter tej vsebini dodati še nekaj svojih misli. Ker pa nisva našla ustrezne metode za izmenjavo najinih informacij, sva se odločila za rešitev problema. Tako je nastal tudi praktičen primer, ki bo uporabljen v tej diplomski nalogi.

Diplomska naloga bo v grobem razdeljena na dva dela. V prvem delu bomo na nekoliko bolj teoretičen način opisali večji del orodij, pristopov in tehnologij, ki so bili uporabljeni pri realizaciji rešitve. Tako bomo najprej nekaj besed namenili spletu druge generacije. Nato bomo predstavili spletne brskalnike. Na kratko bomo povzeli kaj spletne aplikacije sploh so. Nato bo na vrsto prišel programski jezik PHP ter z njim tesno povezano ogrodje Zend Framework, katerega smo pri realizaciji naše aplikacije dodobra tudi izkoristili. Ne moremo niti mimo pristopa Ajax in z njim tesno povezanih spletnih razvojnih tehnologij kot so (X)HTML, stilske predloge CSS, modela DOM, označevalnega jezika XML ter jezika XSLT. Zelo pomemben je tudi objekt XMLHttpRequest, katerega bomo opisali kot naslednjega. Nekaj besed bomo namenili tudi programskemu jeziku JavaScript in knjižnici jQuery, katera nam je zelo olajšala izgradnjo nekaterih funkcionalnosti aplikacije. Nenazadnje pa bomo namenili nekaj besed tudi aktivnim zaznamkom (Bookmarkletom), saj so eden ključnih elementov pri naši aplikaciji. Pri realizaciji naše aplikacije smo uporabili podatkovno bazo MySQL, katero bomo tudi na kratko predstavili na koncu prvega dela te naloge.

Drugi del naloge pa bo namenjen prikazu razvoja naše spletne aplikacije. Tako bomo najprej predstavili sam problem, katerega smo želeli v nalogi rešiti s pomočjo predhodno opisanih orodij, pristopov in tehnologij. Na diagramu primerov uporabe bomo prikazali ter v nadaljevanju tudi opisali funkcionalnosti, katere je morala naša aplikacija tudi omogočati. Za najbolj ključen primer uporabe bomo zgradili tudi diagram zaporedja ter ga opisali. Na koncu pa bomo predstavili našo aplikacijo kot rešitev problema in opisali njen namen, funkcionalnosti ter delovanje uporabniškega vmesnika.

V zadnjem poglavju naloge pa bodo sledile še sklepne ugotovitve, kjer bomo prikazali oceno o opravljenem delu ter povzeli težave na katere bomo morebiti naleteli. Navedli bomo tudi morebitne ideje, ki bodo nastale med delom, in bi lahko bile predmet nadaljnjih raziskav.

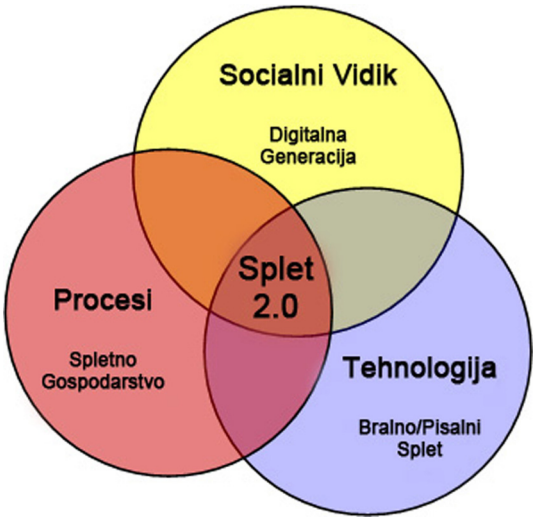
Poglavje 2

Splet 2.0

V začetku devetdesetih let prejšnjega stoletja se je pojavila storitev, ki je širši javnosti omogočala predstavitev statične enciklopedije. Poimenovali so jo svetovni splet (angl. World Wide Web). Prvo generacijo spleta (Web 1.0) smo uporabljali zgolj kot vir informacij. V poznih devetdesetih pa so se začele pojavljati spletne strani, ki so omogočale več. Statično enciklopedijo je zamenjala dinamična, katero je lahko širša javnost pregledovala ter jo hkrati popravljala in spreminjala. Leta 2004 je Tim O'Reilly prvi uporabil izraz Splet 2.0 (Web 2.0) in z njim označil novo generacijo spleta. [1]

Čeprav daje termin Splet 2.0 občutek, da uvaja novo različico svetovnega spleta, pa gre predvsem za izrabo že znanih orodij in tehnologij za gradnjo spletnih aplikacij na uporabnikom bolj domač način in ne za tehnične posodobitve. Splet 2.0 je platforma za množico spletnih storitev in skupnosti, katerih namen je izvajati in vzdrževati sodelovanje ter deljenje med uporabniki. Osredotoča se na vsebine, prispevane od posameznih uporabnikov.

Splet 2.0 lahko razdelimo na tri osnovna področja, katera sestavljajo socialni vidik, procesi in tehnologija (slika 2.1). Socialni vidik je osredotočen na uporabnikovo upravljanje tehnologije za medsebojno povezavo in povezavo s poslovnimi subjekti. Splet 2.0 podpira sodelujoč pristop, v katerem uporabniki ne nastopajo samo kot uporabniki, ampak tudi kot razvijalci aplikacij in vsebin, ki so združljive z ostalimi rešitvami, ki jih vsebuje Splet 2.0. Procesi so osredotočeni na poslovne procese in dodano vrednost. Tehnologija pa se osredotoča na arhitekturo in razvojni model ter predstavlja nov način izgradnje, nameščanja in obratovanja spletnih aplikacij. [2]



Slika 2.1: Osnovna področja Spleta 2.0.

Splet 2.0 uporabnikom omogoča medsebojno sodelovanje in s tem prispevanje k vsebini spletnih strani, v nasprotju s statičnimi spletnimi stranmi, kjer so uporabniki omejeni na pasivno prebiranje informacij, ki jih je nekdo priskrbel.

Splet 2.0 je torej splet druge generacije, ki vključuje bloge, socialne zaznamke, Ajax tehnologijo, mobilnost, RSS, CSS, dizajn, standardizacijo, dostopnost, modularnost, XML, ATOM, XHTML, kar skupaj tvori prijazen splet, katerega vsebina je za uporabnika lažje dostopna in privlačnejša, predvsem zaradi preprostejše uporabe ter novejšega izgleda. [1]

Poglavje 3

Spletni brskalniki

Spletni brskalniki so računalniške aplikacije, ki uporabniku omogočajo dostop do svetovnega spleta tako, da (X)HTML dokumente prenesejo s spletnega strežnika preko protokola HTTP (angl. Hypertext Transfer Protocol) in ga odjemalcu prikažejo kot grafični dokument, ki lahko vsebuje multimedijske vsebine (slike, video ter zvok) in nadbesedilne povezave do drugih dokumentov. Besedilnim spletnim dokumentom pravimo spletna stran, smiselno povezanim spletnim mestom pa spletišče.

Temelje za prvi spletni brskalnik, WorldWideWeb, ki je začel delovati leta 1991, je postavil Tim Berners-Lee že v poznih 80. Letih. Prvi brskalnik, ki je bil obenem tudi urejevalnik spletnih strani, je združil najrazličnejše nove in že obstoječe tehnologije programske in strojne opreme. [4, 5]

Eden izmed prvih spletnih brskalnikov z grafičnim uporabniškim vmesnikom, ki je podpiral HTML, pa je bil Mosaic. Razvila sta ga Marc Andreessen in Eric Bina leta 1993. Zaradi prijaznega in enostavnega uporabniškega vmesnika, zanesljivosti in enostavne namestitve se je hitro razširil med ljudmi. Tako je hitro postal priljubljen tudi globalno in močno prispeval k razvoju interneta.

Dobra zasnova Mosaica pa se kaže tudi po tem, da še današnji vodilni spletni brskalniki, kot sta Microsoftov Internet Explorer in Mozillin Firefox, temeljijo prav na Mosaicu, saj so obdržali veliko značilnosti njegovega uporabniškega vmesnika in delovanja. [6]

Sprva so spletni brskalniki prikazovali le zelo preproste različice HTML-ja.

Hiter razvoj spletnih brskalnikov je privedel do razvoja nestandardnih različic HTML, ki je vodil do težav s prikazovanjem spletnih strani. Da bi rešili težavo je HTML postal standard, kot ga poznamo danes. Sodobni spletni brskalniki tako prikazujejo kombinacijo standardnih HTML ter novejših XHTML standardov, kateri naj bi se pravilno prikazovali v vseh spletnih brskalnikih.

Današnji spletni brskalniki so v primerjavi s prvotnimi brskalniki veliko zmogljivejši, varnejši, stabilnejši ter prijaznejši do uporabnikov. Podpirajo najnovejše tehnologije ter spletne standarde. Vse te pozitivne lastnosti pa omogočajo sodobnim spletnim razvijalcem razvoj večpredstavnostnih, interaktivnih in vedno hitrejših ter uporabnejših spletnih aplikacij.

Poglavje 4

Spletne aplikacije

Najprej si poglejmo kaj spletne aplikacije sploh so in v čem se razlikujejo od običajnih namiznih računalniških aplikacij.

Razvoj računalniških aplikacij se je pričel z razvojem namiznih aplikacij, katere pa so omejene na lokalno uporabo na namiznih računalniških sistemih.

Za namizno aplikacijo danes označujemo vsako aplikacijo, katero namestimo in izvajamo lokalno na namiznem ali prenosnem računalniku in jo tako uporabljamo za opravljanje določenih nalog. Nekatere namizne aplikacije sicer lahko hkrati uporablja tudi več uporabnikov, kateri pa morajo biti povezani v isto lokalno omrežje.

Ob pojavu spletnih aplikacij pa so le-te kmalu postale zelo priljubljene med uporabniki in tako počasi začele zamenjevati običajne namizne aplikacije. Spletne aplikacije omogočajo delo na daljavo, enostavno povezljivost, dostop do oddaljenih in porazdeljenih virov, enostavnejše vzdrževanje itd. Je pa potrebno v spletnih aplikacijah nekoliko bolje poskrbeti glede varnosti.

Visoka stopnja porazdeljenosti nam omogoča dostop do večih podatkovnih virov, večjo dosegljivost aplikacij in učinkovito uporabo sistemskih virov. Visoka stopnja povezljivosti pa omogoča medsebojno sodelovanje aplikacij in poveča pretok podatkov in informacij. Skratka uporabnikom iz dneva v dan nudijo večje prednosti od namiznih, kar kaže na še večji delež uporabnikov v prihodnosti.

Spletne aplikacije so aplikacije, do katerih dostopamo preko spletnega br-

skalnika preko omrežja internet ali intranet. Temeljijo na arhitekturi "odjemalec-strežnik", kjer spletni brskalnik nastopa v vlogi odjemalca. Razvoj takih aplikacij so omogočile tehnologije dinamičnih spletnih strani kot so CGI, ISAPI, NSAPI, Servleti, ASP, PHP, JSP idr. Te tehnologije delujejo tako, da za vsako odjemalčevo zahtevo ustvarijo novo spletno stran in jo nato odjemalcu vrnejo kot odgovor. [7]

V preteklosti so razvijalci spletne aplikacije implementirali s tehnologijo CGI, ki se slabše prilagaja večjim obremenitvam. Programi in CGI skripte močneje obremenjujejo strežnik, saj vsaki zahtevi uporabnika sledi njeno procesiranje na strežniku, ki nato odjemalcu vrne celotno spletno stran. Ves ta čas mora uporabnik čakati na odgovor in je tako priča pogostim časovnim zakasnitvam. Šele po zaključku celotnega cikla ima uporabnik možnost izvršiti naslednji zahtevek. Takemu izvajanju pravimo sinhrono izvajanje po principu "zahtevek-odgovor". [8]

Uporaba takega modela se v praksi izkaže za manj zmogljivega. Novejše tehnologije zato omogočajo asinhrono izvajanje v nitih, ki se lahko predpomnijo in tako razbremenijo strežnik. Primer take tehnologije je pristop Ajax (angl. Asynchronous JavaScript and XML), pri katerem si spletne aplikacije s strežnikom v ozadju asinhrono izmenjujejo samo podatke, ki jih dejansko potrebujejo, pri čemer ni potrebe po ponovnem nalaganju celotne spletne strani. Posledica tega je uporabniški vmesnik, ki se veliko hitreje odziva na vnose uporabnika. [9]

Poglavje 5

Uporabljene tehnologije pri razvoju spletne aplikacije

V tem poglavju so opisana nekatera orodja, pristopi in tehnologije, katere so bile uporabljene pri razvoju naše spletne aplikacije.

