UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Dean Podgornik

Uporaba konceptov spleta druge generacije pri izgradnji spletnih aplikacij

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: viš. pred. dr. Damjan Vavpotič

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: 00063/2011
Datum: 01.02.2011

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

Kandidat: **DEAN PODGORNIK**

Naslov: **UPORABA KONCEPTOV SPLETA DRUGE GENERACIJE PRI IZGRADNJI SPLETNIH APLIKACIJ**

USE OF WEB 2.0 CONCEPTS IN DEVELOPMENT OF WEB APPLICATIONS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Diplomsko delo naj predstavi pojem tako imenovanega spleta druge generacije (Web 2.0). V okviru teoretičnega dela predstavite ključne tehnologije, koncepte in arhitekture povezane s tem pojmom ter kako jih uporabiti za potrebe izgradnje spletnih aplikacij. Še posebej se posvetite pojmom kot so storitveno usmerjena arhitektura, bogate spletne aplikacije in socialni splet. Predstavite tudi ključne razlike med tako imenovanim spletom prve generacije in spletom druge generacije. V okviru praktičnega dela diplomske naloge demonstrirajte uporabo izbranih konceptov spleta druge generacije na primeru manjše spletne aplikacije.

Mentor:

viš. pred. dr. Damjan Vavpotič

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a Dean Podgornik,

z vpisno številko 63070365,

sem avtor/-ica diplomskega dela z naslovom:

Uporaba konceptov spleta druge generacije pri izgradnji spletnih aplikacij

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)

  viš. pred. dr. Damjan Vavpotič

- 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 _____ Podpis avtorja/-ice: _____

ZAHVALA

Za mentorstvo in uporabne nasvete pri izdelavi diplomske naloge se zahvaljujem svojemu mentorju viš. pred. dr. Damjanu Vavpotiču.

Prav tako se zahvaljujem tudi podjetju Editor d.o.o. za posredovanje znanja o spletnih tehnologijah ter staršem in mojemu dekletu, ki so me pri študiju spodbujali in podpirali.

KAZALO VSEBINE

SEZNAM KRATIC IN SIMBOLOV

AJAX (*angl. Asynchronous JavaScript and XML*) asinhroni JavaScript in XML

API (*angl. Application Programming Interface*) programski vmesnik

ASP (*angl. Active Server Pages*) strežniška tehnologija podjetja Microsoft

CGI (*angl. Common Gateway Interface*) skupek pravil, ki določajo način komunikacije strežnika z uporabniško programsko opremo

CMS (*angl. Content Management System*) sistem za upravljanje vsebine

CRM (*angl. Customer Relationship Management*) sistem za upravljanje odnosov s strankami

DOM (*angl. Document Object Model*) programski vmesnik za XML ter HTML dokumente, ki omogoča definicijo logične strukture dokumenta ter interakcijo z njegovimi elementi

ERP (*angl. Enterprise Resource Planning*) integrirani poslovni informacijski sistem

GPL (*angl. GNU General Public License*) najpogosteje uporabljena licenca za prosto programje

JSP (*angl. JavaServer Pages*) strežniška tehnologija, bazirana na programskem jeziku Java

KMS (*angl. Knowledge Management System*) sistem za upravljanje znanja

PHP (*angl. PHP: Hypertext Preprocessor*) trenutno najbolj priljubljen odprtokodni skriptni programski jezik

PRM (*angl. Partner Relationship Management*) sistem za upravljanje odnosov s partnerji

Q&A (*angl. Questions and Answers*) izraz za spletna mesta, kjer uporabniki odgovarjajo na postavljena vprašanja

RDF (*angl. Resource Description Framework*) standardni model za izmenjavo podatkov v spletu

REST (*angl. Representational State Transfer*) stil računalniške arhitekture, ki predstavlja enostavnejšo alternativo pristopu SOAP

RIA (*angl. Rich Internet Application*) obogatena spletna aplikacija

RSS (*angl. Really Simple Syndication*) protokol za objavo in distribucijo spletnih vsebin v zapisu XML

SCM (*angl. Supply Chain Management*) sistem za upravljanje oskrbovalne verige

SOA (*angl. Service-oriented architecture*) storitveno usmerjena arhitektura

SOAP (*angl. Simple Object Access Protocol*) standard za izmenjavo strukturiranih informacij v spletnih storitvah

SQL (*angl. Structured Query Language*) strukturiran povpraševalni jezik za delo s podatkovnimi bazami

SSB (*angl. Site-specific browser*) orodje, ki omogoča prikaz spletne strani kot namizne aplikacije

SSL (*angl. secure socket layer*) protokol, ki zagotavlja varno povezavo med strežnikom in odjemalcem

UDDI (*angl. Universal Description Discovery and Integration*) standard, ki temelji na označevalnem jeziku XML za vnos in iskanje spletnih storitev

URL (*angl. Uniform Resource Locator*) internetni naslov, preko katerega lahko dostopamo do želene vsebine

W3C (*angl. World Wide Web Consortium*) mednarodna organizacija za spletne standarde

WSDL (*angl. Web Services Description Language*) XML format za opis spletnih storitev

XML (*angl. Extensible Markup Language*) format podatkov za izmenjavo strukturiranih dokumentov v spletu

POVZETEK

Splet je od njegovih prvih začetkov oziroma prve delujoče spletne strani, zgrajene leta 1990, prehodil že dolgo pot. S stalnim razvojem, z uporabo novih konceptov in sodobnih tehnologij se je pritihotapil v sleherni del našega vsakdana. Torej, splet ni več le statičen vir informacij, ampak je postal orodje za podporo tako poslovnim procesom kot vsakdanjim opravilom. Ta preobrat se je izpostavil že leta 2004, posledica česar je bila uveljavitev pojma Web 2.0 oziroma spleta druge generacije, kateremu se danes pripisujejo sledeče lastnosti: dinamičnost, interaktivnost, interoperabilnost, povezljivost informacij, uporabniško generirana vsebina ter zmožnost sodelovanja uporabnikov.

V tej diplomski nalogi sem predstavil splet v njegovi drugi evolucijski fazi ter uporabo njegovih konceptov pri izgradnji spletnih aplikacij. V teoretičnem delu sem natančneje predstavil pomen spleta druge generacije, okoliščine njegove pojavitve, kritike ter njegove tri ključne elemente, in sicer RIA, SOA ter socialni splet (*angl. social web*). Poudarek pa sem namenil tudi značilnim funkcionalnostim spletnih mest (*angl. websites*) spleta druge generacije, uveljavljenim predstavnikom tega novega spleta, ključnim razlikam z njegovim predhodnikom, ključnim spletnim tehnologijam ter potencialni prihodnosti spleta. Za utrditev teoretičnega dela diplomske naloge sem glavne koncepte oziroma značilnosti spleta druge generacije združil v izgradnjo manjše spletne aplikacije. Tako sem namreč z uporabo sodobne spletne tehnologije in omenjenih konceptov oziroma značilnosti spleta druge generacije predstavil, kako na enostaven in hiter način zgraditi manjšo spletno aplikacijo, ki ji upravičeno lahko rečemo spletna aplikacija spleta druge generacije.

Ključne besede: Web 2.0, RIA, SOA, socialni splet, spletna aplikacija

ABSTRACT

The web has travelled a long way from its first website created in 1990. With its continuous revolutionary development, new used concepts and modern technologies, it became an important part of our everyday life. The web is not just a static database with information anymore, but a tool for many business processes and ordinary daily tasks. This shift was already exposed in 2004, and from then the web has acquired a new term, exactly Web 2.0, which relates to the second generation of the web and has the following properties: dynamics, interactivity, interoperability, information connectivity, user-generated content and the ability for providing user participation.

In this diploma thesis I presented the web in its second evolution phase and the use of its concepts at development of web applications. In the theoretic part I presented the meaning of the Web 2.0, circumstances of its appearance, critiques and its three key elements (RIA, SOA and social web). I also focused on the typical functionality of modern websites, representatives of Web 2.0 websites, key differences between Web 2.0 and its predecessor, key web technologies and on the potential future of the web. With the intention to support the theoretical part, I merged most of the presented Web 2.0 features and characteristic into development of a smaller web application. With this developed Web 2.0 application I demonstrated that a web application can be built in an easy and quick way by using modern web technologies and Web 2.0 concepts.

Keywords: Web 2.0, RIA, SOA, social web, web application

1 UVOD

Web 2.0 ali v slovenskem prevodu splet 2.0 je pojem, ki se nanaša na drugo generacijo svetovnega spleta. Čeprav sam izraz nakazuje na drastične spremembe v tehnoloških specifikacijah, te niso bistvo spleta druge generacije. Splet si je novo različico namreč prislužil na podlagi novega koncepta razvoja in novega načina uporabe storitev spleta. Lastnosti novega spleta so tako dinamičnost, interaktivnost, interoperabilnost, povezljivost informacij, uporabniško generirana vsebina ter zmožnost sodelovanja uporabnikov. Ker pa je večina Web 2.0 funkcionalnosti ponujenih kot brezplačna storitev, je ena izmed njegovih ključnih značilnosti tudi ta, da se hitro širi med končnimi uporabniki in se razvija z osupljivo hitrostjo.

Za ta pojem obstaja več definicij, nobena izmed njih pa ga ne določa natančno, oziroma ne postavi ostre meje med njim in njegovim predhodnikom. Med to množico definicij velja izpostaviti Tim O'Railly-evo, ki se glasi:

»Web 2.0 je poslovna revolucija v računalniški industriji, katere povod je pojav interneta kot platforme in poskusa izrabe pravil uspeha le-te. Glavno pravilo uspeha te nove platforme pa je grajenje aplikacij, ki izkoriščajo omrežne oziroma spletne vplive in se z rastočo količino uporabnikov sorazmerno izboljšujejo.«

Omenjeni izraz običajno asociira tudi na načine izvedbe novodobnih spletnih mest, kot so na primer: blogi, wiki-ji, socialni zaznamki, socialna omrežja (*angl. social networks*), strani za delitev multimedijskih vsebin ter spletne aplikacije. Pogosto pa izraz asociira tudi na sodobne spletne tehnologije: JavaScript, AJAX, JavaScript/AJAX ogrodja (*angl. JavaScript/AJAX frameworks*), SOAP ter REST.

Izraz je postal precej prepoznaven tudi med laiki, kar je privedlo do snovanja novih pojmov, povezanih s spletom druge generacije. Razne dejavnosti, ki so v svoje delovanje vpeljale nove koncepte, bazirane na spletnih tehnologijah, so v svojem nazivu pridobile končnico 2.0. Tako so se pojavili pojmi: Library 2.0, Government 2.0, Enterprise 2.0, Classroom 2.0, Travel 2.0, itd.

2 PREDSTAVITEV SPLETA DRUGE GENERACIJE

2.1 Zgodovina Web 2.0

Izraz Web 2.0 naj bi skovali na sestanku oziroma na soočanju idej med podjetji O'Reilly in Medialive International. Na njem so predstavniki obeh podjetij izpostavili dejstvo, da se splet razvija z izjemno hitrostjo, z uporabo novih konceptov ter tehnologij. Ugotovili so, da je v razvoju svetovnega spleta prišlo do prelomnice, ki ga je razdelila na stari (Web 1.0) in novi splet (Web 2.0). Iznajdbo pojma Web 2.0 se tako pripisuje Timu O'Reillyju, direktorju podjetja O'Reilly, čeprav obstajajo tudi določeni viri, ki nakazujejo na uporabo tega pojma že pred omenjenim sestankom.

Web 2.0 se je prvič predstavil širši javnosti leta 2004 na konferenci O'Reilly Media Web 2.0, ki sta jo organizirali podjetji O'Reilly in MediaLive International. Uvodna govorca John Battelle in Tim O'Reilly sta izpostavila splet kot novo platformo, kjer se lahko razvijajo ravno tako kompleksne aplikacije, kot so običajne, namizne aplikacije. Izpostavila sta tudi prednosti teh aplikacij oziroma predstavila, kako se lahko uporabniško generirana vsebina uporabi v prid svojega posla. Opredelila pa sta še razlike med Web 1.0 in Web 2.0 in to na nasprotujočih primerih, kot so Google in Netscape ter Encyclopædia Britannica Online in Wikipedia. Konferenca je dvignila veliko prahu in tako tudi poskrbela, da se je izraz Web 2.0 na področju informacijske tehnologije precej uveljavil. Velik uspeh je prispeval k temu, da se je ta konferenca vsako leto do sedaj tudi ponovila.

Pojem Web 2.0 se je po letu 2004 začel množično širiti tudi med laiki. To dokazujejo številne objave tako tehničnih kot tudi poljudnoznanstvenih revij ter blogov. Najvišjo točko popularnosti med laiki pa je vsekakor dosegel s člankom »Person of the Year: You«, ki je bil objavljen leta 2006 v svetovno znani reviji Time. Članek govori o neverjetno veliki skupnosti končnih uporabnikov, ki s skupnimi močmi ustvarjajo vsebino na spletnim mestih, kot so: Wikipedia, YouTube in MySpace. Bralcem obrazloži, kaj Web 2.0 je in kaj jim prinaša.

Priljubljenost pojma je še naprej rasla in tako je leta 2009 Web 2.0 našel svoje mesto tudi v slovarju. Priznala ga je organizacija, ki skrbi za slovar Global Language Monitor. Omenjena organizacija pridobiva nove zapise na podlagi števila pojavitev besed oziroma besednih zvez na celotnem svetovnem spletu. Besedna zveza Web 2.0 je namreč presegla magično mejo 25.000 pojavitev in se tako uvrstila v omenjeni slovar. Ostali klasični slovarji pa besedne zveze žal še ne priznavajo, saj odgovorne organizacije menijo, da je za uveljavitev nove besedne zveze potrebno daljše časovno obdobje.

2.2 Ključni elementi spleta druge generacije

Celoten koncept spleta druge generacije je mogoče opredeliti zgolj s tremi ključnimi elementi. Ti elementi so RIA, SOA in socialni splet. Vsak omenjen element posebej ne toliko bistveno doprinese, kot pa kombinacija vseh treh elementov skupaj. Šele v združitvi teh elementov se pokaže prava moč spleta druge generacije oziroma spletnih aplikacij, ki ustrezajo konceptom spleta druge generacije. V nadaljevanju tega poglavja je za vsak element posebej predstavljen njegov pomen ter njegove bistvene značilnosti.

2.2.1 RIA

RIA oziroma v slovenskem prevodu obogatena spletna aplikacija je izraz za tiste spletne aplikacije, ki imajo vse značilnosti in funkcionalnosti klasičnih namiznih aplikacij. Za ta izraz so se nekaj časa nazaj uporabljale tudi sopomenke Remote Scripting, X Internet, Rich Web Clients in Rich Web Application, ki pa se danes opuščajo. Za to vrsto aplikacij je značilna tako imenovana bogata uporabniška izkušnja oziroma visoka interaktivnost. Končni uporabnik ne opazi več klasičnega osveževanja strani ter komunikacije med strežnikom in odjemalcem, ampak uporabo aplikacije doživlja kot običajno, dinamično aplikacijo, ki se izvaja v nameščenem operacijskem sistemu.

RIA aplikacije se od običajnih, namiznih aplikacij razlikujejo po tem, da zanje ni potrebna namestitev, da so dostopne iz katerekoli napredne naprave, priključene na splet, ter da se lahko izvajajo v varnem okolju, imenovanem sandbox[1]. Vizualno pa je vidna le razlika, da se RIA aplikacije izvajajo v spletnem brskalniku, namizne aplikacije pa kar neposredno v operacijskem sistemu. To razliko pa je moč tudi odpraviti s pomočjo orodij SSB. Omenjena orodja ustvarijo bližnjico na namizju uporabnika, ki želeno spletno aplikacijo odpre v novem oknu, brez odvečnih menijev ali pa orodnih vrstic spletnega brskalnika. Na tak način so vizualne razlike med RIA aplikacijami in namiznimi aplikacijami skoraj povsem odpravljene.

Od običajnih spletnih aplikacij pa se RIA aplikacije razlikujejo po načinu uporabe strežniške (*angl. server-side technology*) in odjemalčeve tehnologije (*angl. client-side technology*). Pri RIA aplikacijah je namreč tehnologija odjemalčeve strani mnogo bolj obremenjena. Skoraj celotno procesiranje aplikacije se zgodi na odjemalčevi strani in le del procesiranja oziroma pretežno shranjevanje podatkov poteka na strežniški strani. Običajne spletne aplikacije pa celotno procesiranje izvajajo na strežniški strani in le rezultat procesiranja prenesejo na odjemalčevo stran, kjer se ta tudi prikaže. Prav tu se pokaže ena izmed prednosti RIA aplikacij. Če končni uporabnik ne razpolaga s hitro internetno povezavo, postane interakcija z navadno spletno aplikacijo precej počasna in nelagodna, saj je za vsako operacijo potrebno preko mreže poslati zahtevek ter prejeti odgovor. RIA aplikacije se tej anomaliji delno izognejo tako, da celotno logiko izvajanja aplikacije prenesejo iz strežniške strani samo na začetku, ob prvem dostopu, in nato strežniško tehnologijo uporabljajo le za manjše operacije, kot je na primer shranjevanje podatkov. V takem primeru pa se čas zagona aplikacije bistveno podaljša.

Kot že omenjeno, bogato uporabniško izkušnjo RIA aplikacij skoraj v celoti omogoča tehnologija odjemalčeve strani, pod katero pa se natančneje šteje JavaScript oziroma AJAX ter razne vtiče za spletne brskalnike. Najpogosteje uporabljeni vtiči spletnih brskalnikov so: Adobe Flash, Java in Microsoft Silverlight. Logotipi omenjenih vtičev so prikazani na sliki 1.

Slika 1: Logotipi najpogosteje uporabljenih vtičev za spletne brskalnike v RIA aplikacijah.

[1] Varnostni mehanizem, ki omogoča varno izvajanje programske kode nepreverjenih virov.