5.1 Programski jezik PHP

PHP (trenutno tričrkovni rekurzivni akronim za PHP: Hypertext Preprocessor, izvorno pa Personal Home Page Tools, slovensko orodja za osebno spletno stran) je razširjen odprtokodni programski jezik, ki je namenjen strežniški uporabi oziroma za razvoj dinamičnih spletnih vsebin. Lahko ga primerjamo s sistemi, kot so Microsoft-ov ASP (angl. Active Server Pages), Sun-ov JSP (angl. JavaServer Pages) ter sistemom CGI (angl. Common Gateway Interface).

Podoben je običajno strukturiranim programskim jezikom, najbolj pa jezikoma C in Perl. Izkušenim programerjem pa dovoljuje razvijanje kompleksnih sistemov brez dolgega učenja. [10]

PHP običajno teče na spletnem strežniku, kjer sprejema PHP izvorno kodo za vhod in generira spletno stran kot izhod. PHP interpreter izvaja le kodo, ki je v HTML kodo vrinjena med posebnimi mejami "**<?php**" in "**?>**" (slika 5.1).

Poleg izvajanja izvorne kode na strani strežnika (angl. Server-side scripting), kot del PHP-ja členimo tudi zaganjanje skript v ukaznem načinu (angl.

Command line scripting) ter razvoj namiznih aplikacij (angl. writing Desktop applications).

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
 <html>
 <head>
  <title>PHP</title>
 </head>
 <body>
  <?php echo "Pozdravljen, Svet!"; ?>
 </body>
 </html>
```

Slika 5.1: Primer PHP kode.

Pri razvoju spletne aplikacije smo se odločili za uporabo PHP-ja predvsem zaradi njegove razširjenosti in enostavnosti.

5.1.1 Ogrodje Zend Framework

Ogrodje Zend Framework, katerega nekateri poimenujejo kar PHP knjižnica, nudi množico komponent za hitrejši razvoj, lažje upravljanje ter nadgrajevanje aplikacij. Temelji na arhitekturi MVC (angl. Model-View-Controller) (slika 5.2), kar omogoča enostavnejše vzdrževanje aplikacij, preglednejšo programsko kodo ter lažjo delitev nalog na več razvijalcev, zato je zelo priljubljen način izgradnje spletnih strani in aplikacij. [11]

Delo v ogrodju Zend Framework je razdeljeno na tri dokaj neodvisne dele, kar je še posebej dobrodošlo, kadar dela na projektu več kot en razvijalec.

Razvoj: Razvijalci, ki razvijajo model (angl. MVC model), imajo tipično znanje s področja PHP-ja, podatkovnih baz, algoritmih, arhitekturi, določajo pa tudi način interakcije s podatki. [12]
MVC model je del aplikacije, ki skrbi za hranjenje in obdelavo podatkov. Ker običajno predstavlja programski približek procesov, ki tečejo v realnem svetu, se za njegovo definiranje lahko uporabijo običajne tehnike modeliranja (najpogosteje razredni diagrami in diagrami stanj). [13]

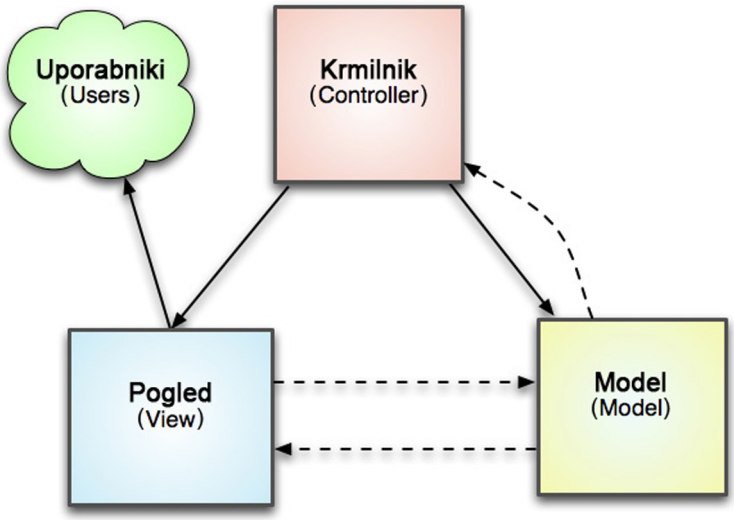
Oblikovanje: Oblikovalci skrbijo za pogled (angl. MVC view) in so odgovorni za izgled aplikacije. Grafični oblikovalci morajo biti dobro podkovani v tehnologijah, kot so HTML, CSS in JavaScript. Tipično so v interakciji z

naročnikom, saj lahko tako kar najuspešnejše sledijo naročnikovim željam, jih izpolnjujejo ter po potrebi tudi posredujejo ostalim udeležencem projekta. [12]

MVC pogled dostopa do podatkov modela ter jih ustrezno prikazuje. Poleg tega pa je naloga pogleda tudi, da se ustrezno osvežuje, v primeru sprememb modela, kar se lahko doseže na dva načina. V primeru uporabe *modela poriva* (angl. push model), za spremembe pogleda poskrbi model tako, da se pogled registrira v modelu. V textit modelu potega (angl. pull model) pa sam pogled kliče model, ko želi prejeti sveže podatke. [13]

Integracija: Integracija se zgodi v krmilniku (angl. MVC controller), kjer se združi delo razvijalcev in oblikovalcev. Zadolženi za integracijo imajo tipično manj izkušenj kot razvijalci. Njihova naloga je skrb za dinamičnost statičnih predlog oblikovalcev, skrbijo pa tudi za sistem posredovanja podatkov iz poizvedb aplikaciji, torej za del, ki podatke iz obrazcev posreduje modelu, interpretira rezultate in jih poda pogledu. [12]

Naloga ***MVC krmilnika*** je, da se spremembe v uporabniškem vmesniku ustrezno prenesejo na model. Akcija modela je lahko, na primer, zagon poslovnega procesa ali sprememba stanja modela. Krmilnik, na podlagi uporabniških akcij in akcij modela, preusmeri podatke na ustrezen pogled. [13]



Slika 5.2: MVC arhitektura.

Ogrodje Zend Framework se MVC vzorca ne drži popolnoma, a tega mu ne štejemo v slabost, kajti s tem omogoča uporabo funkcij Zend Framework-a brez neposredne uporabe ogrodja in celo skupaj z drugimi ogrodji. Seveda pa

se je možno striktno držati MVC vzorca, v kolikor razvijalec to želi.

Zend Framework že vsebuje nekatere funkcije, ki se pogosto pojavljajo v aplikacijah. S tem nam omogoča, da pridobljeni čas, katerega bi porabili za pisanje takih funkcij, posvetimo razvoju drugih funkcionalnosti naše spletne aplikacije.

Ogrodje Zend Framework med drugimi vključuje rešitve za:

- avtorizacijo - *Zend_Auth*,
- dostop do podatkovnih baz - *Zend_Db*,
- globalno postavitev spletne strani - *Zend_Layout*,
- upravljanje sej - *Zend_Session*,
- implementacijo pogleda MVC - *Zend_View*,
- osnovno vzpostavitev aplikacije - *Zend_Application*.

Zend Framework ponuja še veliko večjo množico komponent, našteli smo le nekaj rešitev, pomembnih za našo aplikacijo. [12]

5.2 Pristop AJAX

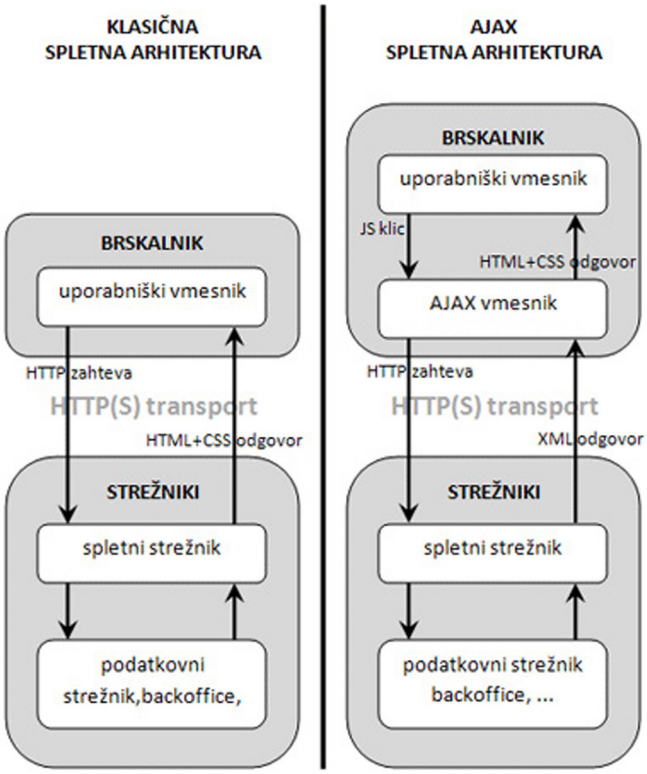
Ajax (angl. Asynchronous JavaScript and XML) je množica medsebojno povezanih spletnih razvojnih tehnologij, uporabljenih za ustvarjanje interaktivnih spletnih aplikacij.

Ker si spletne aplikacije grajene po Ajax pristopu izmenjujejo podatke s strežnikom asinhrono v ozadju, ni potrebe po ponovnem nalaganju celotne spletne strani ob vsaki zahtevi, kakor je to pri klasičnem (PHP) pristopu. Bistveno razliko opazimo na strani odjemalca, kjer imamo v Ajax modelu tudi Ajax vmesnik. Posledica tega so manjši odzivni časi, kar daje uporabniku občutek hitrega odzivanja na podane zahteve in s tem bolj interaktiven način uporabe spletne aplikacije. [14]

Podatki se prenašajo s pomočjo objektov XMLHttpRequest ali s pomočjo Remote Scriptinga (v starejših brskalnikih, ki ne podpirajo Ajaxa). Uporaba

tehnologij Ajax je značilna za Splet 2.0. Navkljub imenu pa uporaba tehnologij JavaScript in XML ni pogoj za izvajanje Ajaxa. [14]

Ajax aplikacije dajejo vtis, kot da v celoti tečejo na računalniku uporabnika. Običajna spletna aplikacija namreč za vsako spremembo na strani pošlje zahtevo HTTP, strežnik pa kot odgovor pošlje celotno spletno stran (slika 5.3). Brskalnik mora zato osvežiti celotno stran, s tem pa pride do motečega obnavljanja strani. Spletna aplikacija pa je zato počasna in čisto nič podobna namiznim. [14]



Slika 5.3: Primerjava spletnih arhitektur.

Aplikacije Ajax so prirejene za generiranje poizvedb za strežnik tako, da pošljejo samo tiste podatke, ki jih dejansko potrebujejo. Klic se opravi kot asinhrona komunikacija, torej medtem ko aplikacija čaka podatke iz strežnika, lahko uporabnik nemoteno uporablja spletno stran. Ko so podatki pripravljeni, določena funkcija v JavaScriptu prikaže podatke na strani, brez potrebe po ponovnem nalaganju.

Posledica tega je uporabniški vmesnik, ki se veliko hitreje odziva na vnose uporabnika. Razlog za vpeljavo te tehnologije je tudi dejstvo, da se med odjemalcem (brskalnikom) in strežnikom prenese veliko manj podatkov ter da poteka nalaganje podatkov asinhrono. Poleg tega se zmanjša obremenitev spletnega strežnika, ker se veliko obdelave lahko naredi na strani odjemalca. [14]

Pojem Ajax je leta 2005 skoval in prvič omenil Jesse James Garrett v svojem članku *Ajax: A New Approach to Web Applications*. Tehnologije, na katerih temelji, so se začele razvijati že v letu 1996, ko je podjetje Microsoft v svojem spletnem brskalniku Internet Explorer predstavilo konstrukt IFrame. V letu 1998 Microsoft predstavi Remote Scripting, kjer podatke bere javanska aplikacija, s katero komunicira odjemalec s pomočjo programskega jezika JavaScript. Nadalje leta 1999 Microsoft ustvari XMLHttpRequest objekt kot kontrolnik ActiveX v brskalniku Internet Explorer 5.0, čemur so kmalu sledili ustvarjalci brskalnikov Mozilla in Safari. Aprila 2006 je konzorcij W3C (World Wide Web Consortium) pripravil prvi osnutek specifikacije XMLHttpRequest z željo ustvariti uradni standard za spletne strani. [14]

Ajax torej ni nova tehnologija, ampak gre za množico že znanih tehnologij, ki jih lahko zasledimo v večini sodobnih spletnih brskalnikih. V svojem članku je Jesse James Garrett napisal, da se pojem Ajax nanaša na naslednje tehnologije:

- HTML ali XHTML ter CSS za predstavitev,
- DOM (angl. Document Object Model) za dinamično prikazovanje vsebine in interakcijo s podatki,
- XML in XSLT za delo s podatki,
- objekt XMLHttpRequest za asinhrono komunikacijo,
- ter JavaScript, ki povezuje vse te tehnologije (JavaScript služi tudi kot vmesnik med posameznimi komponentami). [15]

Zaradi napredka v razvoju in tehnologijah pa je sedaj del Ajax-a tudi sledeče:

- poleg jezika JavaScript kot skriptni jezik na strani odjemalca nastopa tudi VBScript

- XML in XSLT nista več obvezna za izmenjavo in obdelavo podatkov. Pogosto se kot alternativa uporablja JSON (angl. JavaScript Object Notation), čeprav je možno uporabiti tudi druge načine predstavitev podatkov, na primer HTML ali pa celo golo besedilo (angl. plain text).

Ajax rešitve ne vsebujejo nujno vseh omenjenih tehnologij. Zadošča že uporaba XHTML, DOM, JavaScript in *XMLHttpRequest objekta*. XHTML skrbi za opis strukture tekstovnih informacij v dokumentu, JavaScript in DOM poskrbita za naslavljanje določenih delov spletne strani in ažuriranje vsebine, *XMLHttpRequest objekt* pa pošlje HTTP poizvedbo na strežnik in počaka na odgovor.

Poleg tehnologij na strani odjemalca so potrebne še strežniške komponente, ki skrbijo za izvedbo poslovne logike in vračajo rezultate. Strežniške komponente so neodvisne od Ajaxa, saj le-ta zahteva ustrezen podatkovni format. Tako lahko na strežniku izvajamo .NET, JSP ali PHP aplikacije.

5.2.1 Označevalni jezik (X)HTML in Stilske predloge CSS

Jezik za označevanje nadbesedila, oziroma HTML (angl. Hyper Text Markup Language) je označevalni jezik za izdelavo spletnih strani in predstavlja osnovo spletnega dokumenta. S pomočjo HTML-ja ustvarimo strukturo in semantično ureditev dokumenta. [16]

V spletne brskalnike je vgrajen HTML interpreter, kateri pri nalaganju dokumenta posebne HTML značke obdela in vsebino dokumenta prikaže kot spletno stran. Interpreter poleg prikaza besedila omogoča tudi prikaz slik, obrazcev za vnos podatkov ter tabel. V besedilo lahko vključimo tudi povezave z drugimi dokumenti v svetovnem spletu. Zaradi njegove preprostosti pa ga lahko pišemo v vsakem urejevalniku besedil ter ga lahko kombiniramo tudi z drugimi programskimi jeziki.

XHTML (angl. Extensible HyperText Markup Language) je razširjeni označevalni jezik, ki ima enak namen kot HTML, vendar je usklajen s sintakso XML (angl. Extensible Markup Language) ter bolj natančno določen. Lahko bi ga označili kot posodobljeno različico jezika HTML.

Medtem ko je HTML različica SGML-a (angl. Standard Generalized Markup Language), zelo prilagodljivega označevalnega jezika, je XHTML prilagojen XML-u, strožji podmnožici SGML-a. Ker mora biti dokument XHTML tvorjen po določenih pravilih, ga lahko samodejno razčlenimo kar s preprostimi razčlenjevalniki XML, za razliko od HTML, ki potrebuje razmeroma zapleten, prizanesljiv in večinoma po meri narejen razčlenjevalnik. XHTML 1.0 je postal priporočilo inštituta W3C (angl. World Wide Web Consortium) 26. januarja 2000, XHTML 1.1 pa 31. maja leto kasneje. [17] Trenutno je v razvoju XHTML 5, kot del specifikacije HTML 5.

Kot smo že omenili, lahko jezika HTML ter XHTML kombiniramo tudi z drugimi programskimi jeziki. Eden izmed njih je označevalni jezik CSS oziroma stilske predloge CSS (angl. Cascading Style Sheets), katere določajo izgled spletne strani. Z njimi določamo obliko pisave, velikosti črk ter vizualno predstavitev spletne strani. Če naj bi HTML predstavljal semantično strukturo in smiselno hierarhijo dokumenta, pa CSS predstavlja predstavitveno vlogo. Z drugimi besedami, CSS omogoča ločevanje dokumenta z vsebino od dokumenta, ki vsebuje informacije o predstavitvi. Prednost tega je, da se s takim ločevanjem izognemo nepotrebnemu mešanju programske kode z opisno. Spletna stran je tako fleksibilnejša, zmanjša pa se tudi zahtevnost strani. Vzdrževanje take programske kode je enostavnejše.

CSS nam pomaga predstaviti spletno stran z enako vsebino v več različnih oblikah. Tako lahko spletno stran, ki jo bomo npr. natisniti, predstavimo v taki obliki, ki bo primerna za tiskanje.

Standard CSS podpira večina novejših spletnih brskalnikov. Spletni brskalniki, ki pa standarda CSS morda ne podpirajo, pa bodo v primeru rabe CSS-a prikazali običajen, neoblikovan HTML dokument. [18]

5.2.2 Model DOM (Document Object Model)

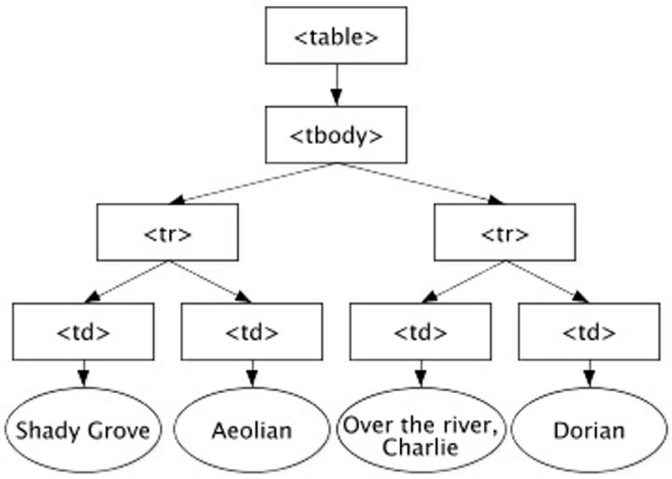
DOM (angl. Document Object Model) predstavlja platforemsko ter jezikovno neodvisen vmesnik, ki zagotavlja urejanje vsebin, struktur ter slogov znotraj posameznega dokumenta. [19]

DOM je namenjen predvsem XML dokumentom, kjer so podatki strukturirani po določenih standardih. V modelu DOM so podatki iz dokumenta (tekst, elementi in atributi) (slika 5.4) predstavljeni kot vozlišča v drevesni strukturi (slika 5.5). V vrhu drevesa se nahaja korenski element dokumenta,