Glede na delež uporabe so vtiči za spletne brskalnike v veliki prednosti pred AJAX tehnologijo. Razlog je v tem, da je bila JavaScript oziroma AJAX tehnologija precej časa okrnjena z njenimi funkcionalnostmi, medtem ko so razni spletni vtiči že od vsega začetka ponujali širok nabor funkcionalnosti. Poleg tega pa je RIA aplikacijo, ki bo enako delovala v vseh bolj uporabljenih spletnih brskalnikih, težje razviti z JavaScript oziroma AJAX tehnologijo kot pa na primer s tehnologijo Adobe Flash. Vsi spletni brskalniki namreč ne upoštevajo natančno spletnih standardov, v tem primeru JavaScript standardov, kar privede do razlik v funkcionalnostih, ki otežujejo razvoj. Pri uporabi vtičev za spletne brskalnike, kot je na primer Adobe Flash, pa do teh anomalij ne more priti, saj je platforma, v tem primeru Adobe Flash, v vseh spletnih brskalnikih enaka.

V zadnjem času se je pojavila tudi nova spletna tehnologija, natančneje standard HTML5. Ta prinaša veliko novosti, ki ustrezajo konceptom RIA. Kot primer njenih novih funkcionalnosti lahko izpostavim podporo za predvajanje video in avdio vsebin ter element canvas[2] za upravljanje z grafiko. Trenutno v RIA aplikacijah, z vidika uporabljene tehnologije za zagotavljanje bogate uporabniške izkušnje, še vedno prevladujejo vtiči za spletne brskalnike, vendar se lahko situacija kar hitro obrne.

Kot primer dobre prakse lahko izpostavim spletno mesto Google Docs. Gre za zbirko pisarniških aplikacij, zgrajenih s pomočjo JavaScript in AJAX tehnologije, ki omogočajo spletno urejanje raznih dokumentov. Omembe vredna je tudi velika količina spletnih iger, na primer na spletnem mestu bwin.com, pri katerih je pretežno uporabljena tehnologija vtičev spletnih brskalnikov oziroma Adobe Flash.

2.2.2 SOA

SOA oziroma storitveno usmerjena arhitektura označuje nov koncept načrtovanja programskih arhitektur, kjer so glavni gradniki storitve. Temelj takega načina arhitekture je omogočanje komunikacije med različnimi storitvami ter združevanje teh storitev v delujoč sistem.

Za vpeljavo SOA morajo biti izpolnjene naslednje zahteve:

- storitve morajo biti med seboj neodvisne, vsaka mora imeti nadzor nad svojo logiko izvajanja,
- logiko izvajanja je potrebno porazdeliti na storitve, da se tako omogoči ponovna uporaba le-te,
- storitev mora biti abstraktna, njena logika izvajanja ne sme biti dostopna ostalim storitvam oziroma uporabnikom,
- storitve morajo biti načrtovane tako, da jih je mogoče opisati s pomočjo meta podatkov[3] in na ta način kasneje tudi efektivno najti,
- storitve morajo biti med seboj čim bolj neodvisne, vsaka storitev mora o drugi storitvi vedeti le to, da ta obstaja,
- storitve morajo upoštevati komunikacijske dogovore, ki so določeni v opisih storitev,
- storitve morajo poskrbeti za minimalno porabo virov, tako da preložijo upravljanje z informacijami stanja, ko je to potrebno.

---

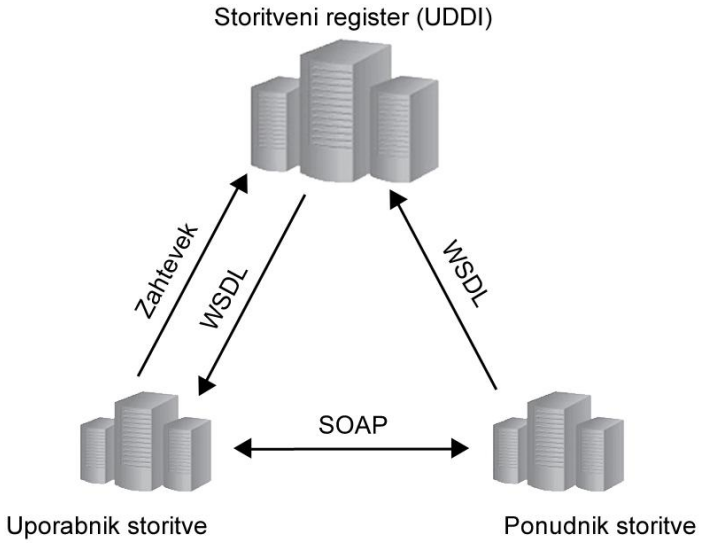
[2] Funkcionalnost označevalnega jezika HTML5 za predstavitev grafike.
[3] Podatki, ki opisujejo druge podatke, oziroma ponujajo informacije o določeni vsebini.

Ta pristop arhitekture se v zadnjem času vse bolj uporablja, saj sistemi potrebujejo več fleksibilnosti. Prednosti pri uvedbi take arhitekture pa je kar precej. Sistem je lahko porazdeljen - storitve se lahko izvajajo na ločenih lokacijah in niso več vezane na določeno platformo. Spremembo delovanja določene storitve je mogoče enostavno izvesti, saj ta ne bo ogrožala povezave z ostalimi storitvami oziroma deli sistema. Zaradi preglednosti nad izvajanjem storitev postane razvijanje večjih aplikacij bistveno hitrejše ter posledično cenovno ugodnejše.

SOA je ključnega pomena tudi v poslovno informacijski arhitekturi (*angl. enterprise architecture*), ki z namenom uskladitve informacijske tehnologije s poslovanjem, vzpostavlja oziroma povezuje več-funkcijske informacijske sisteme (ERP, CRM, PRM, SCM in KMS). Z vključitvijo konceptov SOA v poslovno informacijsko arhitekturo se namreč poslovna opravila, znotraj omenjenih informacijskih sistemov, realizira kot storitve skladno s pravili SOA, s čimer se omogoči večjo prilagodljivost hitro spreminjajočim se poslovnim procesom. Poslovni procesi postanejo bolj interoperabilni ter enostavnejši za spreminjanje oziroma nadgrajevanje, kar se posledično izraža tudi v večji agilnosti, učinkovitosti in produktivnosti poslovanja podjetja.

Običajno se izraz SOA povezuje kar s spletnimi storitvami, kar pa ni najbolj pravilno, saj so te le eden izmed načinov izvedbe SOA. Res pa je, da je ta način v praksi najpogosteje uporabljen. Ker je SOA široko področje, se bom v tem poglavju osredotočil predvsem na spletne storitve, ki predstavljajo tudi presek med področji SOA in spletom druge generacije.

Pri spletnih storitvah je komunikacija med storitvami najpogosteje implementirana s pomočjo tehnologij SOAP, WSDL ter UDDI. Spletna storitev lahko igra vlogo ponudnika storitve, uporabnika storitve, lahko pa tudi obe omenjeni vlogi. Arhitektura spletnih storitev temelji na treh osnovnih elementih, in sicer na uporabniku storitve, ponudniku storitve ter storitvenemu registru. Omenjeni elementi, skupaj z njihovimi medsebojnimi povezavami, so prikazani na sliki 2.



Slika 2: Temeljni elementi SOA.

Storitveni register je neke vrste spletni imenik za storitve, ki hrani podatke o posameznih ponudnikih storitev oziroma posameznih storitvah. Lahko je javni ali pa dostopen le znotraj

določenega omrežja. Za upravljanje s podatki uporablja neodvisni protokol UDDI, ki za posamezno spletno storitev omogoča vnos raznih meta podatkov ter WSDL zapis z informacijami o vmesniku storitve. Na podlagi teh podatkov lahko uporabniki oziroma razvijalci iščejo želene storitve in za njih pridobijo informacije o dostopu, obliki potrebnega zahtevka oziroma odgovora ter razne pravice uporabe. Naslednji element, ponudnik storitve, je v bistvu strežnik, ki izvaja storitve na podlagi uporabnikovih zahtevkov. Uporabnik storitve pa je lahko aplikacija ali pa kakšna druga spletna storitev, ki želi uporabljati določeno storitev ponudnika storitev.

Potek komunikacije med omenjeno infrastrukturo, z vidika storitvenega registra, pa poteka na sledeči način: ko ponudnik storitev razpolaga z novo storitvijo, se za slednjo v storitveni register objavi želene meta podatke ter WSDL zapis o informacijah njenega vmesnika. Uporabnik storitve se lahko nato poveže s storitvenim registrom in od njega zahteva opis želene storitve. Storitveni register mu želen opis vrne v formatu WSDL, iz katerega pridobi potrebne definicije parametrov, informacije za dostop do vmesnika ter informacije za uporabo storitve. Na podlagi teh podatkov uporabnik storitve zgradi zahtevek v obliki XML ter ga ponudniku storitve pošlje kot sporočilo SOAP, preko protokola HTTP. Ponudnik spletne storitve zahtevek obdela oziroma izvede storitev ter na koncu pošlje še rezultat v obliki XML nazaj uporabniku storitve, ravno tako kot sporočilo SOAP, preko protokola HTTP.

Kot primer uporabe spletnih storitev lahko navedem spletne aplikacije za turizem. Na primer, spletna aplikacija za rezervacijo letalskih vozovnic lahko pri nakupu letalske vozovnice ponudi popust pri nakupu storitve najema avtomobila neke druge sodelujoče družbe. Spletne storitve se tu uporabijo za uskladitev potrebnih podatkov med spletno aplikacijo letalske družbe in spletno aplikacijo družbe za najem avtomobilov. Tako se storitvi združita v delujoč sistem in končni uporabnik lahko nakup želenih storitev opravi na enem mestu.

Z vpeljavo SOA oziroma natančneje spletnih storitev v spletne aplikacije te pridobijo pravi pomen spleta druge generacije. Aplikacije postanejo bolj interoperabilne, bolj dinamične ter še bolj medsebojno povezane. Za širok nabor storitev ni več potrebno graditi ogromnih aplikacij oziroma za administracijo sistema ni več potrebna ogromna organizacija, ampak lahko s sodelovanjem tudi manjše organizacije ponujajo širok nabor kakovostnih storitev.

2.2.3 Socialni splet

Socialni splet označuje način uporabe spleta, ki izkorišča komunikacijske in socialne trende ter tako uporabnike spleta spodbuja, da opustijo klasično brskanje in postanejo aktivni soustvarjalci spleta. Tako splet ni več le podatkovna baza informacij, ampak kraj, kjer se uporabniki srečujejo, izmenjujejo vsebino, poslujejo, zabavajo ter še veliko več. Ob socialnem spletu se pogosto omenja tudi izraz kolektivna inteligenca, ki označuje pojav, ko veliko etičnih, ustvarjalnih in kritični uporabnikov spleta soustvarja ogromno količino vsebin. Primer je prosta spletna enciklopedija Wikipedija.

Večina današnjih socialnih spletnih mest temelji na skupnosti uporabnikov, ki jih družijo skupni interesi. Kot primer lahko izpostavim ljubitelje fotografije, ki svoje izdelke delijo z ostalimi sodelujočimi preko spletnih mest, kot sta Kodak gallery ter Flickr. Na podlagi skupnih interesov lahko socialna spletna mesta razdelimo na naslednjih pet glavnih kategorij:

- komunikacijska mesta: blogi, socialna omrežja;
- mesta za grajenje kolektivne inteligence: wikiji, socialni zaznamki;

- mesta za izmenjavo vsebin: izmenjava multimedijskih vsebin, izmenjava predstavitev;
- mesta za izmenjavo mnenj: Q&A;
- mesta za zabavo: virtualni svetovi, spletno igralništvo.

Kaj pa pravzaprav uporabnike žene k sodelovanju na raznih socialnih spletnih mestih? Odgovor na vprašanje je precej psihološke narave: želja po povečani prepoznavnosti, občutek efektivnosti ter občutek pripadnosti določeni skupnosti.

Za vpeljavo konceptov socialnega spleta v spletne aplikacije obstaja ogromno načinov. Seveda bodo tisti bolj razširjeni in inovativni privabili več sodelujočih ter posledično več prispevali k namenu aplikacije. Najbolj razširjeni načini za vpeljavo tega koncepta oziroma za omogočanje uporabniško generirane vsebine pa so: omogočanje komentiranja, ocenjevanja, komunikacije med uporabniki (zasebna sporočila in klepet), omogočanje uporabniško generiranih značk (*angl. tags*) ter omogočanje deljenja vsebine preko socialnih omrežij ali elektronske pošte. Za omogočanje vseh naštetih funkcionalnosti pa je potreben poglavitni dejavnik, to je profil uporabnika. Ta v bistvu služi za predstavitev uporabnika na določenem spletnem mestu ter kot referenca za vso njegovo ustvarjeno vsebino. Izoblikuje pa ga uporabnik sam na podlagi kombinacije naslednjih sedmih socialnih atributov:

- identiteta: Kdo si?
- ugled: Kako te ostali sodelujoči cenijo?
- prisotnost: Kje, na katerih socialnih mestih si dostopen?
- razmerja: S kom si povezan ter komu zaupaš?
- skupine: V katere skupnosti si vključen?
- pogovori: O čem se pogovarjaš z ostalimi sodelujočimi?
- deljena vsebina: Kakšne vrste vsebino deliš z ostalimi sodelujočimi?

V zadnjem času se je pojavil tudi nov trend v zvezi s socialnim spletom, in sicer integracija API-jev socialnih omrežij na razna mesta po celotnem svetovnem spletu. S temi API-ji se končnim uporabnikom poenostavi interakcijo med želenim spletnim mestom in povezanim socialnim omrežjem. Neposredni cilj tega početja oziroma natančneje razvijanja teh API-jev pa je pritegniti spletne razvijalce in posledično ustvariti več prometa končnih uporabnikov na socialnem omrežju, ki ponuja te vtiče. Posredni cilj pa je poenostavljanje ustvarjanja uporabniško generirane vsebine ter spodbujanje k še večji količini ustvarjanja te vrste vsebine. Primer prodirajočega API-ja, v kontekstu socialnega spleta, je tako imenovani Facebook Connect, ki je v lasti najbolj razširjenega socialnega omrežja Facebook. Slednji ponuja možnost, da se lahko uporabniki v različna spletna mesta prijavijo kar s Facebook identiteto. Tako se končnemu uporabniku olajša začetek sodelovanja na nekem novem spletnem mestu, saj mu ni več potrebno ustvariti novega uporabniškega računa, ki bo aktiven samo na tem spletnem mestu. Še en razširjen primer tovrstnih API-jev, ravno tako razvit s strani socialnega omrežja Facebook, je Facebook Like, ki omogoča preprosto deljenje všečne vsebine določenega spletnega mesta z ostalimi povezanimi uporabniki preko socialnega omrežja Facebook. Grafični vmesnik API-ja Facebook Like je prikazan na sliki 3.

Slika 3: Grafični vmesnik API-ja Facebook Like.

2.3 Značilnosti spletnih mest spleta druge generacije

Od začetkov spleta pa vse do danes se je na raznih spletnih mestih izoblikovalo nekaj značilnosti, ki se uvrščajo v sodobni splet oziroma se uporabljajo kot primer dobre prakse v aplikacijah spleta druge generacije. Med te značilnosti spadajo: značke, iskalniki, spletni viri (*angl. web feeds*), koncept povezovanja, koncept skupnega ustvarjanja vsebine ter uporaba tehnologije razširitev. Vsaka omenjena značilnost je v nadaljevanju natančneje obrazložena v svojem podpoglavju.

2.3.1 Značke

Značka je običajno ena beseda ali pa sklop besed, ki opisujejo vsebino članka ali druge vrste vsebin na raznih spletnih mestih. Kombinacija značk omogoča lažje opisovanje vsebine, lažjo kategorizacijo ter seveda lažje iskanje. Značke za izbrano vsebino običajno določi avtor vsebine, lahko pa tudi bralec, če sistem oziroma aplikacija to dovoljuje. Lahko se poljubno dodajajo ali pa so omejene z določenim dovoljenim naborom značk, odvisno od posameznega sistema. Prvotno so se značke pojavile na nekaterih spletnih mestih spleta druge generacije, kasneje pa so se zaradi njihove primernosti uporabe vpeljale skoraj na vsako spletno mesto in tudi v nekatere namizne aplikacije.

Za preslikavo omenjene teorije na praktičen primer si lahko predstavljamo fotografijo, na kateri je prikazan pes, ki leži v dnevni sobi na zofi. Tej fotografiji lahko na primer avtor dodeli znački pes in dnevna soba, kakšen drugi uporabnik, ki je drugačnega mišljenja, pa isti fotografiji lahko doda še tretjo oznako, na primer zofa.

Na zgornjem primeru se dobro izraža še en način uporabe spleta druge generacije, tako imenovana folksonomija oziroma uporabnikova taksonomija. Izraza se nanašata na razvrščanje oziroma natančneje na označevanje z značkami, ki ga za svoje spletne strukture opravi skupina sodelujočih končnih uporabnikov. Poleg prednosti - hitro rastoče uporabniško generirane klasifikacije vsebin - prinaša tudi nekaj slabosti. Kar hitro namreč lahko pride do anomalij, kot je dvojni pomen značk ali pa pretirano označevanje določene vsebine.

Na raznih spletnih mestih, ki ustrezajo konceptu spleta druge generacije in uporabljajo značke, je pogosto implementiran tudi oblak značk (*angl. tag cloud*). Slednji je v bistvu grafična upodobitev značk, ki so v uporabi na določenem spletnem mestu. Z uporabo različnih barvnih odtenkov in različnih velikosti pisave se izraža pomembnost oziroma gostota pojavitev posamezne značke. Vizualizirane značke običajno predstavljajo kar povezavo na prikaz, kjer je naveden spisek vsebin, označenih z določeno značko. Primer oblaka značk je na sliki 4.

Slika 4: Oblak značk.