```
<table>
  <tbody>
    <tr>
      <td>Shady Grove</td>
      <td>Aeolian</td>
    </tr>
    <tr>
      <td>Over the River, Charlie</td>
      <td>Dorian</td>
    </tr>
  </tbody>
</table>
```

Slika 5.4: Struktura podatkov v modelu DOM.

pod njim pa so hierarhično urejena ostala vozlišča. Vozlišča, ki se v drevesu nahajajo pod trenutnim vozliščem, se imenujejo potomci. Z izjemo korenskih elementov, ki so brez staršev in listov, ki so brez otrok, ima vsako vozlišče vsaj enega starša in vsaj enega otroka. Za vozlišča, ki imajo skupnega starša pa pravimo, da so v sorodu. [20]



Slika 5.5: Drevesna struktura v modelu DOM.

Kot že rečeno, spletni brskalnik ob prikazu spletne strani zgradi hierarhično obliko HTML elementov (DOM drevo). Tako vsak element postane kontrola v DOM strukturi, ki jo lahko spreminjamo in vsebuje dogodke, lastnosti in

metode posameznega elementa. [19]

Ker pa skoraj vsak spletni brskalnik vključuje DOM vmesnik nekoliko drugače, nastajajo težave pri prikazu enakih spletnih strani v različnih brskalnikih. V našem primeru smo spletno aplikacijo gradili in optimizirali za prikaz v brskalniku Mozilla Firefox.

5.2.3 Razširljiv označevalni jezik XML in jezik XSLT

XML (angl. Extensible Markup Language) je razširljiv označevalni jezik za zapisovanje strukturiranih dokumentov in podatkov v spletu, hkrati pa velja za enega najpogostejših standardov za zapis podatkov za prenos po internetu. [21]

XML je zelo je uporaben za komunikacije, saj ima zelo preprosto in pregledno zgradbo. Najbolj prilagodljive rešitve omogoča podpora z obeh strani, tako strežniške kot tudi odjemalčeve. Mogoče je vzpostaviti direktno povezavo s strežniško stranjo ter uporabniškim vmesnikom, kar omogoča ločitev teh dveh aplikacijskih nivojev. Pri taki ločitvi imamo ločene dele aplikacije, na katerih lahko hkrati nemoteno razvija več razvijalcev. [21]

Razširljivost jezika dosežemo s samim načinom označevanja vsebine, ki temelji na značkah (angl. Tags), katere pojasnjujejo pomen podatkov. Značke poimenujemo kar sami. Omejeni smo le s sintakso jezika, katere osnovo tvorijo elementi. Vsebino elementov pa tvorijo znaki, drugi elementi, ukazi za procesiranje, reference do entitet in znakov, atributi ter komentarji. Sintaktična sestava spominja na jezik HTML iz katerega se je XML tehnologija tudi razvila. XML poleg razširljivosti poenostavi izmenjavo podatkov in omogoča jasno ločitev vsebine od prikaza, kar v HTML-ju skoraj ni mogoče. [21]

XML je razdeljen na 3 dele:

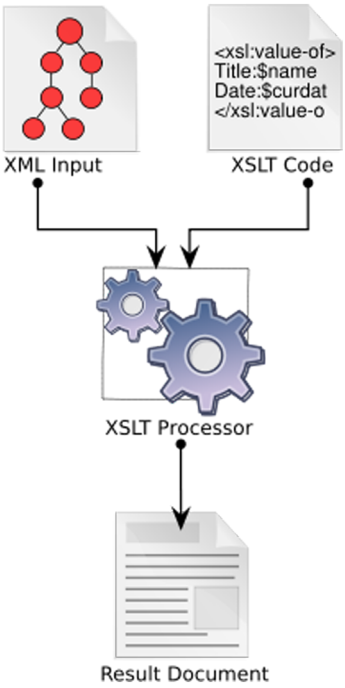
- podatkovni (vanj shranimo podatke, označene z značkami),
- deklarativni (pri dodajanju novih podatkov služi za nekakšen opis značk),
- predstavitveni (z njim oblikujemo izpis podatkov). [21]

Kot smo že omenili, so dokumenti XML strukturirani, navadno po neki shemi, po kateri se dokumenti med seboj razlikujejo. Kadar pa potrebujemo

podatke iz obstoječega dokumenta v drugačni obliki si pomagamo s pretvorbami, katere ponuja jezik XSLT (angl. Extensible Stylesheet Language Transformations). [23]

XSLT je torej deklarativen programski jezik za pretvarjanje dokumentov XML v druge dokumente XML in je eden izmed sestavnih gradnikov za XSL. Standard XSL je bil namenjen tako pretvorbi dokumentov XML za izmenjavo, kot tudi oblikovanju podatkov iz dokumentov za prikaz.

Pretvorba dokumentov XML s predlogo XSLT v ustrezni izhodni dokument poteka tako, da se vzame izvorni dokument, ki se ga nato uporabi za drevo vozlišč, in da se iz njega ustvari drugi dokument XML v neki drugi obliki (slika 5.6). Seznam slogov XSLT ponuja pravila in obliko, ki določa, kako se gradi izhodni dokument. Če je izhodni dokument v ustrezni obliki, kot je HTML, se ga lahko uporabi za predstavitev izvornega dokumenta. [22]



Slika 5.6: Pretvorba dokumentov XML s predlogo XSLT.

XSLT pretvorbo navadno uporabimo, kadar želimo podatke iz dokumentov XML prikazati v drugačni obliki. V praksi se XSLT največkrat uporablja kot

nek vmesni format podatkov v komunikaciji med različnimi računalniškimi okolji. Če lahko podatke zapišemo v obliki dokumenta XML, jih lahko s pomočjo pretvorbe XSLT prilagodimo ustrezni obliki in vsebini, ki ju zahteva zunanje računalniško okolje. [23]

5.2.4 Objekt XMLHttpRequest (XHR)

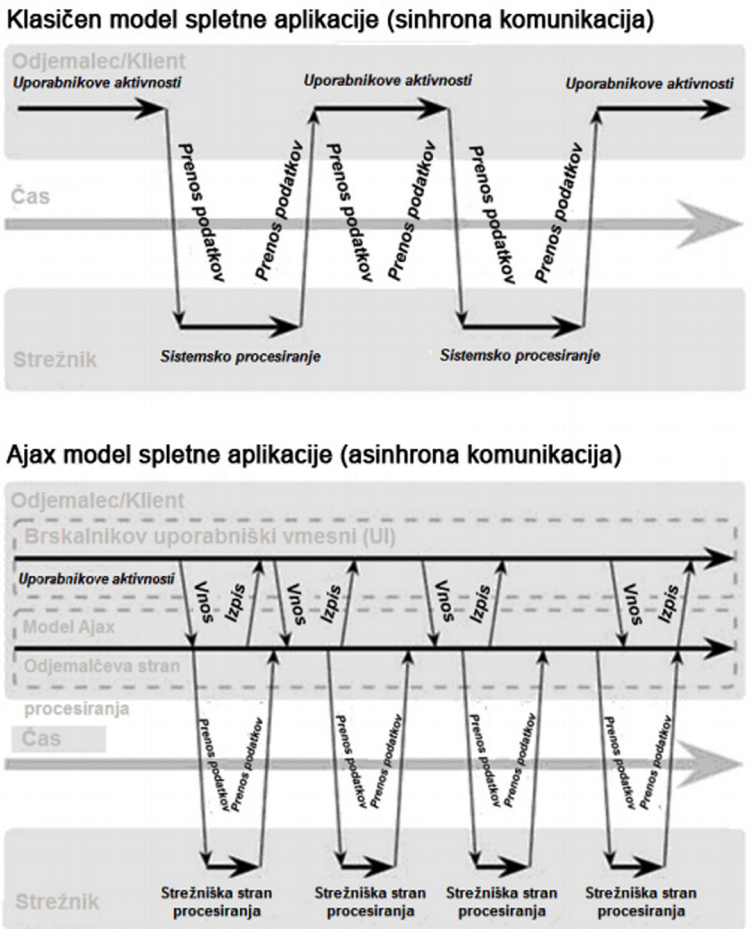
Kot smo že omenili, se zahtevki med odjemalcem in strežnikom pri Ajax pristopu prenašajo asinhrono v ozadju, pri čemer ima objekt XMLHttpRequest (XHR) pomembno vlogo. Objekt XMLHttpRequest je vmesnik, ki ga lahko spletni brskalniki, kateri podpirajo skriptne jezike (npr. JavaScript), uporabljajo za prenos podatkov med spletnim strežnikom in brskalnikom v XML ter drugih oblikah. [24]

Asinhron prenos podatkov predstavlja dvosmerno komunikacijo izvedeno s časovno zamudo (slika 5.7). Odgovor na podatke ni časovno omejen in se izvede takrat, ko so le-ti na voljo. Torej, ker ni potrebe po čakanju strežnika na odgovor, lahko normalno upravljamo z drugimi podatki, medtem pa bomo vseeno sprejeli odgovor s strežnika. Asinhron prenos že v osnovi vsebuje model zahteva/odgovor (angl. request/response), ki je v Ajaxovem razvojnem okolju del objekta XMLHttpRequest, kar pomeni, da prenos podatkov po modelu zahteva/odgovor ni predhodno določen po posameznih intervalih. [25]

Objekt XMLHttpRequest torej s pomočjo JavaScripta omogoča izvajanje asinhronih zahtevkov s HTTP strežniških zahtev. To nam omogoča ustvarjanje HTTP zahtevkov, katerih odgovor ne osveži celotne spletne strani v brskalniku. Ker se prenosi vršijo v ozadju, uporabnik pri tem opazi samo spremembo vsebine določenega dela spletne strani, katerega smo želeli spremeniti. S tem zagotovimo odzivno uporabniško izkušnjo ter skrajšamo čas, ki bi bil porabljen pri osveževanju celotne spletne strani, hkrati pa lahko tudi razbremenimo pasovno širino. Svetovno znane spletne aplikacije, kot so Gmail, Google Maps, Facebook in veliko drugih, se za zadovoljitev želja svojih uporabnikov poslužujejo prav enake metode kot smo zapisali zgoraj. [24]

5.2.5 Programski jezik JavaScript

JavaScript je objektno orientiran skriptni programski jezik, ki se izvaja na strani odjemalca v spletnih brskalnikih. Razvil ga je Netscape z namenom pomagati spletnim razvijalcem pri razvoju interaktivnih spletnih strani.



Slika 5.7: Prikaz sinhrone in asinhrone komunikacije.

Jezik je bil razvit neodvisno od Jave, vendar si z njo deli številne lastnosti in strukture. JavaScript lahko v sodelovanju s HTML-jem poživi spletno stran z dinamičnim izvajanjem. Danes jezik JavaScript podpirajo vsi novejši spletni brskalniki.

Sintaksa jezika JavaScript ohlapno sledi programskemu jeziku C. Prav tako kot C, tudi JavaScript nima vgrajenih vhodno izhodnih funkcij, zato je izvedba teh funkcij odvisna od gostitelja. JavaScript pa za razliko od C-ja, ni tipiziran programski jezik. JavaScript je interpretiran jezik, kar pomeni, da se izvorna koda ne prevede v izvršno, ampak se prenese s strežnika na odjemalca, kjer se v brskalniku neposredno prevede.

JavaScript je bil v osnovi zasnovan za pisanje manjših skript za izvedbo določenih funkcionalnosti. V začetku se je uporabljal samo vgrajen znotraj posameznega dokumenta HTML kot enostavna, nekaj vrstična programska koda. V začetku se je uporabljal samo vgrajen znotraj posameznega dokumenta HTML kot enostavna, nekaj vrstična programska koda. Danes je način uporabe drugačen. To se kaže predvsem pri sodobnih spletnih aplikacijah, kjer se njihovo delovanje v veliki meri zanaša prav na jezik JavaScript.

Jezik JavaScript pa se lahko uporablja na različnih področjih, od nastavitev administrativnih opravil v okolju Windows do manipulacij datotek PDF. Aplikacije imajo svoje objektne modele, ki zagotavljajo dostop do gostiteljevega okolja, samo jedro jezika JavaScript pa je v vseh aplikacijah večinoma enako.

JavaScript postaja zelo pomemben tudi na strani strežnika, pri sodelovanju s tehnologijami HTML, CSS in XML. [26]

Pri razvoju naše spletne aplikacije smo si delo olajšali z uporabo JavaScript knjižnice jQuery.

5.2.5.1 Knjižnica jQuery

jQuery je knjižnica vnaprej pripravljenih JavaScript funkcij. Napisal jo je John Resig in leta 2006 tudi predstavil. Kot tudi druge knjižnice je bil jQuery napisan z razlogom, da razvijalcem olajša delo. To je jasno tudi po njegovi filozofiji, ki se glasi: “Write less, do more”.

jQuery je usmerjen predvsem v delo s tehniko DOM in implementiranje funkcije za izvajanje s tem povezanih nalog (spreminjanje lastnosti CSS, animiranje elementov, prejemanje vsebin preko Ajax-a, ipd). Poleg tega zagotavlja še nekatere druge metode (za urejanje nizov, premikanje po tabelah, itd.), ki niso povezane s tehniko DOM, vendar so splošno koristne. Ne obravnava koncepta dedovanja, razredov, podrazredov, vmesnikov, obravnava pa koristne metode za prirojene osnovne podatkovne tipe, funkcije, tabele jezika Javascript.

Ne glede na to, da ima večina današnjih spletnih brskalnikov vgrajen svoj lasten interpreter JavaScript, vsi ukazi, ki jih nudi knjižnica jQuery, delujejo na

vseh teh spletnih brskalnikih, za kar skrbi knjižnica sama. To pa je tudi eden ključnih razlogov, da jQuery velja za eno najbolj priljubljenih programskih ogrodij za JavaScript. [27]

5.2.5.2 Aktivni zaznamki (Bookmarklets)

Aktivni zaznamek (v nadaljevanju bookmarklet) je majhen programček, napisan v jeziku JavaScript, ki je kot URL naslov shranjen v zaznamku v spletnem brskalniku ter omogoča izvajanje določenih funkcionalnosti. Izvaja se v spletnem brskalniku. Bookmarklet je sestavljenka besed zaznamek (angl. bookmark) in aplet (angl. applet).

Ker lahko bookmarkleti trenutno spletno stran spreminjajo in preiskujejo, so primerni predvsem kot hitro dostopna orodja, ki nadgradijo posamezno spletno stran ali razširijo zmogljivost spletnega brskalnika. Vsebina bookmarkleta je URL naslov, podan kot *javascript:URL*. Ker spletni brskalniki v tem URL naslovu prepoznajo protokol *javascript:*, je ta naslov razčlenjevalniku enak kot ostali URI-ji (angl. Uniform Resource Identifier). Interno brskalnik tako prepozna protokol *javascript:*, zato se ostali del kode izvede kot JavaScript programska koda.

Namestitev bookmarkletov najlažje opravimo tako, da obiščemo spletno stran, katera tovrstne zaznamke ponuja in jih shranimo kot druge, običajne zaznamke. Lahko pa enostavno v spletnem brskalniku kreiramo nov zaznamek in kot URL naslov navedemo programsko kodo v jeziku JavaScript.

Bookmarklete uporabljamo prav tako kot ostale spletne zaznamke, razlika je le v tem, da nas bookmarkleti ob kliku ne preusmerijo na neko drugo spletno stran, ampak ostanemo na trenutni, nad katero se nato izvede programska koda, ki jo zaznamek vsebuje. Bookmarklet aktiviramo oz. poženemo s klikom na zaznamek, kar pa je tudi edini način zagona. Skratka bookmarklet se nikakor ne more izvesti samodejno npr. skupaj ob nalaganju strani, kar pa je tudi eden večjih razlogov, zakaj se bookmarkleti v več kot 15 letih obstoja niso bolje uveljavili. [28]

5.3 SUPB MySQL

MySQL (angl. My Structured Query Language) je odprtokodni sistem za upravljanje zbirk podatkov. Zbirka podatkov je v MySQL strukturirana zbirka po-

datkov. Lahko je karkoli od preprostega nakupovalnega seznama do slikovne galerije ali ogromna količina informacij v omrežju podjetja. Za dodajanje in obdelavo podatkov v zbirki podatkov potrebujemo sistem za upravljanje zbirk podatkov, kot je strežnik MySQL.

MySQL je relacijski podatkovni upravljalni sistem. Relacijska podatkovna zbirka shrani podatke v ločenih tabelah in ne v enem velikem hranilniku. To povečuje hitrost in fleksibilnost. SQL je najbolj uporabljen standardiziran jezik, ki se ga uporablja za dostop do zbirk podatkov in je določen s standardom ANSI/ISO SQL. Standard SQL se je razvijal od 1986 in danes obstaja več različic. Izraz *standard SQL* uporabljamo za poimenovanje trenutne aktualne različice standarda SQL v nekem časovnem obdobju.

Programska oprema MySQL je odprtokodna (angl. Open Source). To pomeni, da ima vsak možnost dostopa do programske opreme in jo lahko priredi za svoje potrebe. Vsakdo lahko prenese programsko opremo MySQL z interneta in ga uporablja, ne da bi moral za to kaj plačati. Če želimo, lahko preučimo izvorno kodo in jo spremenimo tako, da ustreza našim potrebam. Programska oprema MySQL je zaščitena z licenco GPL (GNU, General Public License) s katero se določa, kaj lahko in česa ne smemo početi s programom v različnih situacijah. Če nam GPL ne ustreza ali želimo v kodo MySQL vstaviti v komercialno aplikacijo, lahko kupimo komercialno licenčno različico MySQL.

MySQL je v lasti in sponzoriran s strani Švedskega podjetja SQL AB, ki ima nadzor za večino kode. To je na primer v nasprotju z licenco, ki ga ima npr. Apache, ki je v javni lasti in ga vzdržuje internetna skupnost in avtorske pravice pripadajo posameznim avtorjem. Podjetje vzdržuje razvoj, podporo trženju in vzdrževanju sistemov. Če imamo ustrezno licencirano programsko opremo MySQL, imamo na voljo podporo proizvajalca. Lahko pa imamo stike z množico ljudi, ki po celotnem svetu prostovoljno skrbijo za razvoj programske opreme in pri pomoči.

Sistem MySQL deluje v načinu *odjemalec-strežnik*. Podatkovna programska oprema MySQL je sistem, ki je sestavljen iz strežnika SQL, ki podpira različne odjemalske programe in knjižnice, administrativna orodja in velik razpon aplikacijskih programskih vmesnikov. Na voljo je velika količina dodatne programske opreme za MySQL. [29]

Poglavje 6

Razvoj spletne aplikacije WebLyr

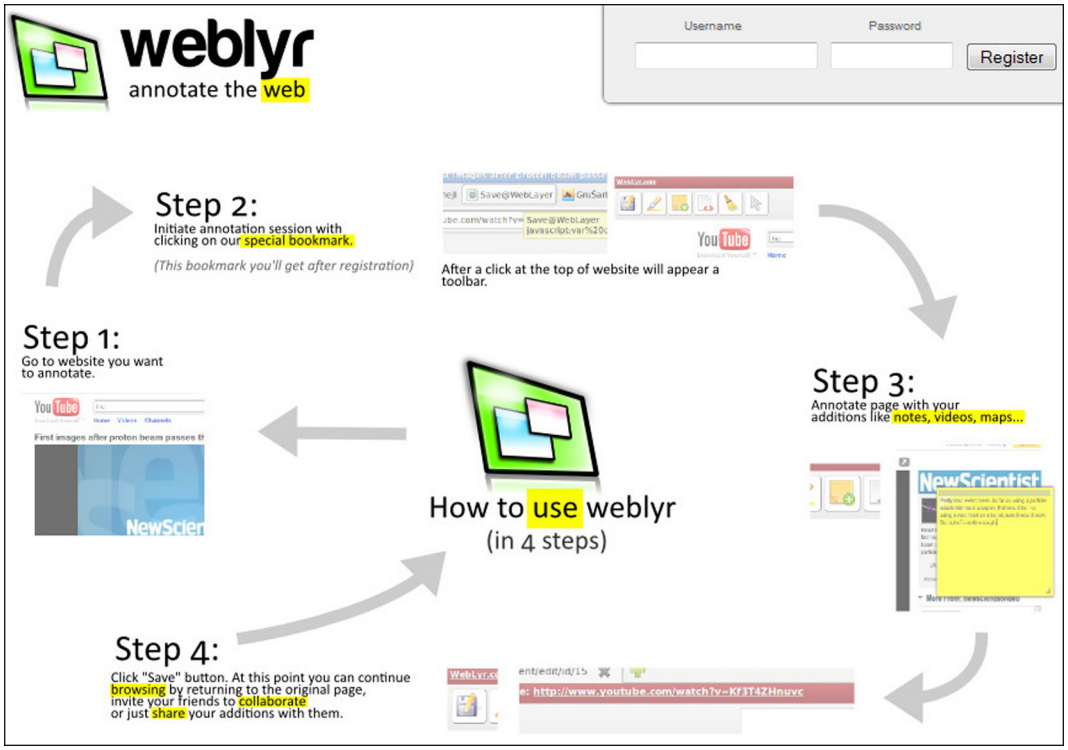
V tem delu naloge bomo na praktičnem primeru prikazali razvoj spletne aplikacije, ki je bila izdelana s pomočjo zgoraj opisanih orodij, pristopov in tehnologij.

Ideja za aplikacijo se je pojavila, ko sva si s kolegom hotela preko spleta izmenjati določene informacije, ker pa nisva našla ustreznega načina izmenjave, sva se odločila za rešitev problema. Spletno aplikacijo, ki rešuje najin problem sva poimenovala WebLyr (slika 6.1). Kakšen problem aplikacija pravzaprav rešuje pa si poglejmo v nadaljevanju.

6.1 Opis problemske domene

Spletna aplikacija naj uporabnikom omogoča medsebojno sodelovanje ter preko spletnih vsebin izmenjavo idej, obvestil ter podobnih informacij.

Aplikacija naj registriranim uporabnikom ob obisku poljubne spletne strani omogoča le-to spletno vsebino poljubno opremiti z raznimi funkcionalnostmi kot so poudarjanje besedila, dodajanje opomb, vstavljanje objektov s pomočjo vrivanja programske kode (angl. Embed), brisanje objektov na spletni strani, uokvirjanje objektov itd. Spletno vsebino naj uporabniki urejajo s pomočjo orodne vrstice, katera jim je na voljo preko bookmarkleta. Registrirani uporabniki lahko tako, po meri opremljeno, spletno stran preko elektronske pošte posredujejo prijateljem, lahko pa jo le shranijo v svojem uporabniškem računu, kjer jim je kasneje na voljo za dodatno urejanje.



Slika 6.1: Spletna aplikacija WebLyr - vstopna stran.

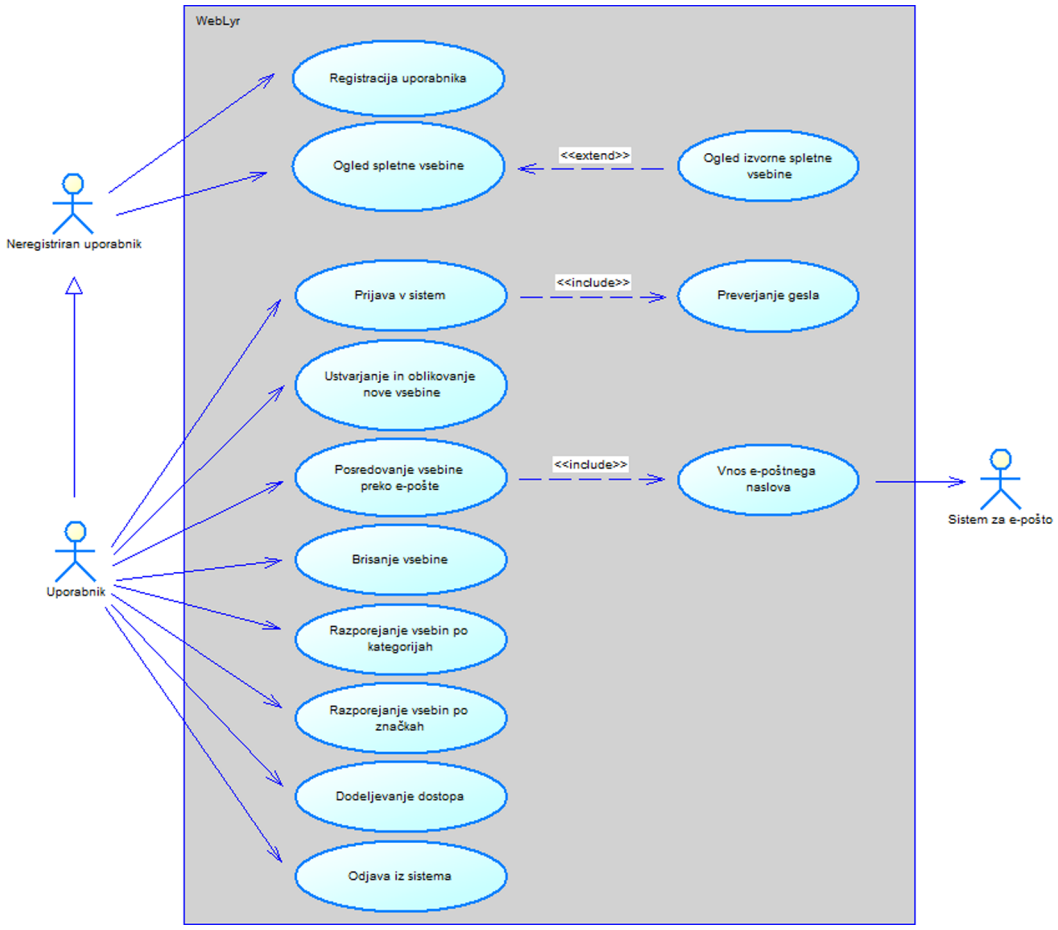
Uporabniški vmesnik naj omogoča pregled shranjenih spletnih vsebin, katerim lahko uporabniki dodajajo značke (angl. Tags) ter jih kategorično razporejajo. Aplikacija naj samodejno beleži vpise ter jih kategorično porazdeljuje po domenah. Vsakemu vpisu lahko uporabniki določijo dostop, kateri je lahko bodisi javen in je tako na voljo tudi drugim uporabnikom, bodisi zaseben in je na voljo samo dotičnemu uporabniku. Preko uporabniškega vmesnika lahko uporabniki vpise tudi iščejo, pregledujejo, urejajo, brišejo ter jih preko e-poštnega naslova pošljejo drugim uporabnikom interneta. Uporabniki lahko bodisi shranijo neko obstoječo spletno vsebino ter jo opremijo z raznimi oznakami itd., bodisi ustvarijo novo, sprva prazno spletno vsebino in jo nato dopolnijo, kasneje pa po želji prav tako opremijo z oznakami.

Spletna aplikacija naj zgoraj opisane funkcionalnosti omogoča le registriranim uporabnikom. Neregistriranim uporabnikom pa naj bo preko URL naslova, ki jim ga lahko posredujejo registrirani uporabniki, na ogled spletna vsebina v takšni obliki, kot jo je shranil posredovalec. Neregistriranim upo-

rabnikom naj bo posredovana spletna vsebina samo na ogled, ostale funkcionalnosti pa naj bodo na voljo le registriranim uporabnikom.

6.2 Diagram primerov uporabe

Funkcionalnosti, ki jih mora obravnavani sistem nuditi, dokumentiramo z modelom primerov uporabe. Ta vsebuje predvidene funkcije sistema (primeri uporabe), njegovo okolje (akterje) in odnose med primeri uporabe in akterji (diagrami primerov uporabe). Najvažnejša vloga modela primerov uporabe je komunikacija. Ta omogoča izmenjavo mnenj o funkcionalnosti in obnašanju sistema med strankami oz. končnimi uporabniki in tistimi, ki ga načrtujejo.



Slika 6.2: Diagram primerov uporabe.

Na sliki (slika 6.2) so prikazani primeri uporabe ob uporabnikovem obisku spletnega servisa. Vsak uporabnik se lahko po predhodni registraciji tudi prijavi v sistem. Kot kaže diagram je uporabnik po prijavi deležen dodatnih funkcionalnosti, katerih pa neregistriran oz. neprijavljen uporabnik ne more koristiti.