2.3.2 Iskalnik

Skoraj vsako spletno mesto, ki ustreza konceptom spleta druge generacije in razpolaga z nekoliko večjo količino podatkov, vsebuje že svoj lastni iskalnik po vsebini. Slednji je postal skorajda nepogrešljiv del novodobnega spletnega mesta. Iskalnik lahko rezultate pridobi iz katerihkoli tekstovnih polj, shranjenih v podatkovni bazi - seveda odvisno od sistema (kaj ta omogoča ter kako je zmogljiv). Običajno se rezultati iskanja, tako imenovani zadetki, sklicujejo na izseke iz vsebinskega teksta, ključnih besed, značk ter opisov oziroma imen raznih multimedijskih vsebin.

Pri iskalnikih se pogosto omenjajo ključne besede, ki so precej konvergentne z že predhodno omenjenimi značkami. Ravno tako kot značka je ključna beseda običajno ena beseda ali sklop besed, ki opisujejo določeno vsebino. Razlika med omenjenima izrazoma je v tem, da so ključne besede običajno skrite končnemu uporabniku in se striktno uporabljajo le v raznih sistemih za lažje iskanje, medtem ko so značke končnim uporabnikom dostopne in jih ti lahko uporabljajo za lažje brskanje ter iskanje.

Da je splet druge generacije postal to, kar je danes, gre velika zasluga tudi večjim spletnim iskalnikom, kot so na primer Google, Yahoo in Bing, ki so v današnjem času postali nepogrešljivo orodje za iskanje podatkov po celotnem svetovnem spletnem prostoru. Zadetke vrnejo na enak način, kot je že omenjeno zgoraj pri enostavnejših iskalnikih, z razliko, da so tu zadetki pridobljeni iz celotnega svetovnega spleta. Za pridobivanje ažurnih podatkov ti spletni iskalniki uporabljajo tako imenovane spletne pajke, ki ves čas njihovega delovanja preiskujejo spletna mesta po celotnem svetovnem spletu.

V zadnjem času se je pojavila tudi združitev natančnih rezultatov najbolj razširjenega spletnega iskalnika Google in udobnosti iskanja podatkov znotraj določenega spletnega mesta. Podjetje Google je namreč ponudilo storitev z imenom Google Site Search. Gre za API, ki ga je mogoče vstaviti na katerokoli spletno mesto in omogoča iskanje zadetkov povezanih samo s spletnim mestom implementacije. Zaradi enostavnosti njegove implementacije in točnosti pridobljenih rezultatov se je funkcionalnost razširila na precej spletišč (*angl. websites*), primer takega spletišča je w3schools, ki je dostopno na URL naslovu http://www.w3schools.com/. Privzeta oblika omenjenega vmesnika je prikazana na sliki 5.

Slika 5: Privzeta oblika vmesnika Google Site Search.

2.3.3 Spletni vir

Spletni vir je podatkovna struktura, namenjena uporabnikom, ki spremljajo pogosto vsebinsko spreminjajoča se spletna mesta, kot so na primer blogi in novičarski portali. Razvili so se zaradi potrebe po hitrejšem dostopu do želenih informacij ter potrebe po samodejnem obveščanju o novih vsebinah. Raznim uporabnikom, predvsem tistim s počasnejšimi internetnimi povezavami, je namreč stalno pregledovanje novih vsebin vzelo precej časa. Spletni viri zato vsebujejo le vsebino prispevkov in najpomembnejše informacije, tako da je pregledovanje novih vsebin oziroma dostopanje do želenih vsebin res hitro ter tudi pregledno.

Tehnološko ozadje spletnih virov temelji na označevalnem jeziku XML. Natančneje, za vsak posamezen prispevek znotraj spletnega vira se ustvari svoj XML element, ki za svoj navezujoči prispevek vsebuje še XML podelemente za naslov, povezavo na izvor, identifikacijsko številko, datum objave, opis ter po možnosti še povezave do navezujočih multimedijskih vsebin. Za implementacijo spletnega vira se razširjeno uporabljata dva standardna formata, in sicer RSS ter Atom. Med njima je nekaj manjših zmogljivostnih razlik, bistveno pa se razlikujeta le v sintaksi.

Za prebiranje vsebine iz spletnih virov je potrebna programska oprema, označena pod imenom zbiralnik virov (*angl. feed aggregator*). Najpogosteje je ta integrirana kar v spletnem brskalniku, tako da uporabniku ni potrebno nameščati dodatne programske opreme. Omenjena oprema omogoča samodejno periodično preverjanje ažuriranosti podatkov, sporočanje o posodobitvi vsebine ter pregled želenih prispevkov. Da končni uporabnik začne uporabljati določeni spletni vir, je potrebno zbiralniku virov posredovati le naslov želenega spletnega vira.

Čeprav je bil spletni vir prvotno namenjen odpravi raznih nevšečnosti pri dostopu do vsebin s strani končnih uporabnikov, je bil načrtovan tudi tako, da omogoča enostavno razčlenjevanje oziroma enostavno branje s strani raznih aplikacij. Ta njegova značilnost se pogosto uporablja pri izmenjavi podatkov med ločenimi sistemi. Preprost primer je spletna aplikacija, ki pod svojo vsebino vključuje tudi določene podatke iz nekega drugega spletnega mesta. Da ob posodobitvi teh podatkov ni vedno potrebna človeška interakcija, je proces avtomatiziran tako, da spletna aplikacija periodično preverja ažuriranost njenih podatkov v skladu s podatki na spletnem viru drugega spletnega mesta ter po potrebi izvede še enosmerno sinhronizacijo. Za to početje, izmenjavo podatkov med spletnimi aplikacijami preko spletnega vira, se uporablja izraz Web syndication, katerega pravi pomen pa je večkrat zanemarjen ter asociiran kar s pomenom spletnih virov.

V spletu druge generacije je omogočanje spletnega vira na spletnih mestih postalo skorajda že pravilo. S pridom ga uporablja že večina uporabnikov spleta ter tudi enostavnejše spletne aplikacije za izmenjavo vsebin. Precej časa pa je tudi že uveljavljen razpoznavni znak, ki na posameznem spletnem mestu označuje razpoložljivost spletnega vira. Ta znak je prikazan na sliki 6.

Slika 6: Znak, ki na spletnem mestu označuje razpoložljivost spletnega vira.

2.3.4 Povezovanje

Koncept povezovanja se nanaša na povezovanje funkcionalnosti ter podatkov, porazdeljenih po svetovnem spletu, v delujoč sistem. Ta koncept se precej prekriva s principi SOA, ki so že opredeljeni v poglavju 2.2.2. Kot je že omenjeno, spletne aplikacije, ki ustrezajo temu konceptu, omogočajo večjo interoperabilnost, dinamičnost ter zmanjšujejo stroške razvoja. Pod koncept povezovanja pa spadajo tudi manj kompleksni načini implementacije, ki jih SOA sicer ne predpisuje. Ni namreč potrebno, da so vse operacije definirane kot storitve, bistveno je zniževanje stroškov razvoja z vključevanjem razpoložljivih funkcionalnosti iz svetovnega spleta ter širjenje prepoznavnosti aplikacije z omogočanjem razpoložljivosti določenih funkcionalnosti ostalim aplikacijam preko spleta. Omenjeni pristop je običajno realiziran z uporabo tako imenovanih API-jev, podrobneje predstavljenih v naslednjem podpoglavju 2.3.4.1, oziroma z združevanjem API-jev v storitev, imenovano mashup, ki je podrobneje predstavljena v podpoglavju 2.3.4.2.

2.3.4.1 API

API oziroma programski vmesnik je abstrakcija, ki opisuje potrebna pravila ter specifikacije za interakcijo s funkcionalnostmi kakšne druge aplikacije ali pa sistema. Izraz je sicer prvotno označeval zmožnost dostopa do funkcij operacijskega sistema in je šele nato pridobil širši pomen v medsebojno povezanih aplikacijah. Na platformi spleta je API običajno realiziran kot HTTP zahtevek, ki za odgovor dobi XML ali JSON zapis. Pogosta realizacija pa je tudi z neposredno vključitvijo celotne zaslonske maske API-ja v spletno aplikacijo. Ustrezen primer je široko razširjen Google Maps API, ki z uporabo skriptnega programskega jezika JavaScript omogoča prikaz Google Maps zemljevida neposredno v spletni aplikaciji ter upravljanje z njegovimi funkcionalnostmi, kot so na primer prikazovanje določene točke na zemljevidu ter izris najoptimalnejše poti med dvema lokacijama.

API-ji v bistvu predstavljajo enostavnejšo alternativo kompleksnejšemu SOA pristopu oziroma natančneje spletnim storitvam. Čeprav imata omenjena pristopa navidezno podoben cilj, se med seboj precej razlikujeta. SOA pristop strmi k natančni arhitekturi, ki lahko podpre mnogo operacij, vendar pa s to fleksibilnostjo narašča tudi kompleksnost izvedbe ter uporabe. Po drugi strani pa API-ji strmijo k hitri in enostavni izvedbi ter uporabi. Zadeva je podobna tudi pri uporabljeni tehnologiji. SOA je pretežno realizirana z uporabo kompleksne tehnologije SOAP, medtem ko so API-ji realizirani s preprosto in fleksibilno tehnologijo REST.

2.3.4.2 Mashup

Mashup je izraz za hibridno oziroma prepleteno storitev, ki integrira podatke ter funkcionalnosti iz več kot enega vira v novo delujočo storitev. Za njegovo izvedbo se običajno uporabljajo že omenjeni API-ji, ki poskrbijo za enostavno in hitro integracijo.

Primer dobre prakse takih storitev je spletno mesto woozor.com, ki za želeno točko na zemljevidu vizualizira vremensko napoved. Za vizualizacijo zemljevida uporablja Google Maps API, za pridobitev vremenoslovskih podatkov pa API spletišča Weather.com.

Slika 7: Spletno mesto woozor.com, kot primer mashup-a.

Arhitektura mashup-ov je sestavljena iz treh temeljnih gradnikov:

- komunikacije med različnimi viri s pomočjo API-jev,
- agregacije podatkov,
- vizualizacije oziroma predstavitve rezultata.

Glede na način predstavitve rezultata lahko mashup storitve razdelimo na dve kategoriji: podatkovno orientirane ter uporabniško orientirane. Podatkovno orientirana mashup storitev rezultat vrne ponovno v obliki podatka, medtem ko uporabniško orientirana mashup storitev rezultat vizualizira na uporabniku prijazen način.

Tako kot API-ji, ki v večini primerov omogočajo delovanje mashup storitev, imajo tudi mashup storitve enake navidezne podobnosti s koncepti SOA in ponovno se razlika pokaže v kompleksnosti izvedbe.

2.3.5 Skupno ustvarjanje vsebine

Skupno ustvarjanje vsebine lahko v kontekstu spleta druge generacije povežemo s pojmom kolektivne inteligence oziroma že omenjenim socialnim spletom, ki je kot ključni element spleta druge generacije predstavljen v zgornjem poglavju 2.2.3. Bistvo tega pristopa je namreč množično sodelovanje končnih uporabnikov, posledica česar je tudi ogromna količina uporabniško generirane vsebine na svetovnem spletu. Splet se tako bogati s komentarji, objavami na blogih, stalno dopolnjujočo vsebino wiki strani, skratka z vso vsebino, ki si jo uporabniki delijo preko svetovnega spleta.

2.3.6 Uporaba tehnologije razširitev

Uporaba tehnologije razširitev se nanaša na uporabo razširitev ali vtičev za spletne brskalnike pri razvoju RIA aplikacij. Tudi ta tema je že predstavljena, in sicer v poglavju 2.2.1 kot ena izmed temeljnih elementov spleta druge generacije. Če povzamem, z uporabo vtičev za spletne brskalnike, kot so Adobe Flash, Java ter Microsoft Silverlight, postanejo spletne aplikacije visoko interaktivne, odpravi se klasično osveževanje strani, skoraj celotno procesiranje pa se zgodi na uporabnikovi strani. Na tak način uporabnik spletno aplikacijo doživlja kot klasično namizno aplikacijo, ki se izvaja neposredno v operacijskem sistemu.

2.4 Značilna spletna mesta spleta druge generacije

V naslednjih podpoglavjih so predstavljena spletna mesta, ki vsebujejo večino karakteristik spleta druge generacije in se tudi uvrščajo med same predstavnike te nove različice spleta. To so wiki-ji, blogi, socialni zaznamki, socialna omrežja, mesta za deljenje multimedijskih vsebin ter spletne aplikacije.

2.4.1 Wiki

Wiki je izraz za spletišče, ki svojim etičnim, ustvarjalnim ter kritičnim uporabnikom omogoča kreiranje in urejanje neomejenega števila medsebojno povezanih spletnih wiki strani preko uporabniku prijaznega uporabniškega vmesnika. Beseda wiki, kot internetni žargon, izhaja iz havajske besede wikiwiki, kar pomeni »hiter« in se v kontekstu spleta nanaša na hitro ustvarjanje in urejanje vsebin. Wiki je med uporabniki postal zelo prepoznaven in tudi njegova uporaba je postala precej intuitivna. K temu je brez dvoma veliko pripomoglo trenutno največje in najbolj prepoznavno wiki spletno mesto z nazivom prosta enciklopedija Wikipedija. Logotip Wikipedije je prikazan na sliki 8.

Slika 8: Logotip Wikipedije.

Wiki spletišča najpogosteje poganja odprtokodna programska oprema MediaWiki, zaščitena z licenco GPL. Z istim naborom funkcionalnosti pa je na razpolago tudi manj razširjena konkurenčna programska oprema: Foswiki, MoinMoin, Tiki Wiki CMS Groupware, XWiki ter DokuWiki.

Glavni namen wiki-ja je omogočanje skupinskega dela oziroma grajenja tako imenovane kolektivne inteligence, ki pa se lahko izraža na različne načine. Na primer, v podjetniškem

svetu se wiki lahko uporabi kot nadomestek intranetu[4] ali pa za dokumentacijo določenega izdelka, v akademskem svetu pa za spodbujanje skupinskega učenja. Poleg urejanja vsebin wiki premore še vrsto ostalih funkcionalnosti, kot so na primer: iskanje želene vsebine, brskanje po vsebini, notranje povezovanje vsebin, nadzor nad spremembami vsebine ter urejanje pravic uporabnikov.

2.4.2 Blog

Blog ali spletni dnevnik je spletno mesto, ki ga urejevalec oziroma bloger uporablja z namenom deljenja informacij ter izkušenj na določeno temo s širšo množico uporabnikov spleta. Vsebina bloga je lahko poljubna in ne zahteva kakršnekoli sofisticiranosti ali kompleksnosti. Oblikovno ga zaznamujejo kronološko urejene objave, od najnovejše do najstarejše.

Njegov prvotni izraz je bil weblog, šele kasneje je pridobil krajše obliko imena, blog. Čeprav je kategoriziran kot predstavnik spleta druge generacije, njegove prve izvedbe segajo še v tako imenovani stari splet, ko je svoj blog posedovala le peščica spletnih uporabnikov. Njegova priljubljenost pa je iz dneva v dan rasla in danes prišla do točke, ko se nov blog na svetovnem spletu ustvari vsako sekundo. Z razcvetom spleta druge generacije pa je tudi pridobil določene dodatne funkcionalnosti. Tako blog danes omogoča: komentiranje objav, dodajanje značk, prikazovanje oblaka značk, prikazovanje arhiva objav glede na želeno časovno obdobje, deljenje vsebine preko socialnih spletnih mest, uporabo spletnega vira ter vpis na seznam spremljevalcev bloga. Na podlagi velike količine pojavljajočih blogov je bil skovan tudi nov izraz, blogosfera, ki se uporablja za sklicevanje na vse bloge in njihove medsebojne povezave na celotnem svetovnem spletu.

V današnjem času je ustvarjanje lastnega bloga postalo zelo enostavno in hitro opravilo. Na razpolago je namreč veliko število spletišč, ki ponujajo storitev registracije brezplačnega bloga zgolj v nekaj minutah. Tak primer je priljubljeno spletišče blogger.com, ki je v lasti podjetja Google. Na tržišču pa je prisoten tudi brezplačen CMS sistem Wordpress, ki je namensko izdelan prav za upravljanje blogov. Za uporabo tega pa je potreben lastni spletni prostor oziroma nakup storitve gostovanja.

2.4.3 Socialni zaznamki

Socialni zaznamki se navezujejo na spletna mesta, ki so se razvila z namenom nadomestitve klasičnega načina shranjevanja zaznamkov znotraj uporabnikovega spletnega brskalnika z novim načinom, baziranem na spletu. Glavno vodilo takih spletnih mest je odpravljanje treh glavnih pomanjkljivosti pri klasičnem hranjenju zaznamkov znotraj spletnega brskalnika:

- dostop do lastnih zaznamkov je mogoč samo iz določenega brskalnika oziroma računalnika,
- zaznamki so običajno neorganizirani in jih je posledično težje najti,
- nezmožnost enostavnega deljenja zaznamkov z ostalimi sodelujočimi uporabniki.

Z uporabo spletišča, ki ustreza konceptu socialnih zaznamkov, kot je na primer delicious.com, se omenjene pomanjkljivosti popolnoma odpravijo. Uporabniki, ki svoje zaznamke

[4] Notranji informacijski sistem zaključene skupine uporabnikov.

shranjujejo na želeno spletno mesto socialnih zaznamkov, niso več vezani na svoj brskalnik oziroma napravo, ampak do zaznamkov lahko dostopajo od kjerkoli in kadarkoli. Vsakemu posameznemu zaznamku lahko uporabnik dodeli želene značke, kar bistveno poenostavi organizacijo zaznamkov in tudi kasnejše iskanje. Spletna mesta socialnih zaznamkov omogočajo tudi sodelovanje med uporabniki ter tudi že omenjeno folksonomijo. Uporabniki lahko namreč želeno skupino zaznamkov delijo z ostalimi sodelujočimi ter tako tudi omogočijo skupno dodeljevanje značk.

2.4.4 Socialna omrežja

Socialno omrežje je novodobno spletno mesto, ki nudi širok spekter običajno brezplačnih storitev druženja, zabave ter deljenja vsebin v spletni skupnosti medsebojno povezanih uporabnikov.

Koncept sodobnih socialnih omrežij je v celoti zgrajen okrog končnega uporabnika in njegovih povezav z ostalimi uporabniki. Vsak uporabnik je tako rekoč prisiljen, da ustvari svoj uporabniški račun ter posreduje čim več osebnih podatkov, ki bodo opisovali njegov profil. Kasneje pa sledi spodbujanje k neprestanemu spremljanju ostalih uporabnikov istih interesov oziroma sodelovanju z njimi ter tudi lastnemu ustvarjanju vsebine. To sodelovanje oziroma ustvarjanje se izraža kot deljenje mnenj, ugotovitev, dogodkov, povezav ter tudi kot komentiranje vsebin, objavljanje multimedijskega materiala, vključevanje v razne skupine in še bi lahko naštevali. S prodorov velike količine uporabniško deljene vsebine oziroma razpoložljivih uporabnikovih osebnih podatkov pa so se povečale tudi izrabe razpoložljivih podatkov uporabnikov v škodljive namene. Za boj proti tem izrabam večina socialnih omrežij ponuja nastavitve varnosti, s pomočjo katerih uporabnik sam upravlja z zasebnostjo izpostavljenih podatkov.

Danes najbolj popularni socialni omrežji sta Facebook in Twitter. Na primer, na spletnem mestu Facebook je prisotnih že več kot četrtina prebivalcev Slovenije, kar predstavlja precej velik delež. Twitter, predstavnik tudi mikro bloganja, je sicer v Sloveniji nekoliko manj poznan, je pa precej razširjen v ostalih državah po svetu. Razpoznavna logotipa obeh omenjenih socialnih omrežij sta prikazana na sliki 9.

Slika 9: Razpoznavna logotipa socialnih omrežij Facebook in Twitter.

S konstantno rastjo popularnosti socialnih omrežij se je pojavil tudi nov prodirajoč trend selitve posla na razna socialna omrežja. Vse več družb oziroma organizacij na socialnih omrežjih že uspešno oglašuje svoj posel, blagovne znake, izdelke ali pa storitve. To je postal tudi najcenejši in najenostavnejši način za posredovanje oglasa potencialnim kupcem, saj naročnik oglasa lahko natančno določi želeno ciljno publiko na podlagi spola, starosti, jezika, države, itd.

2.4.5 Mesta za deljenje multimedijskih vsebin

Nezanemarljiv delež spletnih mest, ki ustrezajo konceptu spleta druge generacije, temelji zgolj na deljenju oziroma pregledovanju multimedijskih vsebin. Na taka spletna mesta se uporabnike običajno privabi s ponujanjem brezplačne storitve izmenjave vsebin ter seveda bogate uporabniške izkušnje. Spletna mesta, ki ustrezajo takemu načinu izvedbe in so tudi upravičena do naziva spletnega mesta spleta druge generacije, pa poleg omenjene osnovne funkcije deljenja vsebin omogočajo še klasične načine za ustvarjanje uporabniško generirane vsebine, kot so na primer: dodeljevanje značk, komentiranje, ocenjevanje, deljenje povezave preko socialnih mrež ter ustvarjanje seznama priljubljenih vsebin.

Primer dobre prakse je spletno mesto za izmenjavo videoposnetkov Youtube, ki svojim uporabnikom omogoča neomejeno objavljanje in pregledovanje raznih videoposnetkov ter uporabo vseh že omenjenih klasičnih načinov za ustvarjanje uporabniško generirane vsebine. Njegova preprostost in uporabnost sta pripomogla k ogromni količini aktivnih uporabnikov ter tudi k uveljavitvi tega spletnega mesta v poslovnem svetu filmske in glasbene industrije.

2.4.6 Spletne aplikacije

Kot značilna spletna mesta spleta druge generacije so se precej uveljavile tudi spletne aplikacije. To so spletna mesta, ki svojim končnim uporabnikom preko spleta ponujajo pomoč pri izvajanju raznih specifičnih opravil ali pa reševanju raznih problemov iz realnega sveta. Kot sem že omenil v obrazložitvi RIA aplikacij, so te precej podobne klasičnim namiznim aplikacijam, z razliko, da se spletne aplikacije izvajajo v spletnem brskalniku, da ne potrebujejo namestitve ter da so dostopne iz katerekoli napredne naprave, priključene na splet.

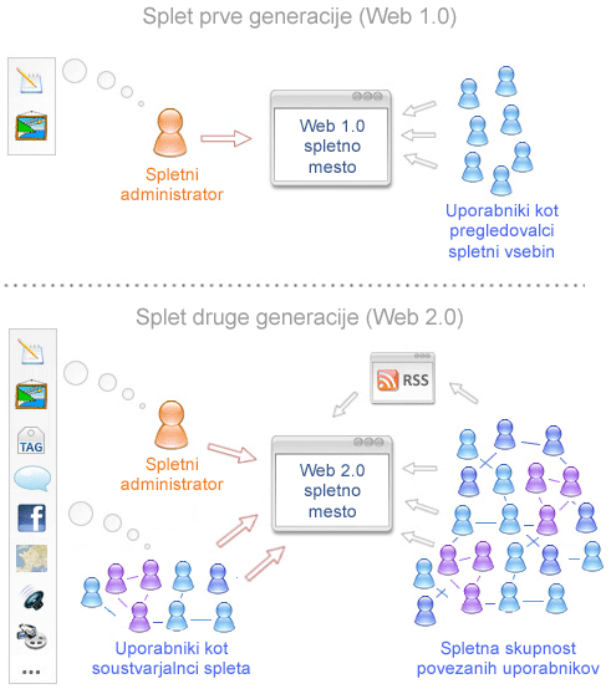
Kljub dokaj jasno podani definiciji spletne aplikacije pa je med njo in ostalimi tipi spletnih mest spleta druge generacije le tanka meja. Spletne aplikacije so namreč lahko poleg zelo kompleksnih primerkov, kot je napreden spletni urejevalnik besedil, tudi precej enostavne, kot na primer aplikacija za komentiranje oziroma skupno ustvarjanje vsebine. Prav ti enostavnejši primerki spletnih aplikacij pa se lahko kar hitro pomešajo z ostalimi novodobnimi spletnimi mesti. Tako se na primer lahko pod spletne aplikacije uvrščajo tudi wiki-ji ali socialna omrežja, ki pa seveda predstavljajo svojo vejo v spletnih mestih spleta druge generacije. Kljub širokemu spektru pa se običajno pod spletne aplikacije uvrščajo njene bolj kompleksne izvedbe, kot na primer: aplikacije za ponujanje spletne pošte, spletni urejevalniki besedil, spletni koledarji, CMS sistemi, aplikacije za vodenje raznih statistik, aplikacije za urejanje grafike, itd.

2.5 Razlika med spletom prve generacije in spletom druge generacije

Za dobro opredelitev meja spleta druge generacije je vsekakor pomembna primerjava z njegovim predhodnikom, tako imenovanim spletom prve generacije. Grobo razliko med različicama spleta lahko opredelimo na sledeč način: splet prve generacije je avtoritativen, statičen in zaprt, medtem ko je splet druge generacije demokratičen, dinamičen in sodelovalen. Natančnejša obrazložitev razlike sledi v nadaljevanju.

Za večino spletnih mest spleta prve generacije je bilo značilno, da je z njimi lahko upravljal le spletni administrator, ostali uporabniki spleta pa so na spletnem mestu lahko le pregledovali

oziroma prebirali vsebino. Tudi vsebina ni bila preveč bogata, običajno je obsegala le tekst in nekaj fotografij. Uporabniki so splet pretežno pregledovali individualno, njihova povezljivost je bila šibko vzpostavljena le preko komunikacijskega medija elektronske pošte. Z revolucijo spletne kulture pa se je komunikacijski in interakcijski proces uporabnikov spleta v celoti spremenil. Uporabniki spleta so postali aktivni soustvarjalci spletnih vsebin, začeli so se povezovati v razne spletne skupnosti in njihova medsebojna komunikacija je začela pretežno teči preko raznih socialnih spletnim mest. Velikega preobrata je bila deležna tudi vsebina. V spletu druge generacije ta namreč ne obsega več le teksta in fotografij, ampak lahko uporabniki na spletu delijo še video ter avdio vsebine, vključujejo funkcionalnosti preko enostavnih API-jev (primer: Google Maps API), želeni vsebini dodeljujejo značke, jo komentirajo ali pa povezavo do nje objavijo na socialnih omrežjih in še bi lahko naštevali. Za dostop do želenih vsebin pa je poleg neposrednega dostopa do spletnega mesta na voljo še enostavnejši in hitrejši način, natančneje dostop preko spletnih virov. Vse omenjene razlike z vidika socialne interakcije uporabnika so vizualizirane na sliki 10.

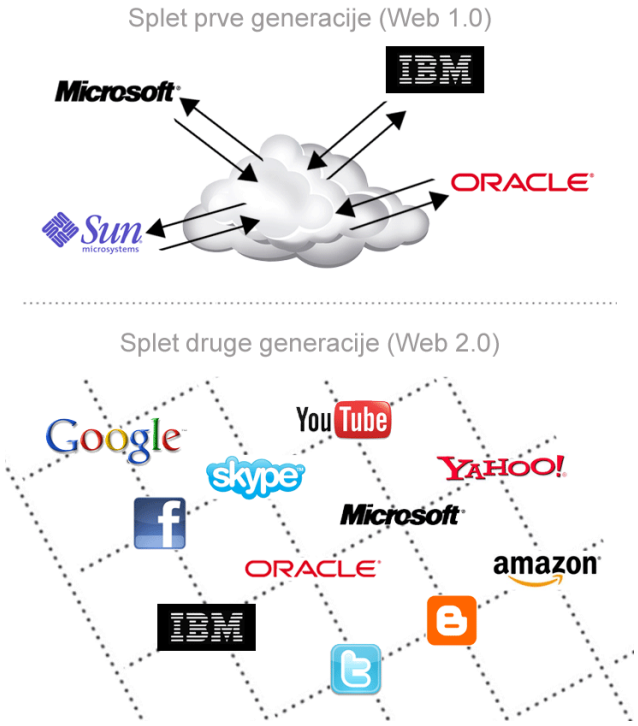


Slika 10: Vizualizacija razlik med spletom prve generacije in spletom druge generacije z vidika socialne interakcije uporabnika.

Na podlagi pridobljenih lastnosti socialne interakcije uporabnika pa so se posledično izoblikovale še dodatne razlike med starim in novim spletom. Splet druge generacije je namreč v primerjavi z njegovim predhodnikom postal manj formalno, končnim uporabnikom bolj dostopno in prijazno okolje.

Razlika med obema različicama spleta se izraža še v konceptu povezovanja storitev. Splet prve generacije velja za razmeroma zaprt sistem, brez možnosti povezovanja razpoložljivih storitev oziroma aplikacij. Večje organizacije, kot so na primer Oracle, IBM, Microsoft in Sun so s svojimi sistemi sicer delno omogočale povezovanja storitev, toda brez večjega uspeha oziroma ne v pravi smeri. V spletu druge generacije pa so koncepti povezovanja storitev v pravem razcvetu. Vsi večji sistemi in aplikacije temeljijo na SOA, oziroma natančneje svoje

storitve ponujajo na razpolago ostalim aplikacijam preko spletnih storitev ali pa API-jev. Omenjena razlika v povezovanju storitev je vizualizirana na sliki 11.



Slika 11:Vizualizacija razlik med spletom prve generacije in spletom druge generacije z vidika povezovanja storitev.

2.6 Tehnologija

Tehnologija, ki poganja današnja številna spletna mesta, se z evolucijo svetovnega spleta ni bistveno spremenila. Kot je že večkrat omenjeno, splet si nove različice ni prislužil na podlagi drastičnih sprememb v tehnoloških specifikacijah, ampak bolj na podlagi novega koncepta razvoja in novega načina uporabe spletnih tehnologij. Ta lastnost se lepo izraža v RIA aplikacijah, kjer bistveno vlogo igra nova tehnologija AJAX, ki je v bistvu le nov način kombinacije starejših tehnologij HTML, XML, JavaScript, XMLHttpRequest in DOM. Kljub majhnemu doprinosu spletnih tehnologij k evoluciji spleta, so spletne tehnologije vseeno omembe vredne in so zato v nadaljevanju natančneje obrazložene.

Celotno tehnologijo spleta delimo v tri skupine: tehnologijo odjemalčeve strani, tehnologijo strežniške strani ter spletne standarde. Vse tri omenjene skupine so predstavljene v naslednjih podpoglavjih.

2.6.1 Tehnologija odjemalčeve strani

Pod tehnologijo odjemalčeve strani se v kontekstu spleta štejejo tehnologije, ki so integrirane ali pa nameščene v uporabnikovem spletnem brskalniku. Zanje je značilno, da izvajajo izvorno kodo, ki se na podlagi zahtevka prenese iz strežnika v uporabnikov spletni brskalnik. Predstavljajo tudi poglavitni element za omogočanje bogate uporabniške izkušnje, ki je

značilna za že omenjene RIA aplikacije. Na podlagi teh kriterijev se v to kategorijo spletnih tehnologij uvrščajo: JavaScript, AJAX, JavaScript/AJAX ogrodja ter razni vtiči za spletne brskalnike, med katerimi je najbolj prepoznaven Adobe Flash.

2.6.1.1 JavaScript

JavaScript je poenostavljen objektni skriptni programski jezik, ki se ga uporablja za zagotavljanje interaktivnosti na spletnih straneh. Razvil ga je Netscape, temelji pa na sintaksi programskega jezika C. Čeprav samo ime precej spominja na programski jezik Java, sta bila programska jezika razvita neodvisno in sta precej različna. Druži ju le podobna sintaksa ter določene podobne knjižnice[5].

Glavni namen JavaScript-a je omogočanje interakcije z označevalnim jezikom HTML ter posledično poživitev strani z dinamičnim izvajanjem. Njegov nabor funkcionalnosti tako obsega: kreiranje HTML elementov, brisanje oziroma urejanje HTML elementov, spreminjanje stilskih lastnosti HTML elementov, upravljanje z dogodki, upravljanje s piškotki[6], prepoznavanje uporabljenega tipa spletnega brskalnika in še bi lahko naštevali. Pogost primer uporabe JavaScript-a v praksi je preverjanje pravilnosti podatkov v vnosnih poljih. Omenjeni programski jezik omogoča, da se ob pravilnosti podatkov izvede želena akcija ali pa se v nasprotnem primeru pojavi obvestilo o napačnih podatkih.

JavaScript je integriran v vseh sodobnih spletnih brskalnikih, tako da ne potrebuje nobene namestitve ali konfiguracije pred njegovo prvo uporabo. S to integracijo pa je onemogočeno njegovo posodabljanje, kar se pokaže kot anomalija, ko se določeni JavaScript ukazi v novejših spletnih brskalnikih drugače izvedejo kot v starejših spletnih brskalnikih.

2.6.1.2 AJAX

AJAX je okrajšava za asinhroni JavaScript in XML, ki označuje skupino medsebojno povezanih spletnih tehnologij, uporabljenih za zagotavljanje interaktivnosti na spletnih straneh. Bistvo njegovega delovanja predstavlja asinhrono izmenjevanje podatkov s strežnikom v ozadju, brez potrebe po klasičnem osveževanju strani. Na tak način zagotavlja tekoč, nemoten, pregleden in odziven način za spremljanje podatkov na želeni spletni strani tudi med komunikacijo s strežnikom. Poleg že omenjene poglavitne dobre lastnosti pa obstaja še ena prednost njegove uporabe, izražena z vidika porabe virov. Ker se z asinhronim prenosom prenašajo le najpomembnejši podatki oziroma želeni deli spletne strani in ne celotna stran, to posledično razbremeni strežnik ter seveda tudi internetno povezavo. Poleg vseh naštetih prednosti pa zraven pridejo tudi določene slabosti. Dinamično spreminjanje vsebine prikaza z uporabo tehnologije AJAX namreč ne spremeni trenutnega URL naslova, kar onemogoča določene samoumevne akcije oziroma funkcionalnosti, kot so na primer: pravilno delovanja gumba »Nazaj«, shranjevanje določenega stanja prikaza pod priljubljene vsebine ter indeksiranje vsebine s strani spletnih pajkov.

---

[5] Zbirka virov (procedur, razredov, vrednosti, …), ki se uporabljajo pri razvoju programske opreme.

[6] Podatki, ki jih v uporabnikovem računalniku spletni brskalnik, na pobudo določenega spletnega mesta, shrani za poznejšo rabo.

Asinhroni prenos podatkov, tako rekoč temelj AJAX-a, se izvaja s pomočjo objekta XMLHttpRequest, ki je integriran v vseh sodobnih spletnih brskalnikih. Za vzpostavitev AJAX-a pa je poleg tega temeljnega objekta zahtevana uporaba še naslednjih tehnologij: HTML-ja za vizualizacijo podatkov, DOM-a za ažuriranje izbranega mesta na spletni strani, XML-ja za upravljanje s podatki ter JavaScript-a za povezovanje naštetih tehnologij. Kljub omenjeni zahtevani tehnologiji XML, pa ta ni nujno potrebna za delovanje AJAX-a. Z začetki uporabe AJAX-a se je namreč večinoma asinhrono v ozadju prenašal le XML, danes pa je tega nadomestila kar HTML koda.

Komunikacija med tehnologijo AJAX-a poteka na sledeč način: ko programska koda JavaScript zazna določen dogodek, objektu XMLHttpRequest sporoči naslov strežnika ter zahtevek, ki naj ga pošlje strežniku. Objekt XMLHttpRequest zahtevek tudi pošlje in ko od strežnika prejme odgovor na zahtevek, ga nemudoma posreduje nazaj JavaScript-u. Nato JavaScript programska koda odgovor strežnika v morebitni XML obliki še obdela v želen rezultat oblike HTML in na koncu, s pomočjo tehnologije DOM, rezultat vstavi v izbrano mesto na spletni strani.