6.2.1 Opis toka dogodkov za primer Registracija uporabnika

Kratek opis:

Primer uporabe omogoča uporabniku uspešno registracijo, katera uporabniku nato omogoča prijavo v sistem. Uporabnik je po prijavi deležen dodatnih funkcionalnosti, katerih pa neregistriran oz. neprijavljen uporabnik ne more koristiti.

Osnovni tok:

0. Uporabnik začne postopek registracije uporabnika na vstopni spletni strani aplikacije, kjer izbere povezavo *Register*.
1. Sistem uporabniku prikaže vnosna polja za vnos osebnih podatkov potrebnih za registracijo.
2. Uporabnik vnese osebne podatke v vnosna polja ter pritisne na gumb *Register*.
3. Sistem preveri, ali uporabnik že obstaja.
4. Sistem prikaže obvestilo o uspešni registraciji uporabnika.

Alternativni tok:

0. Uporabnik začne postopek registracije uporabnika na vstopni spletni strani aplikacije, kjer izbere povezavo *Register*.
1. Sistem uporabniku prikaže vnosna polja za vnos osebnih podatkov potrebnih za registracijo.
2. Uporabnik vnese osebne podatke v vnosna polja ter pritisne na gumb *Register*.
3. Sistem preveri, ali uporabnik že obstaja.
4. Sistem že obstoječemu uporabniku prepreči ponovno registracijo ter prikaže obvestilo o neuspešni registraciji.

Predpogoj:

-

Rezultat in spremembe v sistemu:

Registracija je uspešno opravljena v primeru, da se je izvedel glavni tok dogodkov. V nasprotnem primeru ostane stanje v sistemu nespremenjeno.

6.2.2 Opis toka dogodkov za primer Ogled spletne vsebine

Kratek opis:

Primer uporabe omogoča uporabniku ogled spremenjene spletne vsebine preko spletne povezave, katero je bodisi prejel od drugega uporabnika preko e-pošte, bodisi jo je shranil sam. Spletna vsebina je shranjena v podatkovni bazi spletne aplikacije in vsebuje vse spremembe, katere je naredil uporabnik, ki je spletno stran shranil.

Osnovni tok:

0. Uporabnik začne primer uporabe ogled spletne vsebine, ko odpre povezavo do spremenjene spletne vsebine.
1. Sistem prikaže spremenjeno spletno vsebino.

Alternativni tok:

0. Uporabnik začne primer uporabe ogled spletne vsebine, ko odpre povezavo do spremenjene spletne vsebine.
1. Sistem ne najde vsebine za prikaz, zato prikaže obvestilo, da zahtevana spletna vsebina ne obstaja.

Predpogoj:

Uporabnik, ki je spletno vsebino ustvaril, mora v uporabniškem vmesniku označiti vsebino kot javno.

Rezultat in spremembe v sistemu:

Uporabnik si spletno vsebino lahko ogleda v primeru, da se je izvedel glavni tok dogodkov. V nasprotnem primeru si vsebine ne more ogledati in stanje v sistemu ostane nespremenjeno.

6.2.3 Opis toka dogodkov za primer Ogled izvorne spletne vsebine

Kratek opis:

Primer uporabe omogoča uporabniku ogled izvorne spletne vsebine. To je izvorna spletna vsebina in ne vsebuje nikakršnih uporabnikovih popravkov, dodatkov itd. Uporabnik si to vsebino lahko ogleda preko povezave, ki se nahaja v zgornji opravilni vrstici spremenjene spletne vsebine.

Osnovni tok:

0. Uporabnik začne primer uporabe ogled izvorne spletne vsebine, ko odpre povezavo do spremenjene spletne vsebine.
1. Sistem prikaže spremenjeno spletno vsebino.
2. Uporabnik v zgornji opravilni vrstici izbere povezavo do izvorne, nespremenjene spletne vsebine.
3. Sistem prikaže izvorno, nespremenjeno spletno vsebino.

Alternativni tok:

0. Uporabnik začne primer uporabe ogled izvorne spletne vsebine, ko odpre povezavo do spremenjene spletne vsebine.
1. Sistem prikaže spremenjeno spletno vsebino.
2. Uporabnik v zgornji opravilni vrstici izbere povezavo do izvorne, nespremenjene spletne vsebine.
3. Sistem preusmeri spletni brskalnik na izvorno spletno vsebino, vendar ta vsebina ne obstaja več na tem spletnem naslovu, zato brskalnik prikaže obvestilo, da zahtevana spletna vsebina ne obstaja.

Predpogoj:

Uporabnik, ki je spletno vsebino ustvaril, mora v uporabniškem vmesniku označiti vsebino kot javno.

Rezultat in spremembe v sistemu:

Uporabnik si izvorno spletno vsebino lahko ogleda v primeru, da se je izvedel glavni tok dogodkov. V nasprotnem primeru si vsebine ne more ogledati in stanje v sistemu ostane nespremenjeno.

6.2.4 Opis toka dogodkov za primer Prijava v sistem

Kratek opis:

Primer uporabe omogoča uporabniku prijavo v sistem. S tem je uporabnik deležen dodatnih funkcionalnosti, katerih pa neprijavljen uporabnik ne more koristiti.

Osnovni tok:

0. Uporabnik začne postopek prijave v sistem na vstopni spletni strani aplikacije, kjer v vnosna polja vnese uporabniško ime in geslo ter vnos potrdi s pritiskom na gumb.
1. Sistem preveri ali je uporabniško ime ter geslo pravilno.
2. Sistem prikaže uporabniški meni aplikacije.

Alternativni tok:

0. Uporabnik začne postopek prijave v sistem na vstopni spletni strani aplikacije, kjer v vnosna polja vnese uporabniško ime in geslo ter vnos potrdi s pritiskom na gumb.
1. Sistem preveri ali je uporabniško ime ter geslo pravilno.
2. Sistem zaradi napačnega uporabniškega imena ali gesla uporabniku zavrne dostop do uporabniškega menija aplikacije.

Predpogoj:

-

Rezultat in spremembe v sistemu:

Uporabniku je dostop do aplikacije omogočen v primeru, da se je izvedel glavni

tok dogodkov. V nasprotnem primeru uporabnik nima dostopa do aplikacije in stanje v sistemu ostane nespremenjeno.

6.2.5 Opis toka dogodkov za primer Ustvarjanje in oblikovanje nove vsebine

Kratek opis:

Primer uporabe omogoča uporabniku ustvarjanje in oblikovanje nove vsebine preko bookmarkleta. Po meri oblikovano spletno vsebino lahko uporabnik nato preko e-pošte posreduje prijateljem ali pa jo shrani za lastno uporabo.

Osnovni tok:

0. Uporabnik začne postopek ustvarjanja in oblikovanja nove spletne vsebine na poljubni spletni vsebini, katero si želi shraniti ter po meri oblikovati, s pritiskom na bookmarklet v spletnem brskalniku.
1. Sistem ponudi uporabniku orodno vrstico s katero lahko oblikuje spletno vsebino.
2. Uporabnik z uporabo orodne vrstice poljubno oblikuje spletno vsebino.
3. Uporabnik shrani spletno vsebino.
4. Sistem shrani vsebino ter prikaže obvestilo o uspešno shranjenih spremembah.

Alternativni tok:

0. Uporabnik začne postopek ustvarjanja in oblikovanja nove spletne vsebine na poljubni spletni vsebini, katero si želi shraniti ter po meri oblikovati, s pritiskom na bookmarklet v spletnem brskalniku.
1. Sistem ponudi uporabniku orodno vrstico s katero lahko oblikuje spletno vsebino.
2. Uporabnik z uporabo orodne vrstice poljubno oblikuje spletno vsebino.
3. Uporabnik shrani spletno vsebino.
4. Sistem ne najde potrebnih podatkov o uporabniku, zato prikaže obvestilo o neuspešno shranjenih spremembah.

Predpogoj:

Uporabnik mora bit prijavljen v sistem ter po navodilih aplikacije si mora

predhodno shraniti bookmarklet v spletni brskalnik.

Rezultat in spremembe v sistemu:

Sistem novo vsebino uspešno shrani v primeru, da se je izvedel glavni tok dogodkov. V nasprotnem primeru ostane stanje v sistemu nespremenjeno.

6.2.6 Opis toka dogodkov za primer Posredovanje vsebine preko e-pošte

Kratek opis:

Primer uporabe omogoča uporabniku posredovanje shranjene vsebine preko e-pošte. Uporabniku je tako omogočeno posredovanje dodatnih informacij, katere pa izvorne spletne vsebine ne vsebujejo.

Osnovni tok:

0. Uporabnik začne postopek posredovanja vsebine preko e-pošte v uporabniškem vmesniku aplikacije s pritiskom na gumb za posredovanje vsebine, ki se nahaja ob vsaki vsebini v seznamu.
1. Sistem mu prikaže vnosno polje v katerega vnese e-poštni naslov prejemnika.
2. Uporabnik vnese e-poštni naslov ter vnos potrdi s pritiskom na gumb.
3. Sistem preveri, ali je vnesen naslov pravilno zapisan ter pošlje e-poštno sporočilo na naveden naslov z vsebino o povezavi do spremenjene spletne vsebine, nato prikaže obvestilo o uspešno poslanem sporočilu.

Alternativni tok:

0. Uporabnik začne postopek posredovanja vsebine preko e-pošte v uporabniškem vmesniku aplikacije s pritiskom na gumb za posredovanje vsebine, ki se nahaja ob vsaki vsebini v seznamu.
1. Sistem mu prikaže vnosno polje v katerega vnese e-poštni naslov prejemnika.
2. Uporabnik vnese e-poštni naslov ter vnos potrdi s pritiskom na gumb.
3. Sistem ob preverjanju pravilnosti zapisa ugotovi nepravilnosti v zapisu ter prikaže obvestilo o neuspešno poslanem sporočilu.

Predpogoj:

Uporabnik mora bit prijavljen v sistem.

Rezultat in spremembe v sistemu:

Sistem posreduje vsebino preko e-pošte v primeru, da se je izvedel glavni tok dogodkov. V nasprotnem primeru ostane stanje v sistemu nespremenjeno.

6.2.7 Opis toka dogodkov za primer Brisanje vsebine

Kratek opis:

Primer uporabe omogoča uporabniku brisanje shranjenih vsebin. Pri organiziranju shranjenih vsebin je uporabnik včasih prisiljen shranjene vsebine tudi izbrisati.

Osnovni tok:

0. Uporabnik začne postopek brisanja vsebine v uporabniškem vmesniku aplikacije s pritiskom na gumb za brisanje vsebine, ki se nahaja ob vsaki vsebini v seznamu.
1. Sistem izbriše vsebino ter jo odstrani iz seznama.

Alternativni tok:

0. Uporabnik začne postopek brisanja vsebine v uporabniškem vmesniku aplikacije s pritiskom na gumb za brisanje vsebine, ki se nahaja ob vsaki vsebini v seznamu.
1. Sistem ne najde potrebnih podatkov o uporabniku, zato zavrne brisanje vsebine.

Predpogoj:

Uporabnik mora bit prijavljen v sistem.

Rezultat in spremembe v sistemu:

Sistem izbriše vsebino ter jo odstrani iz seznama v primeru, da se je izvedel

glavni tok dogodkov. V nasprotnem primeru ostane stanje v sistemu nespremenjeno.

6.2.8 Opis toka dogodkov za primer Razporejanje vsebin po kategorijah

Kratek opis:

Primer uporabe omogoča uporabniku razporejanje vsebin po kategorijah.

Osnovni tok:

0. Uporabnik začne postopek razporejanja vsebin po kategorijah v uporabniškem vmesniku aplikacije s pritiskom na gumb za izbiro kategorije, ki se nahaja ob vsaki vsebini v seznamu.
1. Sistem uporabniku prikaže predhodno oblikovan seznam kategorij.
2. Uporabnik izbere kategorijo, v katero želi dodati vsebino.
3. Sistem doda vsebino v izbrano kategorijo ter v seznamu ob vsebini prikaže kategorijo v katero je bila vsebina dodana.

Alternativni tok:

0. Uporabnik začne postopek razporejanja vsebin po kategorijah v uporabniškem vmesniku aplikacije s pritiskom na gumb za izbiro kategorije, ki se nahaja ob vsaki vsebini v seznamu.
1. Sistem uporabniku prikaže predhodno oblikovan seznam kategorij.
2. Uporabnik izbere kategorijo, v katero želi dodati vsebino.
3. Ker uporabnik izbere kategorijo, v katero je vsebina že dodana, sistem ne upošteva sprememb, zato ostane stanje nespremenjeno.

Predpogoj:

Uporabnik mora bit prijavljen v sistem.

Rezultat in spremembe v sistemu:

Sistem razporedi vsebino v kategorijo v primeru, da se je izvedel glavni tok dogodkov. V nasprotnem primeru ostane stanje v sistemu nespremenjeno.

6.2.9 Opis toka dogodkov za primer Razporejanje vsebin po značkah

Kratek opis:

Primer uporabe omogoča uporabniku razporejanje vsebin po značkah. Tako razporejanje je uporabno kadar želi uporabnik določeni vsebini dodeliti več značk, česar pa pri razporejanju po kategorijah ne more storiti.

Osnovni tok:

0. Uporabnik začne postopek razporejanja vsebin po značkah v uporabniškem vmesniku aplikacije s pritiskom na gumb za izbiro značke, ki se nahaja ob vsaki vsebini v seznamu.
1. Sistem uporabniku prikaže predhodno oblikovan seznam značk.
2. Uporabnik izbere značko, katero želi dodati vsebini.
3. Sistem vsebini doda izbrano značko ter v seznamu ob vsebini prikaže značko, katera je bila dodana vsebini.

Alternativni tok:

0. Uporabnik začne postopek razporejanja vsebin po značkah v uporabniškem vmesniku aplikacije s pritiskom na gumb za izbiro značke, ki se nahaja ob vsaki vsebini v seznamu.
1. Sistem uporabniku prikaže predhodno oblikovan seznam značk.
2. Uporabnik izbere značko, katero želi dodati vsebini.
3. Ker uporabnik izbere značko, katera je vsebini že dodeljena, sistem ne upošteva sprememb, zato ostane stanje nespremenjeno.

Predpogoj:

Uporabnik mora bit prijavljen v sistem.

Rezultat in spremembe v sistemu:

Sistem vsebini doda značko v primeru, da se je izvedel glavni tok dogodkov. V nasprotnem primeru ostane stanje v sistemu nespremenjeno.

6.2.10 Opis toka dogodkov za primer Dodeljevanje dostopa

Kratek opis:

Primer uporabe omogoča uporabniku dodeljevanje stopnje dostopa do vsebin. Kadar želi uporabnik določeno vsebino deliti s prijatelji, mora vsebini dodeliti *javen* dostop. V primeru, da želi vsebino hraniti za lastno uporabo pa jo lahko pred drugimi uporabniki zaščiti tako, da dodeli vsebini *zaseben* dostop.

Osnovni tok:

0. Uporabnik začne postopek dodeljevanja stopnje dostopa do vsebin v uporabniškem vmesniku aplikacije s pritiskom na gumb za izbiro stopnje dostopa, ki se nahaja ob vsaki vsebini v seznamu.
1. Sistem uporabniku prikaže seznam z izbiro stopnje dostopa. Izbira lahko med javnim in zasebnim dostopom.
2. Uporabnik izbere stopnjo dostopa, katero želi dodeliti vsebini.
3. Sistem vsebini dodeli stopnjo dostopa ter v seznamu ob vsebini prikaže stopnjo dostopa, katera je bila dodeljena vsebini.

Alternativni tok:

0. Uporabnik začne postopek dodeljevanja stopnje dostopa do vsebin v uporabniškem vmesniku aplikacije s pritiskom na gumb za izbiro stopnje dostopa, ki se nahaja ob vsaki vsebini v seznamu.
1. Sistem uporabniku prikaže seznam z izbiro stopnje dostopa. Izbira lahko med javnim in zasebnim dostopom.
2. Uporabnik izbere stopnjo dostopa, katero želi dodeliti vsebini.
3. Ker uporabnik izbere stopnjo dostopa, katera je v tistem trenutku vsebini že dodeljena, sistem ne upošteva sprememb, zato ostane stanje nespremenjeno.

Predpogoj:

Uporabnik mora bit prijavljen v sistem.

Rezultat in spremembe v sistemu:

Sistem vsebini dodeli oz. spremeni stopnjo dostopa v primeru, da se je iz-

vedel glavni tok dogodkov. V nasprotnem primeru ostane stanje v sistemu nespremenjeno.

6.2.11 Opis toka dogodkov za primer Odjava iz sistema

Kratek opis:

Primer uporabe omogoča uporabniku odjavo iz sistema.

Osnovni tok:

0. Uporabnik začne postopek odjave iz sistema v uporabniškem vmesniku aplikacije s pritiskom na gumb za odjavo iz sistema, ki se nahaja v desnem meniju zgoraj.
1. Sistem uporabnika odjavi iz sistema ter ga preusmeri na vstopno spletno stran aplikacije.

Alternativni tok:

0. Uporabnik začne postopek odjave iz sistema v uporabniškem vmesniku aplikacije s pritiskom na gumb za odjavo iz sistema, ki se nahaja v desnem meniju zgoraj.
1. Sistem ne najde potrebnih podatkov o uporabniku, zato prikaže obvestilo o neuspešni odjavi.

Predpogoj:

Uporabnik mora bit prijavljen v sistem.

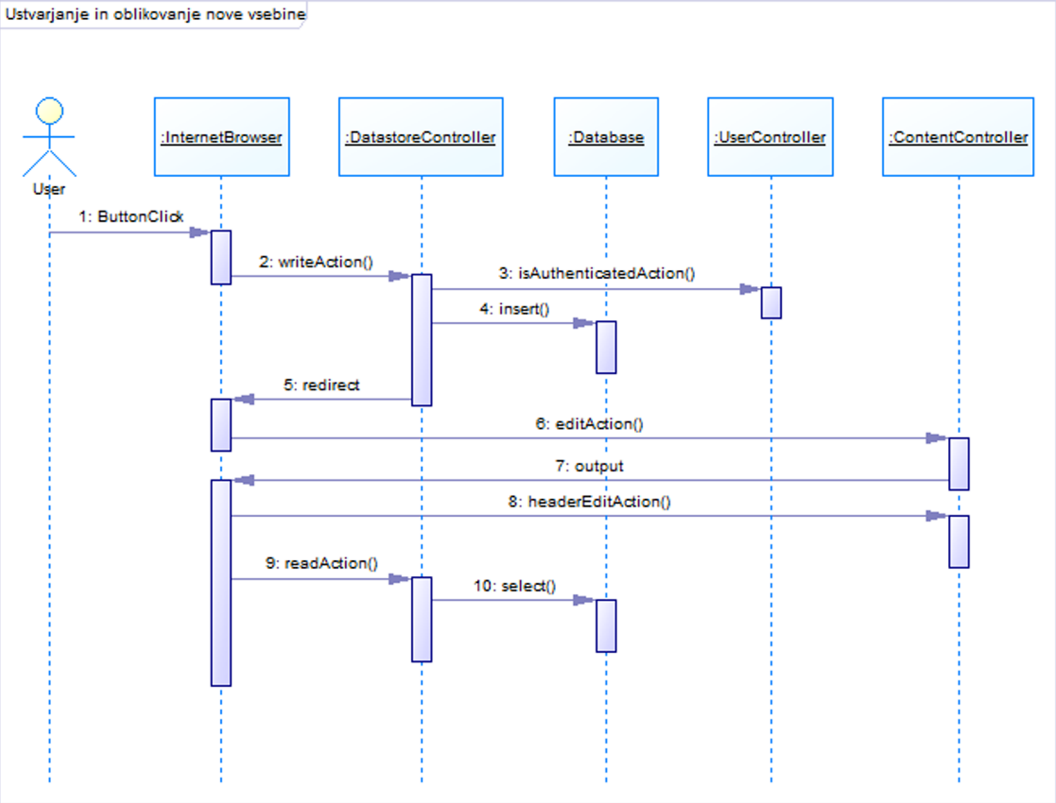
Rezultat in spremembe v sistemu:

Sistem uporabnika odjavi iz sistema v primeru, da se je izvedel glavni tok dogodkov. V nasprotnem primeru ostane stanje v sistemu nespremenjeno.

6.3 Diagrami zaporedja

Diagrami zaporedja prikazujejo interakcije/sodelovanje na osnovi časovnega zaporedja. Prikazujejo, kako objekti sodelujejo in izmenjujejo sporočila. Diagrami ne prikazujejo povezav med objekti. [30]

6.3.1 Diagram zaporedja za primer uporabe Ustvarjanje in oblikovanje nove vsebine



Slika 6.3: Diagram zaporedja za primer uporabe Ustvarjanje in oblikovanje nove vsebine.

Diagram zaporedja (slika 6.3) prikazuje časoven potek dogodkov, ki so potrebni za ustvarjanje in oblikovanje nove vsebine.

Najprej uporabnik *(User)* pritisne na gumb v spletnem brskalniku. Brskalnik *(InternetBrowser)* s pomočjo JavaScript kode vzpostavi povezavo s strežnikom, kamor pošlje podatke o spletni strani, na kateri se uporabnik trenutno nahaja. Strežnik ob prejetju podatkov le-te interpretira in pokliče ustrezno metodo *(writeAction)* ki je zadolžena za vpis vsebine v podatkovno bazo. Preden pa se vpis izvrši, aplikacija preveri prijavne podatke uporabnika oz. aktivnost njegove prijavne seje *(isAuthenticatedAction)*. Po uspešni avtentifikaciji se nato izvede tudi fizičen vpis podatkov v podatkovno bazo *(insert)* ter preusmeritev na nov URL naslov *(redirect)*, kateri se ustvari na podlagi

podatkov o uporabniku ter zapisa vsebine v podatkovni bazi. Nato se izvede metoda *editAction*, ki poskrbi za pravilno izgradnjo spletne strani in omogoča urejanje vsebine. Poleg vsebine izvorne spletne strani, se v glavi strani doda tudi orodna vrstica za urejanje ter povezava do izvorne spletne strani. Za branje ter posredovanje vsebine iz podatkovne baze skrbi metoda *readAction*. Metoda *select* deluje na strani podatkovne baze in skrbi za zajem vsebine iz podatkovne baze za uporabo v poslovni logiki.

6.4 Opis spletne aplikacije

V naslednjih poglavjih bomo predstavili razvito aplikacijo, kjer bomo spoznali njen namen, funkcionalnosti ter uporabniški vmesnik.

6.4.1 Splošno o aplikaciji

Aplikacija je namenjena vsem, ki želijo ob posredovanju povezave do neke spletne vsebine drugim uporabnikom pokazati, razložiti itd. še kaj dodatnega poleg posredovane povezave. Za uporabo vseh funkcionalnosti, ki jih aplikacija ponuja, se je potrebno prijaviti v sistem, predhodno pa registrirati. Aplikacijo pa lahko uporablja vsak, ki ima dostop do interneta.

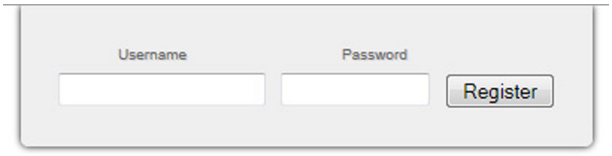