Zgleden primer uporabe AJAX-a je storitev Google Instant, ki jo ponuja spletni iskalnik Google. Ta namreč v storitvi iskanja s pomočjo AJAX-a samodejno predlaga iskane nize, medtem ko uporabnik tipka besedilo v vnosno polje. Poleg tega pa se zadetki prikazujejo samodejno in sočasno glede na iskani niz v vnosnem polju.

2.6.1.3 JavaScript/AJAX ogrodja

JavaScript/AJAX ogrodja oziroma JavaScript knjižnice (*angl. JavaScript libraries*) so zbirka pred-napisane JavaScript programske kode, ki spletnim razvijalcem olajša razvijanje spletnih aplikacij, baziranih na programskem jeziku JavaScript. Z njihovo uporabo se lahko spletni razvijalci osredotočijo bolj na reševanje problemov programerske narave in manj na zagotavljanje dinamičnosti uporabniškega vmesnika. Pogoste funkcionalnosti takih orodij so: skrajševanje programske kode JavaScript, poenostavljeno upravljanje s stilskimi lastnosti HTML elementov, dodajanje animacije HTML elementom (npr. efekta bledenja), poenostavljeno upravljanje z dogodki, poenostavljeno upravljanje z AJAX-om, odpravljanje nekonsistentnosti med različnimi spletnimi brskalniki ter generiranje kontrol (npr. drsniki in koledarji).

Danes najbolj prepoznavna tovrstna ogrodja oziroma knjižnice so: JQuery, Mootools ter Prototype. Logotipi vseh treh omenjenih JavaScript/AJAX ogrodij so prikazani na sliki 12.

Slika 12: Logotip najbolj prepoznavnih JavaScript/AJAX ogrodij.

Omenjena ogrodja so objavljena v obliki odprte kode, tako da jih lahko vsakdo uporablja brez kakršnegakoli plačila. Ker so ta ogrodja v bistvu skupek gole JavaScript kode, lahko vsakdo enostavno, brez kakršnegakoli orodja, vpogleda v programsko kodo ter jo po želji tudi spremeni. Datoteka, ki vsebuje to ogrodje, je običajno stisnjena (*angl. compressed*) oziroma

ima odstranjene komentarje ter nepotrebne presledke, tako da se v uporabnikov spletni brskalnik čim hitreje naloži.

2.6.1.4 Adobe Flash

Adobe Flash je tehnologija podjetja Adobe, ki se na spletnih straneh uporablja za dodajanje interaktivnih animacij in videoposnetkov. Za uporabo njegovih funkcionalnosti je potrebna namestitev vtiča za spletne brskalnike z imenom Adobe Flash Player. Z uporabo te tehnologije se je pred kratkim lahko počelo veliko stvari, ki z uporabo samega HTML-ja niso bile mogoče. Tak primer je predvajanje video in avdio vsebin ter uporaba vektorske grafike, kar pa sedaj omogoča tudi novi standard HTML5.

Za razvijanje interaktivnih spletnih funkcionalnosti s tehnologijo Adobe Flash je potreben plačljiv program, katerega zadnja različica je znana pod imenom Adobe Flash Professional CS5. To je v bistvu multimedijski grafični program za izdelavo animacij, katere se lahko popestri s programsko kodo integriranega objektno usmerjenega programskega jezika ActionScript. S tem razvojnim okoljem oziroma programom se razvija Flash animacije, za katerih prikaz na spletni strani je potrebno vključiti HTML kodo z navedbo poti do želene animacije. Primer take HTML kode je viden na sliki 13.

```
<object width="550" height="400">
  <param name="movie" value="somefilename.swf">
  <embed src="somefilename.swf" width="550" height="400"></embed>
</object>
```

Slika 13: Primer HTML kode za vključitev Flash animacije na spletno stran.

Na spletu se tehnologija Adobe Flash najpogosteje uporablja za reklamne pasice, spletne igre, predvajanje multimedijskih vsebin ter za zagotavljanje bogate uporabniške izkušnje v RIA aplikacijah. S to tehnologijo pa je mogoče zgraditi tudi celotno spletno mesto oziroma celotno spletno aplikacijo, kar se v bistvu z vidika HTML kode kaže kot ena velika Flash animacija. Za take primerke, pa tudi za enostavnejše Flash animacije, je značilno, da se ob uporabnikovem dostopu do njih, te v celoti prenesejo v uporabnikov brskalnik. Za njihovo operiranje oziroma izvajanje v večini koristijo le vire uporabnikove naprave, razen manjših potrebnih operacij, kot sta na primer shranjevanje podatkov ter pridobitev manjše količine ažurnih podatkov, ki koristijo strežniške vire. Tak način izvajanja seveda pripomore k razbremenitvi strežniške tehnologije ter tudi internetne povezave, kot negativno posledico pa prinaša večji zaganjalni čas.

2.6.2 Tehnologija strežniške strani

Na začetku, ko je bil splet še statičen sistem, so strežniki servirali le dokumente in slike, brez kakršnekoli omogočene interakcije, razen brskanja. S pojavom želje po večji interakciji so se začele razvijati tehnologije na strežniški strani, ki so na podlagi parametrov v zahtevku vrnile dinamičen, prilagojen rezultat, ki je temeljil na podatkih, običajno shranjenih v podatkovni bazi. Te tehnologije so se začele razvijati že v spletu prve generacije in večina se jih z manjšimi nadgradnjami še danes uporablja v spletu druge generacije. Tehnologija strežniške strani tako s prehodom iz spleta prve generacije v splet druge generacije ni bila deležna

veliko sprememb. Res je, da se stalno razvijajo nove različice teh tehnologij, vendar koncept ostaja še vedno skoraj isti. Omembe vredna je le manjša sprememba, povezana s prihodom konceptov SOA, in sicer povečanje podpore pri izmenjevanju podatkov med aplikacijami. Kljub majhnim spremembam pa je tehnologija strežniške strani vsekakor pomembna v spletu, saj brez nje splet druge generacije ne bi uspel zaživeti. Današnje najbolj razširjene tehnologije strežniške strani so: PHP, ASP.NET, JSP, ColdFusion, Pearl, Ruby, Python. Vse omenjene tehnologije so natančneje predstavljene v naslednjih podpoglavjih.

2.6.2.1 PHP

PHP oziroma PHP: Hypertext Preprocessor je trenutno najbolj priljubljen odprtokodni, skriptni programski jezik za razvoj dinamičnih spletnih mest. Pojavil se je leta 1995 pod imenom PHP Tools oziroma Personal Home Page Tools kot skupek enostavnih orodij, ki so uporabniku na primer omogočala vpogled v sistemski zapisnik ter enostavno procesiranje spletnih obrazcev. Od njegove prve pojavitve naprej se je hitro razvijal in postal sofisticirana strežniška tehnologija, ki danes poganja večino najbolj obiskanih spletišč na svetovnem spletu. Izvorno ime PHP Tools se je sčasoma porazgubilo, tako da se danes za njegovo poimenovanje uporablja le ime PHP oziroma PHP: Hypertext Preprocessor. Običajno teče na spletnem strežniku Apache, lahko pa ga poganjajo tudi nekateri drugi spletni strežniki.

Spletnim razvijalcem daje na razpolago širok nabor naprednih zmogljivosti, ki ustrezajo resnemu spletnemu razvoju tako majhnih kot velikih projektov. Omogoča vse, od enostavne povezave s podatkovno bazo do kompleksnih komunikacij med aplikacijami. S pomočjo te tehnologije so se razvili tudi močni CMS sistemi, kot so na primer WordPress, Drupal in Joomla, s katerimi lahko tudi manj vešči spletni razvijalci razvijajo napredna spletna mesta. Kljub njegovi veliki priljubljenosti pa je deležen tudi nekaj kritik, kot so na primer: nekonsistentno poimenovanje njegovih funkcij, pred vsakim imenom spremenljivke je potrebno napisati dolar in vsaka njegova nova verzija podre združljivost določenih funkcij za nazaj.

PHP programska koda je shranjena v datotekah s končnico ».php«, v kateri se običajno nahaja tudi HTML koda. Za ločevanje HTML kode od PHP programskih ukazov pa se uporabljata sklopa znakov <?php in ?>, med katerima se lahko nahaja PHP programska koda, zunaj njih pa HTML koda.

2.6.2.2 ASP.NET

ASP.NET oziroma Active Server Pages .NET je napredna strežniška tehnologija podjetja Microsoft za grajenje dinamičnih spletnih strani, spletnih aplikacij in spletnih storitev. Predstavlja naslednico nekoliko starejše tehnologije ASP, za katero je Microsoft leta 2000 ukinil podporo in se tako bolj posvetil razvoju tehnologije ASP.NET. Ta za svoje delovanje uporablja ogrodje Microsoft .NET framework, ki omogoča uporabo širokega nabora uporabnih knjižnic ter zmožnost pisanja programske kode v več različnih programskih jezikih, med katerimi sta najpogosteje uporabljena C#[7] in VB.NET[8]. Kot njegov predhodnik

[7] Splošno uporaben objektno usmerjen psevdokodni programski jezik, ki temelji na programskem jeziku C++.
[8] Kratica za programski jezik Visual Basic .NET.

se tudi ASP.NET izvaja na spletnem strežniku Internet Information Services, ki je ravno tako v lasti podjetja Microsoft.

ASP.NET predstavlja precej močno tehnologijo, ki je še posebej primerna za razvijanje večjih projektov na spletu. Za razvoj majhnih spletnih mest pa se običajno ne uporablja, saj se efektivnost njegove uporabe tu precej zmanjša oziroma pride do vpeljevanja pretirane kompleksnosti v razvoj spletnega mesta.

Ena izmed boljših značilnosti te tehnologije je avtomatizacija večine procesa sestavljanja spletne strani, tako da se spletni razvijalec lahko osredotoči bolj na reševanje problemov programerske narave in manj na samo implementacijo postavitve spletne strani. To se precej dobro izraža v funkcionalnosti vključevanja ter upravljanja s kontrolami, kot so na primer: podatkovna mreža (*angl. data grid*), zemljevid strani, meni ter razna pred-pripravljena vnosna polja.

Pri uporabi tehnologije ASP.NET sta HTML koda in programska koda izbranega programskega jezika običajno ločeni v dve različni datoteki. V datoteki s končnico ».aspx« se nahaja HTML koda za postavitev strani in definicijo kontrol, medtem ko se v istoimenski datoteki s končnico ».aspx.cs« oziroma ».aspx.vb« nahaja programska koda za izvajanje določenih akcij ob deklariranih dogodkih.

2.6.2.3 JSP

JSP oziroma JavaServer Pages je strežniška tehnologija, bazirana na programskem jeziku Java, ki jo je podjetje Sun izdelalo z namenom konkurirati tehnologiji ASP. Predstavlja precej močno tehnologijo, ki je načrtovana prav za grajenje velikih spletišč oziroma večjih spletnih aplikacij. Spletnim razvijalcem daje na razpolago knjižnice programskega jezika Java, z uporabo katerih se količina potrebne napisane programske kode bistveno zmanjša. Za njegovo delovanje je značilno, da se datoteka z izvorno kodo (datoteka s končnico ».jsp«) najprej prevede v Java Servlet[9] (datoteka s končnico ».java«), ki nato na njegov klasičen način obdeluje zahtevke in posreduje odgovore preko protokola HTTP. To tehnologijo običajno poganja spletni strežnik Tomcat, z implementacijo določenih vtičev pa to lahko omogočajo tudi ostali strežniki.

2.6.2.4 ColdFusion

ColdFusion je strežniška tehnologija podjetja Adobe, ki se je prvotno razvila z namenom omogočanja enostavnega povezovanja HTML kode s podatkovno bazo. Vključuje tako svoj strežnik kot tudi svoj programski jezik. Strežnik, katerega zadnja različica je znana pod imenom Adobe ColdFusion 9, za njegovo delovanje v ozadju uporablja zmogljiv programski jezik Java, kar zagotavlja njegovo delovanje na operacijskih sistemih Windows, Mac OS X ter Linux. Za pisanje programske kode ponuja svoj programski jezik z imenom CFML oziroma ColdFusion Markup Language, ki temelji na značkah in je na prvi pogled precej podoben označevalnemu jeziku HTML. Prav ta programski jezik olajša večino težkega programerskega

[9] Aplikacija, napisana v programskem jeziku Java, ki se izvaja na strežniku in odgovarja na zahtevke preko protokola HTTP.

dela, kar naredi to tehnologijo z vidika spletnega razvijalca zelo enostavno za uporabo. Ker je ColdFusion zgrajen s programskim jezikom Java, lahko naprednejši spletni razvijalci, poleg širokega nabora funkcionalnosti programskega jezika CFML, neposredno uporabljajo tudi pred-definirane objekte programskega jezika Java, kar naredi to tehnologijo precej zmogljivo.

2.6.2.5 Perl

Perl je visokonivojski splošno namenski skriptni programski jezik, katerega sintaksa spominja na programski jezik C. Je ena izmed najstarejših strežniških tehnologij, ki se danes še vedno uporablja. Kljub temu, da je tehnologija že starejša, vseeno omogoča vse potrebne funkcionalnosti za napreden razvoj spletnih mest spleta druge generacije. Da Perl postane spletnim razvijalcem bolj prijazen, ti uporabljajo ogrodji Catalyst in Jifty, ki odpravljajo določene slabosti te starejše tehnologije. Za njegovo izvajanje na spletu pa se ga povezuje s spletnim strežnikom Apache.

2.6.2.6 Ruby

Ruby je večnamenski enostaven in močan objektno usmerjen programski jezik, ki nekoliko spominja na programska jezika Python in Perl. Pojavil se je že leta 1995, na spletu pa je zaslovel šele leta 2004 z izidom na njemu zgrajenega odprtokodnega ogrodja Ruby on Rails. To ogrodje je bilo zasnovano z namenom pohitritve in poenostavitve klasičnega spletnega razvoja s pomočjo raznih novih orodij. Primer takega orodja je Scaffolding, ki sam avtomatsko zgradi večino temeljnih elementov spletnega mesta. Za blog, na primer, sam zgradi osnutek prikaza za pregled določene objave ter osnutek prikaza za kreiranje nove objave. Še ena omembe vredna funkcionalnost ogrodja Ruby on Rails je knjižnica ActiveRecord, ki za upravljanje z vsemi podprtimi podatkovnimi bazami omogoča pisanje enakih programskih ukazov, v nasprotju z ostalimi tovrstnimi rešitvami, ki za vsako podprto podatkovno bazo posebej zahtevajo svoje programske ukaze. Skratka, to ogrodje na enostaven in nov način spletnim razvijalcem pomaga prihraniti čas in povečuje njihovo produktivnost. Ruby glede izbire strežnika nima preferenc oziroma se lahko izvaja na kateremkoli strežniku, ki implementira CGI pristope, na primer na spletnem strežniku Apache.

2.6.2.7 Python

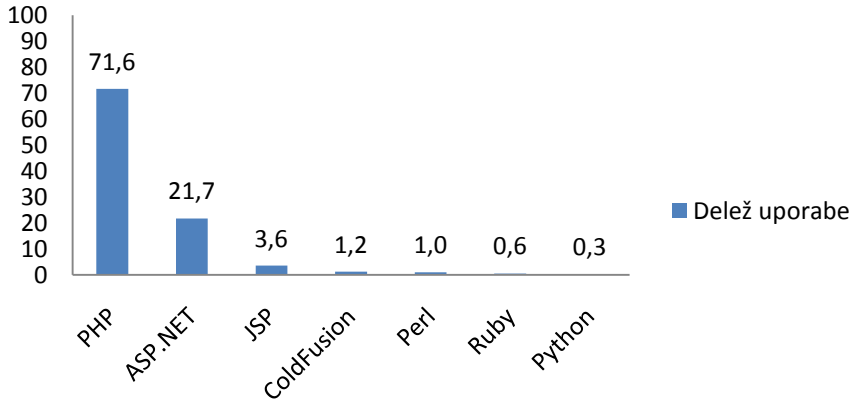
Python je odprtokodni skriptni programski jezik, ki stremi k združevanju izjemne izrazne moči in jasne sintakse. Je integriran kot standardna komponenta na Linux in Mac OS X platformah, lahko pa se ga namesti tudi na Windows platformi. Za lažje spletno razvijanje daje na razpolago ogrodje Django, ki poenostavlja delo z raznimi knjižnicami, namenjenimi spletu. Kljub številnim dobrim lastnostim pa se na spletu žal ni preveč uveljavil, tako da se le malo spletnih razvijalcev odloči za njegovo uporabo. Kot večina ostalih tovrstnih rešitev se tudi Python običajno izvaja na spletnem strežniku Apache.

2.6.2.8 Primerjava tehnologij strežniške strani

Pri razvijanju raznih spletnih mest so spletni razvijalci, še posebej začetniki, večkrat postavljeni pred dilemo, katero strežniško tehnologijo uporabiti za razvoj naslednjega

spletnega mesta. To je vsekakor pomembna odločitev, saj bo spletni razvijalec oziroma ekipa le-teh z uporabo prave tehnologije strežniške strani prihranila čas in denar. Težko pa je soditi, katera izmed trenutno razpoložljivih tehnologij strežniške strani je boljša od druge, saj večinoma vse ponujajo enak nabor funkcionalnosti. Zato se te tehnologije običajno primerjajo glede na količino razpoložljive podpore oziroma glede na njihovo priljubljenost, hitrost učenja spletnih razvijalcev ter ceno potrebnih razvojnih okolij in strežnikov.

Če spletni razvijalec prisega na uporabo tehnologije, ki je najbolj razširjena in ima posledično tudi največ razpoložljive podpore pri reševanju raznih programerskih težav, je njegova odločitev vsekakor tehnologija PHP. Ta tehnologija strežniške strani namreč poganja kar 71,6 % najbolj obiskanih spletnih mest na svetovnem spletu, kar jo uvršča na sam vrh najbolj priljubljenih tehnologij strežniške strani. Deleži razširjenosti vseh trenutno najbolj uporabljenih tehnologij strežniške strani so vizualizirani na sliki 14.



Slika 14: Deleži razširjenosti vseh trenutno najbolj uporabljenih tehnologij strežniške strani na podlagi enega milijona najbolj obiskanih spletnih mest v mesecu marcu leta 2011.

Glede na hitrost učenja posameznega programskega jezika oziroma tehnologije strežniške strani pa prevladuje tehnologija Ruby oziroma ogrodje Ruby on Rails. Slednje z uporabo novih konceptov razvoja precej pohitri učenje spletnih razvijalcev, pa tudi samo izgradnjo spletnega mesta. Z vidika cene gostovanja na strežniku oziroma cene razvojnih okolij so skoraj vse tehnologije ekvivalentne, razen tehnologij ColdFusion in ASP.NET, ki sta plačljivi in zato tudi nekoliko manj priljubljeni s tega stališča.

2.6.3 Standardi

Poleg že omenjenih tehnologij odjemalčeve in strežniške strani so za poganjanje spleta potrebni še spletni standardi. To so tehnične specifikacije, ki definirajo ter opisujejo delovanje spleta, tako z vidika komunikacije strežnika in odjemalca kot tudi ustvarjanja in interpretiranja spletnih vsebin. Za večino spletnih standardov je odgovornost prevzela mednarodna organizacija World Wide Web Consortium, ki jih sedaj razvija in skrbi za njihovo pravilno rabo.

Vsi ti standardi so bili že prisotni v spletu prve generacije, z evolucijo spleta so se le nekoliko posodobili oziroma prilagodili potrebam novega koncepta spletnega razvoja. Temeljni spletni

standardi so: označevalna jezika HTML in XML, stilne predloge CSS, standard DOM, komunikacijski protokol HTTP in standard URI.

2.6.3.1 Označevalni jeziki

2.6.3.1.1 HTML

HTML je štiri črkovna okrajšava za angleški izraz Hyper Text Markup Language, ki se nanaša na standardni označevalni jezik za opis spletnih strani. Omogoča definiranje strukture in izgleda vsebin spletnih strani, ki se prikažejo v spletnem brskalniku. Temelji pa na skupku strukturiranih pred-definiranih značk, kot so na primer <head>, <body> in <h1>, med katerimi ima vsaka svoj pomen in privzeto obliko.

Od prve pojavitve HTML-ja pa do danes je bilo izdanih že kar nekaj različic tega označevalnega jezika, med katerimi je vsaka novejša prinesla nove zmogljivosti in bolj dovršeno sintakso. Trenutno se priporoča uporabo različice XHTML 1.0, ki je zaradi uskladitve s sintakso XML nekoliko čistejša in strožja v primerjavi z njenimi predhodnimi različicami. Na razpolago pa je tudi novejša različica HTML 5, ki jo na žalost še ne podpirajo vsi spletni brskalniki in se zato spletni razvijalci le redko odločijo za njeno uporabo.

2.6.3.1.2 XML

XML oziroma eXtensible Markup Language je enostaven in razširjen označevalni jezik, ki je precej podoben označevalnemu jeziku HTML. Razvit je bil z namenom opisovanja in shranjevanja strukturiranih podatkov ter prenosa le-teh med različnimi aplikacijami. Za strukturirano shranjevanje podatkov uporablja značke, ki pa za razliko od HTML-ja niso vnaprej definirane, ampak jih avtor XML-ja določi kar sam. Enostavna manipulacija s podatki, shranjenimi v XML-ju, je omogočena z uporabo razčlenjevalnika kode (*angl. parser*), ki je danes integriran že v vsakem naprednem programskem jeziku. Omenjeno orodje namreč z raznimi funkcijami branja, pisanja, spreminjanja in brisanja podatkov omogoča, da se razvijalci bolj posvetijo obdelovanju podatkov in manj zamudnemu razčlenjevanju XML-ja.

S prihodom spleta druge generacije in posledično velike potrebe po izmenjevanju podatkov med spletnimi aplikacijami je XML na spletu pridobil svoj pravi pomen. Postal je nepogrešljivo orodje za grajenje tako obsežnejših spletnih aplikacij kot tudi manjših spletih mest.

2.6.3.2 CSS

CSS oziroma Cascading Style Sheets je izraz za stilne predloge, ki določajo izgled spletne strani oziroma natančneje izgled HTML elementov. Z njimi lahko vsakemu HTML elementu določimo cel kup stilnih lastnosti, kot so na primer: postavitev, dimenzije, ozadje, obroba, slog pisave, itd. Razvite so bile z namenom odprave težave velike količine ponavljajoče se HTML kode za oblikovanje besedila, ki se je začela pojavljati s prihodom značke <font>. Stilne predloge so omenjeno težavo rešile tako, da se posamezna oblika definira samo enkrat

in nato uporabi za celoten HTML dokument oziroma za določen sklop enakih HTML elementov. Kodo stilnih predlog se običajno shranjuje v zunanjo datoteko s končnico ».css«, ki se potem vključuje na vse prikaze spletnega mesta in tako tudi omogoči urejanje oblike celotnega spletnega mesta samo iz ene datoteke.

Najnovejša razpoložljiva različica teh stilnih predlog je znana pod imenom CSS 3, ker pa ni podprta še v vseh spletnih brskalnikih, se še vedno uporablja večina ukazov ter pristopov iz predhodne različice CSS 2.

2.6.3.3 DOM

DOM oziroma Document Object Model je standard, ki je neodvisen od platforme in programskega jezika, namenjen je predstavitvi in interakciji z elementi dokumenta HTML in XML. Za vse elemente dokumenta HTML oziroma XML definira objekte, njihove lastnosti in metode za njihovo lažjo manipulacijo, ki jih nato daje na razpolago programskemu jeziku oziroma razvijalcem. Poenostavljeno povedano je DOM nekakšen programski vmesnik, ki za HTML oziroma XML dokument zgradi drevesno strukturo in nato omogoča dostop, spreminjanje, brisanje in kreiranje njenih vozlišč. Implementiran je v vseh spletnih brskalnikih, kar programskemu jeziku JavaScript pravzaprav omogoča dinamično spremljanje in spreminjanje vsebin. Najdemo pa ga tudi v vseh bolj razširjenih programskih jezikih kot sredstvo za interpretacijo in manipulacijo z dokumenti XML.

2.6.3.4 HTTP

HTTP oziroma HyperText Transfer Protocol je standardni komunikacijski protokol za komunikacijo med odjemalcem in strežnikom na svetovnem spletu. Predpisuje, kako mora biti zahtevek oziroma odgovor formiran in poslan ter kakšne akcije morata strežnik in odjemalec izvesti ob določenih zahtevkih in odgovorih.

Prve različice HTTP-ja so ta protokol zaznamovale kot protokol brez trajne povezave, saj so predpisovale, da se takoj, ko odjemalec odda zahtevek in od strežnika prejme odgovor, povezava med strežnikom in odjemalcem poruši. Ta pristop pa je, poleg manjše obremenitve strežnika, kot negativno posledico prinašal večje količine prenosa podatkov. Zadnja različica tega protokola, natančneje HTTP 1.1, to pomanjkljivost uspešno odpravlja, tako da se v novi različici povezava po oddanem zahtevku in prejetem odgovoru ohrani še nekaj časa, v primeru, da bo odjemalec oddal še kakšen zahtevek. Ne glede na izboljšavo trajnosti povezave pa je protokol HTTP še vedno protokol brez stanj (*angl. stateless*). HTTP namreč ne obdrži podatkov posameznega odjemalca čez več povezav, kar onemogoča neposredno sledenje posameznim uporabnikom. Prav zaradi tega morajo spletni razvijalci za zagotavljanje sledenja uporabnikov uporabljati druge pristope, kot so na primer nastavljanje piškotkov ter sej.

Z naraščanjem potrebe po večji varnosti na spletu, na primer pri spletnem plačevanju, se je razvil tudi protokol HTTPS oziroma HyperText Transfer Protocol Secure. Le-ta je v bistvu zavarovana različica protokola HTTP, ki uporablja protokol SSL/TLS za kriptiranje prometa med strežnikom in odjemalcem, kar vmesnim opazovalcem onemogoča krajo uporabnikovih zasebnih podatkov.

2.6.3.5 URI

URI je okrajšava za angleški izraz Uniform Resource Identifier, ki označuje niz znakov za identifikacijo določenega internetnega vira. Obstaja več tipov URI-ja, med katerimi pa je vsekakor najbolj poznan URL oziroma Uniform Resource Locator, ki na svetovnem spletu določa lokacijo spletne strani ter njenih datotek.

Vsak URI je sestavljen iz treh glavnih delov, in sicer komunikacijskega protokola, naslova strežnika ter imena ciljne datoteke oziroma poti do nje. Opcijsko pa ga lahko sestavljajo še uporabniško ime in geslo uporabnika, vrata strežnika, parametri ciljne datoteke ter sidro.

2.7 Kritike pojma Web 2.0

S prodorom pojma Web 2.0 se je pojavilo tudi veliko kritik, ki nasprotujejo rabi tega izraza, češ da je neprimeren oziroma da splet druge generacije sploh še ne obstaja. Najpogosteje se ta pojem označuje kar kot modno besedo (*angl. buzzword*). Izraz Web 2.0 je na tak način komentiral tudi izumitelj svetovnega spleta Tim Berners-Lee z naslednjimi besedami: »Web 2.0 je le žargonski izraz, saj nihče niti natančno ne ve, kaj pomeni ta pojem. Če naj bi Web 2.0 predstavljal skupek blogov in wikijev, je to torej komuniciranje med ljudmi, temu pa je bil splet že od vsega začetka namenjen.«

Najbolj sporni vidik, ki spletu druge generacije otežuje uveljavitev, je njegova uporabljena tehnologija. Kot že omenjeno, je spletna tehnologija, razen manjših nadgradenj, ostala nespremenjena. Večina meni, da si samo zaradi novega načina kombinacije te tehnologije oziroma novega načina izrabe te tehnologije splet nove različice ne zasluži.

Druga različica svetovnega spleta je sporna tudi z vidika časa pojavitve njegovih glavnih značilnosti, natančneje koncepta skupnega ustvarjanja vsebin in koncepta povezovanja funkcionalnosti oziroma podatkov. Prve implementacije teh dveh značilnosti namreč segajo še v splet prve generacije. Na primer, spletna trgovina Amazon je vse od njene pojavitve na spletu, leta 1995, svojim uporabnikom omogočala pisanje pregledov (*angl. reviews*) ter uporabniških vodičev za vse prodajne artikle. Torej je bilo skupno ustvarjanje vsebine na spletu omogočeno že precej časa pred pojavitvijo izraza Web 2.0. Pa tudi spletne storitve, ki omogočajo implementacijo konceptov SOA oziroma koncepta povezovanja funkcionalnosti ter podatkov, so že precej časa prisotne na svetovnem spletu. Leta 2002 je namreč spletna trgovina Amazon ponudila spletnim razvijalcem na razpolago skupek spletnih storitev, znanih pod imenom Amazon web services, ki so temeljili na protokolu SOAP. Še pred tem pa je tudi podjetje Google spletnim razvijalcem odprlo dostop do naprednih API-jev, ki so ravno tako kot danes omogočali enostavno komunikacijo z ostalimi aplikacijami.

Pojavile pa so se tudi kritike v drugi smeri, ki se ne nanašajo na pomen izraza, ampak bolj na slabosti, ki jih prinaša ta evolucija spleta. S tega vidika je zanimivo kritiko predstavil Andrew Keen v njegovi knjigi z naslovom »Cult of the Amateur«. V njej izpostavlja, da splet druge generacije izpodriva strokovno znanje in izkušnje ter da na tak način uničuje kulturo in napada svetovno ekonomijo. Omenja, da današnji splet namesto h kreiranju mojstrovin spodbuja h kreiranju neskončnega digitalnega gozda povprečnosti, kar se kaže kot velika količina neprimernih domačih videoposnetkov, sramujoče amaterske glasbe ter neberljivih esejev oziroma novel. Tako je kritiziral tudi spletno enciklopedijo Wikipedijo, češ da je polna napak, nesporazumov ter neresnice.

2.8 Prihodnost oziroma Web 3.0

Z uveljavitvijo spleta druge generacije so se začela pojavljati tudi ugibanja, kaj nam prinaša prihodnost na področju spleta oziroma kaj bo Web 3.0. Predstavljenih je bilo že mnogo idej oziroma komponent prihajajočega spleta, kot so na primer semantični splet, personifikacija spletnih vsebin, vpeljava umetne inteligence, vpeljava še večje povezanosti podatkov in funkcionalnosti med aplikacijami, integracija televizije v svetovni splet, vpeljava 3D grafike v spletne aplikacije, itd. Svojo napoved je izrazilo tudi podjetje Yahoo in to z enačbo, ki je prikazana spodaj.

$$Web\ 3.0 = 4C + P + VS \qquad (1)$$

Enačba izraža, da bo splet tretje generacije zgrajen na podlagi:

- **4C**: vsebine (*angl. Content*), trgovanja (*angl. Commerce*), skupnosti (*angl. Community*), konteksta (*angl. Context*)
- **P**: personifikacije (*angl. Personalization*)
- **VS**: vertikalnega iskanja (*angl. Vertical Search*)

Med vsemi poskusi napovedovanja prihodnosti spleta pa se za najbolj obetajočo komponento prihajajočega spleta izpostavlja semantični splet, pogosto omenjen tudi kot splet podatkov. Ta ne predstavlja novega ločenega spleta, ampak le logično nadgradnjo že obstoječega. Stremi k izboljšavi trenutne različice spleta na tak način, da bodo lahko sistemi vse podatke na spletu interpretirali, povezovali, procesirali in tako končnemu uporabniku posredovali točno take informacije, kot jih želi.

S praktičnega pogleda pa je semantični splet skupek tehnoloških standardov (URI, UNICODE, XML, RDF, RDFS, OWL, …), ki aplikacijam oziroma napravam omogoča razumeti semantične dokumente ter podatke. S pomočjo standardov, kot je na primer RDF, se namreč na spletu lahko objavijo logični stavki (metapodatki), ki jih bodo inteligentne naprave (agenti) prebrale, jih razumele (s pomočjo ontologije[10]) ter nato primerjale in povezale z ostalimi podatki na spletu. Na podlagi teh akcij lahko inteligentne naprave pridobijo celotno, jasno sliko podatkov in njihovih povezav ter tako omogočijo nove agregacije podatkov, ki doslej niso bile mogoče. Uporabnikom se na primer tako omogoči napredno iskanje želenih informacij ter korelacij med njimi, ne glede na to, kje na spletu se iskani podatki nahajajo (primer: iskanje avtomobilov, oglaševanih po celotnem svetovnem spletu, ki ustrezajo znamki Ford s ceno 7000 € in letom izdelave 2010). Ker spletne aplikacije natančno razumejo vse podatke na spletu in njihove odvisnosti, se lahko za vsakega uporabnika izvede tudi natančno personifikacijo vsebin. Semantični splet namreč lahko z uporabo omenjenih pristopov vsakega uporabnika posebej prepozna, ugotovi, s čim se ukvarja, kaj ga veseli, kakšna je njegova trenutna lokacija in na podlagi teh podatkov ponudi vsebine, ki so za uporabnika trenutno aktualne.

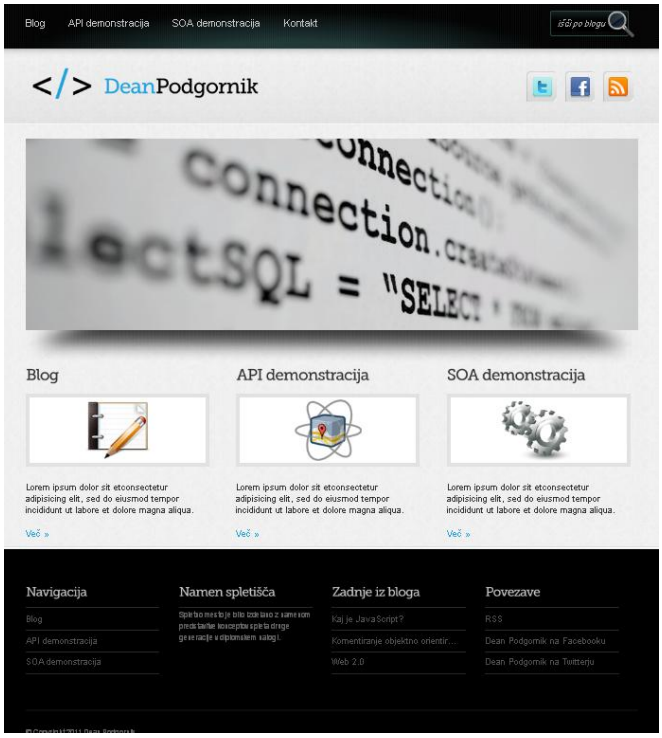
V spletu druge generacije je ta pristop semantičnega spleta še onemogočen, saj je trenutni splet kljub raznim razpoložljivim API-jem še vedno le spletišče datotek ter aplikacij, ki centralizirane podatke hranijo in uporabljajo le za svoje potrebe. Podatki bi namreč morali biti

[10] Ontologije so slovarji znanja o določenih področjih. Vsebujejo definicije pojmov oziroma natančneje razrede, lastnosti ter odnose med njimi.

porazdeljeni, razdrobljeni in dostopni vsepovsod po spletu, tako da bi jih lahko inteligentne naprave našle, analizirale oziroma interpretirale ter procesirale.

3 DEMONSTRACIJA KONCEPTOV SPLETA DRUGE GENERACIJE NA MANJŠI SPLETNI APLIKACIJI

Za utrditev teoretičnega dela diplomske naloge sem izdelal manjše spletno mesto oziroma natančneje manjšo spletno aplikacijo, pri izgradnji katere sem upošteval večino temeljnih konceptov spleta druge generacije. Spletna aplikacija, katere vstopna stran je prikazana na sliki 15, je na svetovnem spletu dostopna preko URL naslova: http://www.deanpodgornik.si.



Slika 15: Vstopna stran izdelane spletne aplikacije za demonstracijo konceptov spleta druge generacije.