Ko nekomu posredujemo povezavo do neke spletne vsebine mu s tem posredujemo le del informacije, s čimer pa lahko povzročimo zmedo pri interpretaciji vsebine. Naša aplikacija rešuje ta problem tako, da lahko spletno vsebino, ki jo želimo nekomu posredovati, obogatimo z raznimi oznakami, dodamo ali pa odstranimo nepomembno vsebino ipd. S tem prejemniku vsebine skrajšamo čas, ki bi ga porabil ob prebiranju nespremenjene vsebine ter zmanjšamo zmedo pri interpretaciji.

Poleg spremenjene vsebine, ki jo je dopolnil pošiljatelj, pa prejme prejemnik tudi povezavo do izvorne, nespremenjene spletne vsebine, kar mu omogoča pregled izvorne vsebine brez kakršnihkoli sprememb s strani pošiljatelja. Aplikacija za sporočanje uporablja e-poštni sistem.

6.4.2 Uporabniški vmesnik

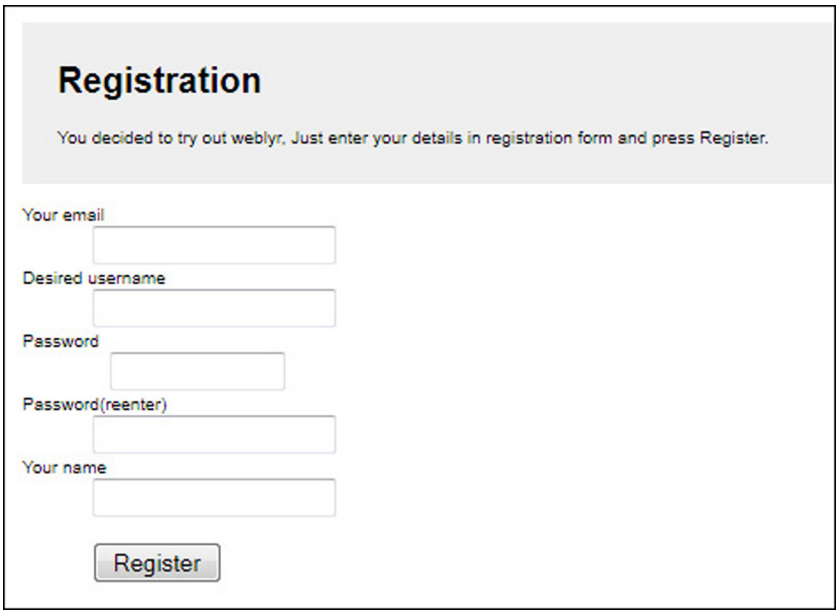
Uporabniku se na vstopni strani poleg kratke slikovne predstavitve o uporabi aplikacije prikaže tudi obrazec za prijavo v aplikacijo (slika 6.4). Obrazec se nahaja v glavi spletne strani. Uporabnik vnese podatke v vnosna polja ter se

s pritiskom na gumb tik ob vnosih poljih prijavi v aplikacijo.



Slika 6.4: Obrazec za prijavo v aplikacijo.

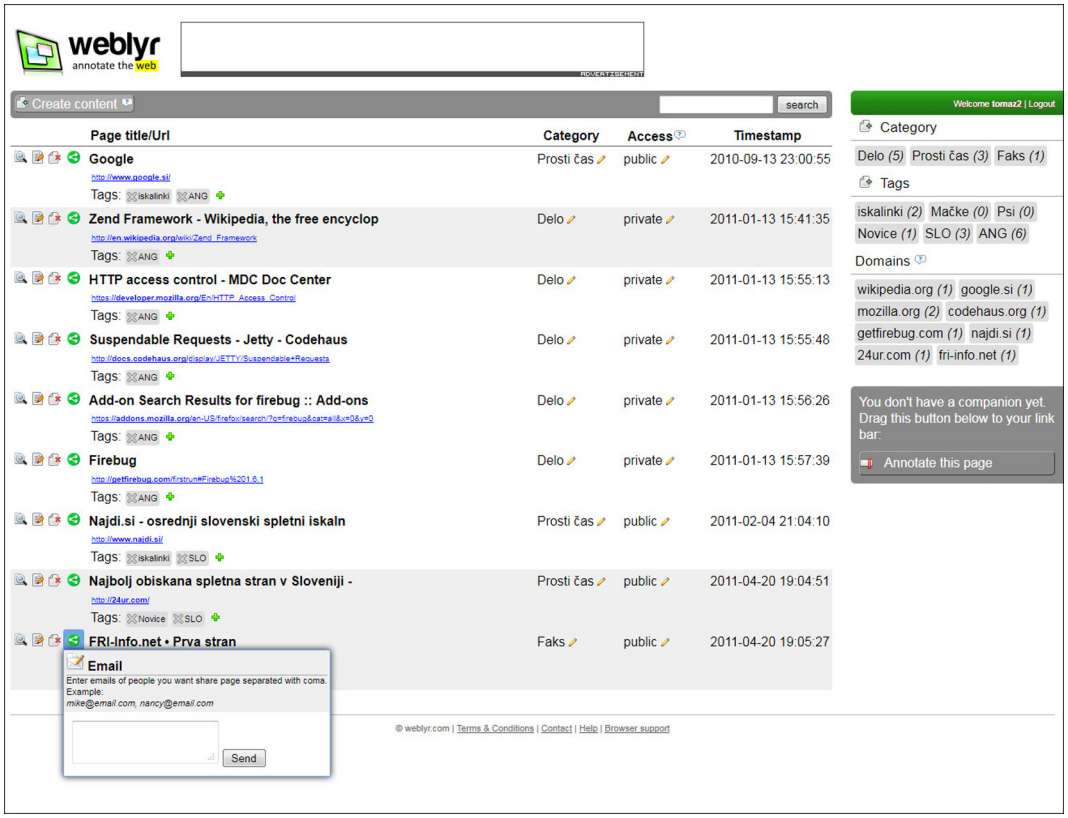
V kolikor uporabnik še ni registriran tudi ni deležen vseh funkcionalnosti, ki jih nudi aplikacija. Registracijo lahko opravi preko povezave, ki se nahaja v nogi spletne strani. Ob pritisku na povezavo se mu prikaže obrazec za vnos osebnih podatkov, ki so potrebni za registracijo (slika 6.5).



Slika 6.5: Obrazec za registracijo.

Po uspešni prijavi v aplikacijo, se uporabniku prikaže uporabniški meni aplikacije (slika 6.6).

Uporabniški meni vsebuje seznam shranjenih vsebin. Posamezna shranjena vsebina v seznamu zavzema eno vrstico (slika 6.7) in vsebuje naslednje podatke: od leve proti desni si najprej sledijo štirje gumbi (to so gumbi za ogled,



Slika 6.6: Uporabniški meni aplikacije.

urejanje, brisanje ter posredovanje vsebine). Nato si eden pod drugim sledijo naslov, izvorni spletni naslov ter morebitne oznake, ki so dodeljene shranjeni vsebini. Sledi ime kategorije v katero spada vsebina ter vrsta dostopa do vsebine. Čisto na koncu pa je podatek o času, ko je bila vsebina shranjena.

Slika 6.7: Primer shranjene vsebine.

Na desni strani se nahaja meni (slika 6.8), kjer lahko uporabnik upravlja s kategorijami ter oznakami. Te lahko dodaja ter si ogleda katere vsebine pripadajo kategoriji. Aplikacija samodejno razvrsti vsebine glede na domeno na kateri se nahaja izvorna spletna vsebina. Uporabnik si tako lahko ogleda

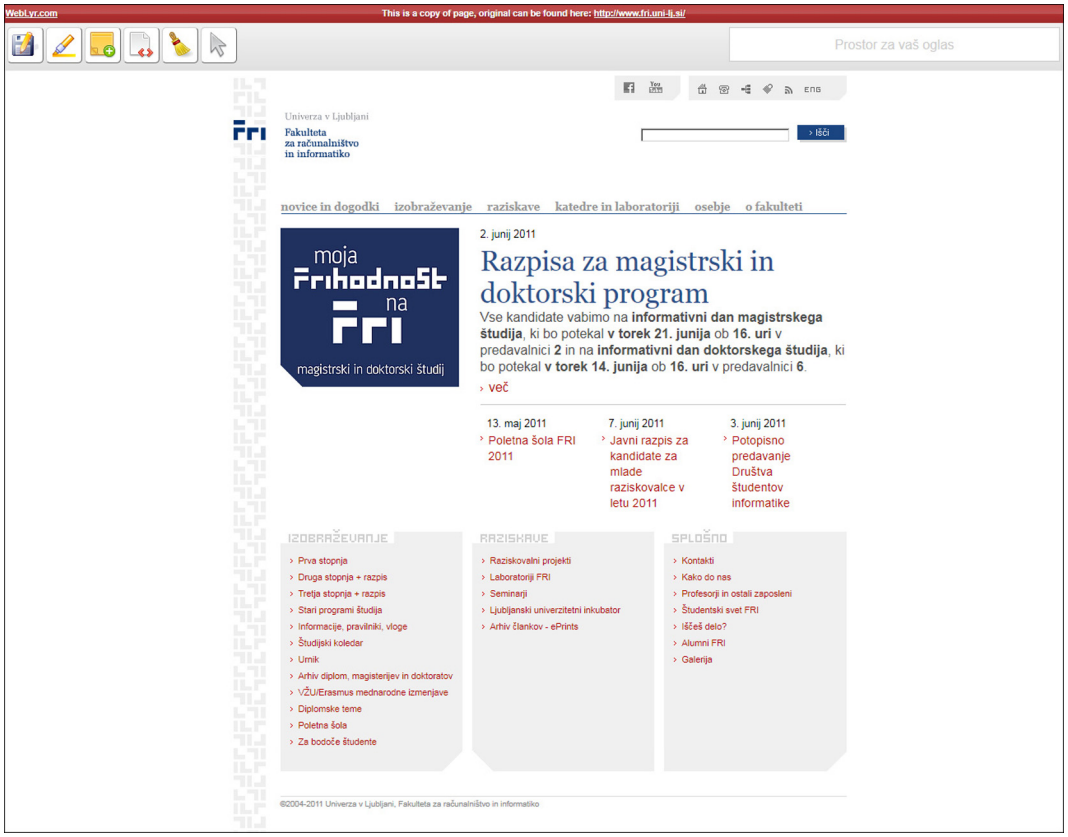
vsebine razvrščene po domenah. Čisto na vrhu menija se nahajajo podatki o uporabniku ter gumb za odjavo trenutno prijavljenega uporabnika. Čisto na dnu pa se nahaja povezava do bookmarkleta, katero uporabnik shrani v spletnem brskalniku.

Slika 6.8: Meni v uporabniškem vmesniku.

Na vrhu seznama vsebin se nahaja orodna vrstica katera vsebuje gumb za kreiranje nove, prazne vsebine ter vnosno polje ter gumb za iskanje že shranjenih vsebin. S pritiskom na gumb za kreiranje nove vsebine aplikacija uporabnika preusmeri na prazno vsebino, katero lahko s pomočjo orodne vrstice tudi poljubno ustvarja ter oblikuje. Uporabnik lahko tudi išče med vsebinami, kar pa stori z vpisom iskanega niza v vnosno polje čisto na desni strani orodne vrstice ter pritiskom na gumb tik ob vnosnem polju.

Orodna vrstica za oblikovanje vsebin (slika 6.9, 6.10) je uporabniku na voljo bodisi preko bookmarkleta, bodisi ob pritisku na gumb za kreiranje nove, prazne vsebine ali pa s pritiskom na gumb za urejanje že shranjene vsebine.

Slika 6.9: Orodna vrstica za oblikovanje vsebine - izrez.



Slika 6.10: Spletna stran z orodno vrstico za oblikovanje vsebine.

Orodna vrstica omogoča uporabniku shranjevanje sprememb opravljenih na vsebini, označevanje besedila (angl. Highlight), dodajanje opomb (angl. Sticky notes), vstavljanje objektov s pomočjo vrivanja programske kode (angl. Embed), brisanje ter označevanje objektov. Orodna vrstica pa nam omogoča tudi obisk izvorne, nespremenjene spletne strani preko URL naslova, ki je prikazan na vrhu orodne vrstice.

Poglavje 7

Sklepne ugotovitve

V diplomskem nalogi smo v prvem delu, ki je nekoliko bolj teoretičen, opisali večji del orodij, pristopov in tehnologij, ki smo jih uporabili pri realizaciji naše spletne aplikacije. V drugem delu diplomske naloge smo predstavili praktično uporabo pred tem opisanih orodij, pristopov in tehnologij na primeru izgradnje spletne aplikacije. Prav uporaba vseh opisanih orodij, pristopov in tehnologij pa je botrovala k razmeroma kratkem času razvoja aplikacije.

Menimo, da nam je v nalogi uspelo realizirati zastavljeni cilj, ki smo si ga zadali na začetku naloge, da naj spletna aplikacija uporabnikom omogoča medsebojno sodelovanje ter s pomočjo uporabe spletnih vsebin izmenjavo idej, obvestil ter podobnih informacij. Pri tem je bilo pridobljenega veliko znanja in izkušenj. Med razvojem spletne aplikacije nismo imeli posebnih težav.

Žal je danes na svetovnem spletu že kar nekaj sorodnih spletnih servisov, ki ponujajo vse, kar ponuja naša aplikacija, nekateri pa tudi več. V fazi načrtovanja teh servisov še ni bilo oz. jih nismo poznali, zato smo implementirali svojo rešitev. Prednost naše aplikacije je predvsem ta, da je nastala namensko, zato jo lahko nadgradimo in spreminjamo po lastnih željah.

Zaradi pomanjkanja časa spletna aplikacija še ni bila v celoti testirana ter vpeljana v delovno okolje. Med samim razvojem smo prišli do določenih idej za izboljšavo aplikacije, ki jih v nadaljevanju tudi podajamo:

- potrditev registracije preko elektronske pošte,
- omogočeno nalaganje datotek,
- urejanje iste vsebine s strani različnih uporabnikov.

Prizadevali si bomo, da bomo v prihodnje aplikacijo v celoti tudi stestirali

ter vpeljali v delovno okolje, omenjene ideje pa realizirali ter aplikacijo po potrebi tudi vzdrževali.

Slike

Literatura

[1] (2010) Web 2.0. Dostopno na:
http://en.wikipedia.org/wiki/Web_2.0

[2] (2010) Tim O'Reilly, What is Web 2.0? Dostopno na:
http://www.seomoz.org/web2.0/zeitgeist

[3] (2010) Splet. Dostopno na:
http://sl.wikipedia.org/wiki/Splet

[4] (2010) Web browser. Dostopno na:
http://en.wikipedia.org/wiki/Web_browser

[5] (2010) World Wide Web. Dostopno na:
http://en.wikipedia.org/wiki/World_Wide_Web

[6] Mosaic (web browser). Dostopno na:
http://en.wikipedia.org/wiki/Mosaic_%28web_browser%29

[7] (2010) Razvoj spletnih aplikacij z uporabo programskega ogrodja JCorporate Expresso. Dostopno na:
http://www.kiblix.org/2003/work_jsp.html

[8] (2010) Web application. Dostopno na:
http://en.wikipedia.org/wiki/Web_application

[9] (2010) Ajax (programming). Dostopno na:
http://en.wikipedia.org/wiki/Ajax_%28programming%29

[10] (2010) Php. Dostopno na:
http://sl.wikipedia.org/wiki/Php

[11] Cal Evans, *Guide to Programming with ZEND FRAMEWORK*, Marco Tabini & Associates, Toronto, 2008.

[12] Kevin McArthur, *Pro PHP Patterns, Frameworks, Testing and More, Apress*, New York, 2008, str. 201-270.

[13] (2010) Java BluePrints, Model-View-Controller. Dostopno na: http://java.sun.com/blueprints/patterns/MVC-detailed.html

[14] (2010) Ajax (programiranje). Dostopno na: http://sl.wikipedia.org/wiki/AJAX_%28programiranje%29

[15] Jesse James Garrett, *Ajax: A New Approach to Web Applications*. Adaptive Path LLC, februar 2005.

[16] (2010) HTML. Dostopno na: http://sl.wikipedia.org/wiki/HTML

[17] (2010) XHTML. Dostopno na: http://sl.wikipedia.org/wiki/XHTML

[18] (2010) CSS. Dostopno na: http://sl.wikipedia.org/wiki/CSS

[19] (2010) Document Object Model (DOM). Dostopno na: http://www.w3.org/DOM/

[20] (2010) Document Object Model. Dostopno na: http://en.wikipedia.org/wiki/Document_Object_Model

[21] (2010) XML. Dostopno na: http://sl.wikipedia.org/wiki/XML

[22] (2010) XSLT. Dostopno na: http://office.microsoft.com/sl-si/infopath-help/xslt-HP001096733.aspx

[23] (2010) XSLT. Dostopno na: http://en.wikipedia.org/wiki/XSLT

[24] (2011) XMLHttpRequest. Dostopno na: http://en.wikipedia.org/wiki/XMLHttpRequest

[25] Kris Hadlock, *AJAX for Web Application Developers*, Sams, november 2006

[26] (2011) JavaScript. Dostopno na: http://sl.wikipedia.org/wiki/JavaScript

[27] (2011) jQuery. Dostopno na:
http://en.wikipedia.org/wiki/JQuery

[28] (2011) Bookmarklet. Dostopno na:
http://en.wikipedia.org/wiki/Bookmarklet

[29] (2011) MYSQL. Dostopno na:
http://sers.s-sers.mb.edus.si/gradiva/w3/omrezja/70_strezniki/03_mysql.html

[30] (2011) Diagrami interakcije. Dostopno na:
http://cot.uni-mb.si/cotl/zima2003/diagrami_interakcije.html