Ker sem se v aplikaciji želel posvetiti izgradnji le temeljnih vidikov spleta druge generacije, sem oblikovanje ter grajenje postavitve strani z uporabo označevalnega jezika HTML in stilnih predlog CSS preskočil oziroma sem uporabil dve že izdelani predlogi. Predlogi sem prevzel iz dveh spletišč, ki ponujata brezplačne že izdelane predloge, in sicer na URL naslovih http://www.site5.com/wordpress-themes/ ter http://jump2top.com/themes/. Predlogi sem skrbno izbral, tako da sta kljub njuni brezplačnosti zelo kvalitetni in v skladu s trenutnimi oblikovnimi trendi spleta druge generacije, kot so na primer: visoke noge spletnih strani, nežni prehodi med barvami (*angl. gradient*), umirjene in nežne barve ozadij, zaobljeni robovi, uporaba ikon, senčenje elementov, itd. Obe predlogi pa sta tudi dobro zgrajeni. Natančneje, koda uporabljenih tehnologij XHTML 1.0 in CSS 2 je v skladu z zahtevami inštituta W3C, njihove strani se pravilno prikažejo v vseh trenutno aktualnih spletnih brskalnikih.

Pri izgradnji spletne aplikacije sem za strežniško tehnologijo uporabil skriptni programski jezik PHP, saj se je ta v kombinaciji z mojim predhodnim programerskim znanjem in veliko količino razpoložljive podpore velikokrat izkazal kot odlična tehnologija za enostavno in hitro izgradnjo tako manjših kot tudi večjih spletnih mest oziroma spletnih aplikacij. Tako navadno vsebino kot tudi podatke, potrebne za zagotavljanje prilagodljivosti ter dinamičnosti prikaza, sem shranjeval v podatkovno bazo MySQL, ki ima ravno tako kot programski jezik PHP

obilo podpore pri reševanju raznih programerskih težav. Za dodatno poenostavitev izgradnje spletne aplikacije sem uporabil še CMS sistem Myportal, ki ga z vizijo enostavnega in močnega sistema že vrsto let razvija slovensko podjetje Editor d.o.o. S tem sistemom sem se izognil grajenju uporabniškega vmesnika za urejanje vsebin in si poenostavil upravljanje s podatkovno bazo. Tako sem se lahko bolj osredotočil na implementacijo funkcionalnosti in značilnosti spleta druge generacije.

Za zagotavljanje bogate uporabniške izkušnje v uporabnikovem spletnem brskalniku sem za tehnologijo odjemalčeve strani uporabil skriptni programski jezik JavaScript, na njem temelječe ogrodje Mootools in tehnologijo AJAX. Tehnologiji Adobe Flash ter ostalim tehnologijam, ki temeljijo na vtičih spletnega brskalnika, sem se raje odrekel, saj bi v primeru moje manjše spletne aplikacije te tehnologije le zavlačevale in oteževale razvoj ter upočasnile delovanje same aplikacije. K tej odločitvi je pripomoglo tudi dejstvo, da omenjena spletna aplikacija ne vsebuje funkcionalnosti, ki se jih ne bi uspelo realizirati že z uporabo enostavnega programskega jezika JavaScript, v nasprotnem primeru bi bila potrebna uporaba močnejše tehnologije, kot je na primer Adobe Flash.

Z uporabo omenjene tehnologije strežniške in odjemalčeve strani sem v aplikacijo vgradil vse tri temeljne vidike spleta druge generacije oziroma implementiral glavne značilnosti socialnega spleta, odpravil večino klasičnega potratnega osveževanja strani v skladu s koncepti RIA ter demonstriral koncepte SOA z uporabo spletnih storitev. Na aplikaciji pa sem prikazal tudi ostale manj pomembne, a zato nič manj razširjene značilnosti trenutne različice spleta, in sicer: uporabo značk, vizualizacijo oblaka značk, omogočanje iskalnika po vsebini, omogočanje uporabe spletnega vira ter uporabo API-jev. Za poglobljeno demonstracijo teh konceptov ter značilnosti spleta druge generacije sem v spletno mesto vključil tudi eno izmed pogostejših oblik novodobnih spletišč, natančneje blog, in prav na njem prikazal večino omenjenih aspektov novega spleta.

3.1 Povezava s CMS sistemom Myportal

Kot sem že omenil, mi je CMS sistem Myportal prihranil veliko zamudnega dela pri izgradnji celotne spletne aplikacije. Med vsemi njegovimi uporabnimi funkcionalnostmi bi še posebej izpostavil modul za nastavitve prikaza, s pomočjo katerega lahko tudi nevešči spletni razvijalci enostavno in hitro zgradijo celotno strukturo prikazov spletnega mesta ter tako tudi definirajo, katera PHP skripta se na določenem prikazu izvede. Razvoj spletne aplikacije sem začel ravno s konfiguracijo tega modula - zamisel delovanja spletne aplikacije sem razdrobil na več majhnih delov oziroma prikazov ter za vsak prikaz določil ime prikaza ter PHP skripto, ki naj se na njej izvede. S to dokončano okvirno strukturo prikazov sem prikaze oblikoval še v skladu s pridobljenimi zastonjskimi predlogami in nato prešel na definiranje strukture vsebin, kot je na primer določanje tipa vnosnih polj za objave bloga (primer vnosnih polj: ime objave, datum, opis, tekst, avtor, …). Uporabniškega vmesnika za urejanje teh vsebin ni bilo potrebno konfigurirati, saj se ta samodejno zgradi glede na definirano strukturo vsebin.

Po opredelitvi strukture sem lahko prešel na kodiranje ter povezovanje podatkov iz podatkovne baze s posameznimi prikazi. V PHP skripti posameznega prikaza sem namreč s pomočjo pred-definiranih Myportal-ovih PHP objektov enostavno določil, katere podatke iz podatkovne baze potrebujem za operiranje oziroma prikazovanje. Nato pa je sistem Myportal sam sestavil in izvedel poizvedbo ter rezultat serviral v tabeli. V primeru zahtevnejših agregacij podatkov oziroma zahtevnejših poizvedb iz podatkovne baze pa sem moral sam

sestaviti poizvedbo SQL in jo nato posredovati Myportal-ovemu pred-definiranemu PHP objektu, ki je poizvedbo izvedel in rezultat ponovno serviral v tabeli. Poleg omenjenih objektov za lajšanje dela s podatkovno bazo, pa mi je sistem Myportal nudil še razne objekte za poenostavitev kodiranja pogostih funkcionalnosti spletnih aplikacij, na primer: pošiljanje elektronske pošte, procesiranje spletnih obrazcev, shranjevanje datotek, beleženje statistike, itd.

3.2 Izgradnja bloga

V spletno aplikacijo sem vključil tudi eno izmed pogostejših oblik novodobnih spletišč, natančneje blog, na katerem sem prikazal večino konceptov oziroma značilnosti spleta druge generacije. Blog je v izdelani spletni aplikaciji dostopen na URL naslovu http://www.deanpodgornik.si/blog/. Pri njegovi izgradnji sem se pretežno zgledoval po funkcionalnostih, ki jih spletišče blogger.com ponuja svojim uporabnikom oziroma blogerjem, in zato vanj tudi implementiral: uporabo značk, vizualizacijo oblaka značk, arhiv objav bloga, kategoriziranih po mesecih, prijavo v skupino spremljevalcev bloga, komentiranje objav ter deljenje posamezne objave bloga preko socialnih omrežij. Na vstopnem prikazu bloga sem navedel seznam vseh objav, kronološko urejenih od najnovejše do najstarejše. Za vsako objavo posebej sem na tem seznamu prikazal naslov objave, datum kreiranja, celoten tekst oziroma vsebino, značke, s katerimi je objava bloga označena, ter število komentarjev za to objavo. Uporabnik lahko celoten blog prebira samo iz tega prikaza, brez kakršnegakoli dodatnega brskanja. Če pa želi pregledovati komentarje oziroma želi tudi sam oddati svoj komentar pod kakšno objavo, mora preko jasno podane povezave najprej vstopiti na prikaz, ki je namenjen izpisu samo določene objave bloga. Na tem prikazu je namreč poleg že vseh izpostavljenih podatkov na prejšnjem seznamu objav omogočena še opcija branja komentarjev oziroma dodajanje le-teh ter tudi možnost deljenja objave preko socialnih omrežij. Torej, zgrajeni blog je v skladu s trendi novodobnih spletišč, saj omogoča večino funkcionalnosti oziroma značilnosti spleta druge generacije, dostopanje do želenih informacij pa je omogočeno na hiter, enostaven in predvsem uporabniku prijazen način.

3.3 Implementacija konceptov socialnega spleta

Kot enega izmed glavnih elementov spleta druge generacije sem v izdelano spletno aplikacijo vgradil tudi socialni splet. Z uporabo njegovih konceptov, ki temeljijo na komunikacijskih in socialnih trendih, kot je na primer integracija socialnega omrežja Facebook v spletno aplikacijo, sem uporabnike privabil in pozval, da opustijo klasično brskanje ter postanejo aktivni soustvarjalci vsebine v spletni aplikaciji. Kot elemente socialnega spleta sem v spletno aplikacijo implementiral grajenje uporabniško generirane vsebine ter deljenje želene vsebine preko socialnih omrežij.

Za omogočanje grajenja uporabniško generirane vsebine sem v spletno aplikacijo vgradil zmožnost komentiranja objav bloga. Na koncu vsake objave bloga sem namreč izpisal vse že oddane komentarje in pod njimi prikazal še vnosno polje za oddajo novega komentarja. Za vsak oddan komentar sem omogočil, da se v podatkovno bazo poleg teksta shrani še datum oddaje komentarja ter identifikacijska številka uporabnika. Seveda pa sem moral za zagotovitev te funkcionalnosti omogočiti še uporabniške račune uporabnikov, ki bodo služili kot referenca za vso ustvarjeno vsebino posameznega uporabnika. Za omogočanje kreiranja teh uporabniških računov sem izdelal prav temu namenjen prikaz, kjer se lahko vsak

uporabnik aplikacije registrira oziroma ustvari svoj uporabniški račun. V skladu z dobrimi praksami novodobnih spletišč sem vključil še funkcionalnost povrnitve gesla, če ga je uporabnik pozabil, ter možnost spreminjanja uporabniških podatkov. Pri razvoju celotnega sklopa funkcionalnosti upravljanja z uporabniškim računom sem si zastavil cilj, da uporabniku omogočim čim hitrejši ter njemu čim bolj intuitiven način izvajanja teh operacij, za kar menim, da mi je tudi uspelo. V skladu z zastavljenim ciljem pa sem implementiral še dodatno opcijo prijave uporabnika v spletno aplikacijo, natančneje zmožnost prijave uporabnika z uporabniškim računom socialnega omrežja Facebook. Za to funkcionalnost sem uporabil API z imenom Facebook Connect, s pomočjo katerega sem novim uporabnikom moje aplikacije omogočil, da lahko preskočijo korak ustvarjanja novega uporabniškega računa za to spletno aplikacijo in takoj začnejo uporabljati funkcionalnosti spletne aplikacije, ki zahtevajo posedovanje uporabniškega računa. JavaScript programska koda, ki omogoča uporabo API-ja Facebook Connect oziroma prijavo uporabnika v spletno aplikacijo z uporabniškim računom socialnega omrežja Facebook, je priložena v prilogi A.

Pri omogočanju funkcionalnosti deljenja vsebin preko socialnih omrežij, sem se posluževal API-jev socialnih omrežij Twitter in Facebook. Obe socialni omrežji ponujata precej enostavno implementacijo tovrstnih API-jev, potrebna je le vključitev določene HTML oziroma JavaScript kode ter navedba parametrov o informacijah spletišča ter želenih nastavitvah. Zahtevana HTML koda za vključitev API-ja, ki omogoča deljenje vsebin preko socialnega omrežja Twitter, je prikazana na sliki 16.

```
<a href="http://twitter.com/share?url=http://www.deanpodgornik.si/blog/web-
20" class="twitter-share-button" data-count="horizontal">Tweet</a>
<script type="text/javascript"
      src="http://platform.twitter.com/widgets.js"></script>
```

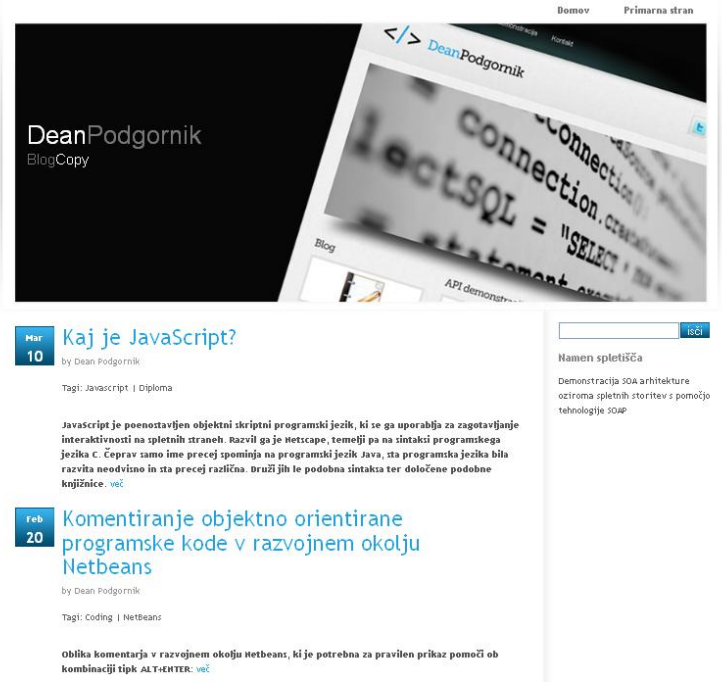
Slika 16: HTML koda za omogočanje deljenja vsebin preko socialnega omrežja Twitter.

Menim, da sem spletno aplikacijo bistveno izboljšal z vključitvijo omenjenih konceptov socialnega spleta. Sedaj ta ni več le baza podatkov oziroma orodje za izvajanje operacij, ampak mesto, kjer lahko uporabniki izrazijo svoje mnenje, aplikacijo vsebinsko dopolnjujejo, izpopolnjujejo in širijo njeno prepoznavnost. Aplikacija je s temi vpeljanimi koncepti postala uporabnikom veliko bolj prijazna – sedaj lahko občutijo pripadnost skupnosti te aplikacije ter posledično tudi več prispevajo k namenu same aplikacije.

3.4 Implementacija SOA

Na podlagi SOA, predstavljene v teoretičnem delu diplomske naloge, sem glavni koncept te arhitekture demonstriral s pomočjo spletnih storitev na izdelavi ponudnika (strežnik) in uporabnika storitve (odjemalec). Ponudnika storitve sem integriral v že predstavljeno spletno aplikacijo na URL naslovu http://www.deanpodgornik.si/, medtem ko sem za bolj jasen način demonstracije uporabnika storitve izdelal še eno manjšo spletno aplikacijo, ki na URL naslovu http://blog2.deanpodgornik.si/ služi samo za demonstracijo tega uporabnika storitve. Z integracijo tega ponudnika storitve v spletno aplikacijo sem v bistvu to aplikacijo odprl navzven oziroma omogočil, da lahko njene storitve koristijo katerekoli zunanje aplikacije. Kot ponujeno storitev ponudnika storitve sem implementiral opcijo iskanja in pridobivanja vsebin iz bloga, ki je dostopen na URL naslovu http://www.deanpodgornik.si/blog. Uporabniku storitve sem s to storitvijo natančneje omogočil, da lahko celoten blog integrira na svoje

spletno mesto in na njem omogoča tako branje objav kot tudi iskanje želenih objav po celotni vsebini bloga. Izdelana spletna aplikacija uporabnika storitve je z omenjeno funkcionalnostjo prikazana na sliki 17.



Slika 17: Izdelana spletna aplikacija za demonstracijo SOA.

Omeniti pa moram, da spletni aplikaciji nisem v celotni zgradil po priporočilih SOA, logike izvajanja teh dveh spletnih aplikacij namreč nisem v celoti porazdelil po storitvah. Določene operacije, kot je na primer prijava uporabnika na seznam spremljevalcev bloga, sem raje implementiral v ne-storitveno usmerjenem arhitekturnem načinu, saj bi sicer pristop SOA, v primeru moje manjše spletne aplikacije, le upočasnil razvoj in ne bi bistveno prispeval h kasnejšemu enostavnejšemu spreminjanju funkcionalnosti aplikacije. Zmanjšala pa bi se tudi odzivnost aplikacije, saj SOA pristopi ne veljajo za ene izmed najhitrejših načinov zagotavljanja komunikacije med storitvami oziroma med strežnikom in odjemalcem.

Za implementacijo ponudnika storitve kot tudi uporabnika storitve sem uporabil brezplačno orodje NuSOAP, ki je v bistvu le skupek PHP razredov za enostavnejše kreiranje in uporabo spletnih storitev, ki temeljijo na tehnologiji SOAP, WSDL in HTTP. Razvijalcem to orodje natančneje omogoča, da se z uporabo raznih NuSOAP PHP funkcij izognejo neposrednemu grajenju WSDL dokumenta oziroma neposrednemu grajenju ter pošiljanju SOAP sporočil in se tako lahko bolj posvetijo implementaciji raznih funkcionalnosti spletnih storitev.

Razvijanje spletne storitve sem s pomočjo orodja NuSOAP začel pri vzpostavljanju WSDL dokumenta, ki sem ga javno objavil na URL naslovu http://www.deanpodgornik.si/sys/soap_server?wsdl. Končna različica tega dokumenta pa je priložena v prilogi B. Za implementacijo WSDL dokumenta sem se odločil predvsem zato, da bodo lahko tudi ostali razvijalci veliko lažje implementirali razpoložljivo storitev. V tem WSDL dokumentu sem na enostaven in jasen način podal:

- URL naslov storitve,
- ime razpoložljive metode v storitvi, ki se glasi »izvedi_iskanje«,

- podatkovni tip vhodnih parametrov metode »izvedi_iskanje«, ki je natančneje novo definirana podatkovna struktura z imenom »struktura_iskalnih_parametrov«,
- podatkovni tip izhoda metode »izvedi_iskanje«, ki je v bistvu kar tabela,
- dokumentacijo, ki je v pomoč razvijalcem pri uporabi te storitve.

Z dokončano implementacijo tega WSDL dokumenta sem tudi odpravil edini izziv, ki sem ga imel pri grajenju tega ponudnika storitve. Implementacijo mehanizmov prejemanja zahtevkov, pošiljanja odgovorov ter seveda grajenje in interpretacijo SOAP sporočil mi je namreč zagotovilo kar orodje NuSOAP. Po dokončani vzpostavitvi WSDL podpore mi je preostala le še izgradnja logike izvajanja metode »izvedi_storitev«, kar pa se v bistvu izraža le kot nekaj vrstic programske kode za pridobitev podatkov iz podatkovne baze.

Po uspešno dokončanem kodiranju programske kode ponudnika storitve sem prešel še na grajenje uporabnika storitve. Pri njegovi implementaciji mi je orodje NuSOAP ponovno olajšalo veliko programerskega dela. Funkcijam tega orodja sem namreč podal le naslov strežnika oziroma ponudnika storitve, ime želene metode ter zahtevane parametre le-te, za vso ostalo komunikacijo oziroma interpretacijo pa je poskrbelo kar orodje NuSOAP. Ko je bil zagotovljen osnovni način komunikacije, pa sem šel še korak dlje, in sicer namesto URL naslova ponudnika storitve sem podal kar URL naslov WSDL dokumenta te storitve. Tako sem namreč omogočil, da ta aplikacija uporabnika storitve iz WSDL dokumenta sama razbere URL naslov ponudnika storitve ter tudi podatkovne tipe vhodnih parametrov. Posledično spletna aplikacija sama preveri podane parametre ter se ob njihovi ustreznosti tudi sama poveže na strežnik storitve. Ta dodatno uporabljena programska interpretacija WSDL dokumenta se v praksi kar pogosto uporablja, saj se izkaže za precej dober pristop pri uporabi pogosto funkcionalno spreminjajočih se spletnih storitvah.

PHP programsko kodo, ki omogoča izvajanje zgoraj omenjenega ponudnika storitve, sem priložil v prilogi C, medtem ko sem PHP programsko kodo za poganjanje uporabnika storitve priložil v prilogi D.

3.5 Implementacija RIA konceptov

Pri izdelavi spletne aplikacije sem kot ene izmed glavnih konceptov, potrebnih za pridobitev naziva spletne aplikacije spleta druge generacije, uporabil tudi koncepte RIA. Za omogočanje bogate uporabniške izkušnje, ki je, kot že večkrat omenjeno, poglavitna lastnost v RIA aplikacijah, sem uporabil tehnologijo AJAX, saj menim, da je ta glede na kompleksnost moje spletne aplikacije še najbolj primerna. V spletni aplikaciji sem to tehnologijo oziroma asinhrono izvajanje poskušal izrabiti na čim več operacijah, sem pa kljub temu velik delež operacij zaradi njihove kompleksnosti raje izvedel na klasičen način osveževanja celotne strani. Z AJAX-om sem zato implementiral le komentiranje objav bloga, prijavo elektronskega naslova v skupino spremljevalcev bloga in prijavo uporabnika v spletno aplikacijo.

Uporabljeni koncepti RIA aplikacij so pri vseh naštetih operacijah pretežno enaki, tako da se bom v tem poglavju osredotočil le na operacijo komentiranja objav bloga. Za omogočanje komentiranja sem na dnu prikaza, namenjenega izpisu posamezne objave bloga, dodal vnosno polje, natančneje HTML element textarea, sliko trenutno prijavljenega uporabnika, skrito identifikacijsko številko tega uporabnika ter gumb za potrditev. Temu skupku HTML elementov sem še omogočil, da se prikaže le, če je uporabnik prijavljen, saj se tako onemogoči komentiranje anonimnim uporabnikom. Za začetek implementacije operacije

komentiranja sem v JavaScript programski kodi definiral vse HTML elemente in dogodke, ki so potrebni za izvajanje omenjene operacije. V naslednjem koraku sem spisal še PHP programsko kodo, ki podane parametre, natančneje identifikacijsko številko uporabnika in komentar, shrani v podatkovno bazo. Po postavljeni osnovi sem lahko prešel na implementacijo AJAX-a. S pomočjo JavaScript ogrodja Mootools, ki poleg njegovih številnih zmogljivosti omogoča tudi podporo AJAX-u, sem deklariral objekt za upravljanje s tehnologijo AJAX-a ter mu podal sledeče parametre: URL naslov PHP skripte oziroma prikaza za shranjevanje komentarja, metodo pošiljanja podatkov (post), podatke oziroma parametre (identifikacijsko številko uporabnika ter komentar) za PHP skripto in referenco na HTML element, v katerem naj se izpiše rezultat AJAX-a. Po tej deklaraciji je AJAX ter tako tudi operacija komentiranja že začela delovati in uspešen oziroma neuspešen rezultat AJAX-a se je tudi že pravilno izpisal v predvideni HTML element. Za še bolj jasno obvestilo o uspešni operaciji oddaje komentarja sem dodal še funkcionalnost, da se novo dodani komentar takoj vizualizira. Ob uspešno dokončani operaciji oddaje komentarja sem namreč na enak način vzpostavil še en AJAX, ki za zadnjim prikazanim komentarjem prikaže novo oddani komentar. JavaScript programska koda, ki omogoča to operacijo komentiranja, se nahaja v prilogi E.

Menim, da sem z vključitvijo AJAX-a mojo spletno aplikacijo bistveno izboljšal, predvsem z vidika interaktivnosti. Uporabniku sedaj ni več potrebno za vsako majhno operacijo osveževati celotnega prikaza, kar mu seveda omogoča veliko lažje sledenje izvajajočih se operacij. Moram pa omeniti, da moja manjša spletna aplikacija z vpeljavo konceptov RIA ni kaj bistveno bolj podobna klasičnim namiznim aplikacijam z vidika interaktivnosti. Precej operacij sem namreč realiziral na klasičen način osveževanja celotne strani, tako da je meja s klasičnimi namiznimi aplikacijami ostala skoraj nespremenjena. Za odpravo te meje bi moral za večino prikazov omogočiti nalaganje z AJAX-om, za kar pa menim, da ni najbolj optimalna rešitev v primeru moje spletne aplikacije, saj se tako izgubi sledenje zgodovini brskanja. Kljub še vedno jasno določeni meji s klasičnimi namiznimi aplikacijami pa je tehnologija AJAX v moji manjši spletni aplikaciji vsekakor ustvarila velik napredek pri povečanju interaktivnosti oziroma omogočanju bolj prijaznega uporabniškega vmesnika, tako da je bila implementacija konceptov RIA vsekakor vredna vloženega truda.

3.6 Implementacija značk

V izdelano spletno aplikacijo, natančneje blog, sem vgradil tudi uporabo značk in tako končnemu uporabniku omogočil hitrejše iskanje želene objave bloga in hitrejše razpoznavanje oziroma kategoriziranje vsebine določene objave. Pri implementaciji te značilnosti spleta druge generacije sem se odločil za uporabo omejenega nabora značk, natančneje dodeljevanja značk objavam bloga iz končne množice značk. Menim, da je ta pristop bolj optimalen oziroma bolj obvladljiv, pa tudi enostavnejši za realizacijo v primerjavi z neomejenim naborom značk, kjer se posamezni objavi lahko dodeli katerokoli značko. Kljub temu pa sem omogočil še možnost, da lahko administrator razširi nabor značk z novimi značkami, saj sicer lahko ta hitro postane neustrezen ali pa zastarel. Za implementacijo folksonomije oziroma omogočanja uporabniško generiranih značk se nisem odločil, saj je blog precej majhen in ga lahko administrator kar sam obvladuje. Sem se pa odločil za implementacijo grafične upodobitve značk oziroma oblaka značk, ki za prikazovanje gostote pojavitev značk uporablja različne velikosti pisave. Za njegovo izgradnjo sem uporabil enačbo, ki je prikazana spodaj.

$$velikost\_pisave = \left\lceil \frac{velikost\_pisave_{max} \times (st\_pojavitev - st\_pojavitev_{min})}{st\_pojavitev_{max} - st\_pojavitev_{min}} \right\rceil \text{če je } st\_pojavitev > 0 \qquad (2)$$

Prikazana enačba na podlagi maksimalnega ($st\_pojavitev_{max}$) in minimalnega ($st\_pojavitev_{min}$) števila pojavitev vseh značk, maksimalne velikosti pisave ($velikost\_pisave_{max}$) ter števila pojavitev (st_pojavitev) posamezne značke vrne velikost pisave (velikost_pisave) za posamezno značko, če ima ta značka vsaj eno pojavitev (st_pojavitev > 0). Izračunano velikost pisave posamezne značke sem enostavno s pomočjo programskega jezika PHP posredoval stilnemu atributu določenega HTML elementa, ki je nato vsebujočo značko tudi prikazal v določeni velikosti. Vsem prikazanim značkam v oblaku značk sem pripel tudi povezavo na prikaz, kjer se izpiše seznam objav, označenih z izbrano značko, kar uporabniku posledično omogoča, da lahko na enostaven in hiter način prebira le objave, ki ga zanimajo. Menim, da sem z vključitvijo tega oblaka značk blog precej izboljšal z vidika interakcije uporabnika, saj ta lahko sedaj na veliko hitrejši in enostavnejši način išče in prebira želene vsebine.

3.7 Implementacija iskalnika po vsebini

Kot vsako spletno mesto spleta druge generacije, ki razpolaga z nekoliko več vsebine, ima tudi moja manjša spletna aplikacija vgrajen iskalnik po vsebini. Ta v mojem primeru služi za natančno iskanje želene vsebine po vseh objavah bloga. Rezultate iskanja, tako rekoč zadetke, prikazuje v isti obliki kot običajen spisek objav, tako da lahko uporabnik želene zadetke oziroma objave prebira kar na prikazu, namenjenemu izpisu zadetkov iskanja, brez kakršnegakoli dodatnega brskanja. Poskrbel sem tudi, da so rezultati iskanja vedno ažurni, zato se ti vsakič ponovno dinamično generirajo iz zapisov, ki so shranjeni v podatkovni bazi, na podlagi iskanega niza vsebovanega v tekstovnih poljih imena objave, opisa, teksta oziroma vsebine ter značk, s katerimi je objava označena. Stavek SQL, ki to poizvedbo v podatkovni bazi izvede, je prikazan na sliki 18.

```
SELECT articles.*,articles_categories.name AS kategorija
FROM articles
LEFT JOIN articles_categories ON (
      articles.cid=articles_categories.cid AND
      articles_categories.lang = articles.lang)
WHERE articles.cid='1' AND articles_categories.published=1 AND
      articles.published=1 AND articles.lang='si' AND(
            articles.name LIKE '%$GET[iskana-beseda]%' OR
            articles.description LIKE '%$GET[iskana-beseda]%' OR
            articles.text LIKE '%$GET[iskana-beseda]%' OR
            articles.tags LIKE '%$GET[iskana-beseda]%')
```

Slika 18: Stavek SQL za omogočanje iskanja želene vsebine po objavah bloga.

Prikazani stavek SQL, tako rekoč poizvedba, vrne vse podatke oziroma vsa polja člankov (articles), ki se nahajajo v kategoriji člankov (articles_categories), namenjeni objavam bloga. Glavni pogoj, da se članek prikaže med zadetki, pa je vsebovanost iskanega niza, začasno shranjenega v spremenljivki $GET['iskana-beseda'], v kateremkoli podnizu imena objave, opisa, teksta oziroma vsebine ali pa značk, s katerimi je objava označena. Rezultat te poizvedbe, natančneje seznam ustrezajočih objav bloga, sem s pomočjo programskega jezika PHP enostavno shranil v dvodimenzionalno tabelo, iz katere sem nato tudi izpisal vsebino za vsako objavo posebej.

V primeru moje manjše spletne aplikacije je iskalnik precej komplementaren z že predstavljenim oblakom značk, oba namreč uporabniku vrneta ožji izbor objav bloga na podlagi zahtev uporabnika. Kljub temu pa se oblak značk, z njegovim hitrim in uporabniku prijaznim načinom razvrščanja objav bloga, ne more kosati z iskalnikom po vsebini, saj ta lahko zagotavlja mnogo bolj natančnejše rezultate, ki so v večjih spletiščih spleta druge generacije vsekakor nepogrešljivi.

3.8 Implementacija spletnega vira

V namen demonstracije uporabe spletnih virov sem v spletno aplikacijo implementiral tudi spletni vir formata RSS, ki podatke črpa iz istega podatkovnega vira kot blog. Namenjen je tako končnim uporabnikom za hitrejše dostopanje do objav bloga kot pa tudi aplikacijam za enostavnejše razčlenjevanje in pridobivanje želenih vsebin iz bloga. Kot že omenjeno, sem za izgradnjo spletnega vira uporabil format RSS, to pa zgolj zato, ker je ta med razpoložljivimi rešitvami trenutno najbolj razširjen in priljubljen. Implementacija spletnega vira omenjenega formata velja za zelo enostavno opravilo, saj je potrebno le poznavanje označevalnega jezika XML oziroma pravilne sintakse RSS ter dinamičnega načina pridobivanja želenih podatkov iz podatkovne baze. V XML značke, ki jih predvideva format RSS, namreč le vstavimo želene podatke iz podatkovne baze in spletni vir je že pripravljen za njegovo uporabo. Testni primer spletnega vira, ki sem ga vgradil v mojo manjšo spletno aplikacijo, je prikazan na sliki 19.

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0"
  xmlns:content="http://purl.org/rss/1.0/modules/content/"
  xmlns:wfw="http://wellformedweb.org/CommentAPI/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  >
  <channel>
    <title>Blog Deana Podgornika</title>
    <link>http://www.deanpodgornik.si</link>
    <description>Zadnje objave iz bloga Deana Podgornika</description>
    <language>si</language>
    <item>
      <title>Kaj je JavaScript?</title>
      <link>http://www.deanpodgornik.si/blog/kaj-je-javascript</link>
      <description><![CDATA[JavaScript je poenostavljen objektni skriptni
programski jezik.]]></description>
      <content:encoded><![CDATA[Glavni namen JavaScript-a je omogočanje
interakcije z označevalnim jezikom HTML ter posledično poživitev strani z
dinamičnim izvajanjem. Njegov nabor funkcionalnosti tako obsega: krairanje
brskalnikih...]]></content:encoded>
      <guid>http://www.deanpodgornik.si/post/kaj-je-javascript</guid>
      <pubDate>Thu, 10 Mar 2011 23:26:02 +0100</pubDate>
    </item>
    <item>
      <title>Web 2.0</title>
      <link>http://www.deanpodgornik.si/blog/web-20</link>
      <description><![CDATA[Web 2.0 oziroma Splet 2.0 je pojem, ki se
nanaša na drugo generacijo svetovnega spleta.]]></description>
      <content:encoded><![CDATA[Splet si je novo različico prislužil na
podlagi novega koncepta razvoja in novega načina uporabe spletnih
storite...content:encoded>
      <guid>http://www.deanpodgornik.si/post/web-20</guid>
      <pubDate>Sat, 12 Feb 2011 11:59:02 +0100</pubDate>
    </item>
  </channel>
</rss>
```

Slika 19: Primer XML kode za spletni vir formata RSS.

Kot je razvidno iz zgornje slike, sem v spletnem viru s pomočjo označevalnega jezika XML oziroma sintakse RSS na enostaven in jasen način definiral: ime spletnega vira (title), povezavo na izvorno spletno mesto (link), opis spletnega vira (description) in še navedbo jezika (language), v katerem je vsebina napisana. Za vsemi opredeljenimi osnovnimi podatki spletnega vira pa sem vključil še glavni del, in sicer spisek objav, med katerimi je vsaka posamezna objava navedena v svojem XML elementu »item«. Znotraj vsakega XML elementa »item« pa sem še za posamezno objavo podal njeno: ime (title), povezavo na prikaz (link), opis vsebine (description), celotno vsebino (content:encoded), identifikator (guid) ter datum objave (pubDate). Vsi našteti podatki spletnega vira oziroma posameznih objav so v bistvu le najbolj osnovni, obstaja namreč še mnogo drugih, katerih namen pa v mojem kontekstu bloga ni bil primeren, zato sem se njihovi uporabi raje izognil.

Ker se v XML elemente za hranjenje omenjenih podatkov vstavlja vsebino iz običajnih člankov, objavljenih na spletnem mestu, se kar hitro zgodi, da se med besedilom pojavijo znaki <, > ali pa &, ki pa so v označevalnem jeziku XML rezervirani. Z njihovo nepravilno pojavitvijo lahko naredijo XML neveljaven ter posledično onemogočijo pravilno prikazovanje in branje spletnega vira. Tej potencialni grožnji sem se enostavno izognil tako, da sem vse

kritične vsebine obdal z nekoliko drugačno standardno značko CDATA oziroma natančneje z začetnim sklopom znakov <![CDATA[in končnim sklopom znakov]]>. Bralniki spletnih virov namreč to posebno značko zaznajo in omenjene rezervirane znake znotraj določenega območja obravnavajo kot navadne znake, tako da se lahko spletni vir pravilno interpretira.

Za informiranje končnih uporabnikov spletne aplikacije, da jim le-ta daje na razpolago uporabo spletnega vira, sem na vseh prikazih vključil razpoznavni znak spletnega vira RSS in nanj pripel povezavo na URL naslov http://www.deanpodgornik.si/rss, kjer je omenjeni spletni vir tudi dostopen. Na vseh prikazih spletne aplikacije pa sem vključil še poseben HTML element, natančneje meta vrstico z vsebujočim URL naslovom spletnega vira, s pomočjo katerega lahko novejši spletni brskalniki razberejo razpoložljivost spletnega vira in končnemu uporabniku to informacijo tudi ponudijo na različne načine.

3.9 Uporaba API-jev

V spletno aplikacijo sem za zaključek demonstracije konceptov in značilnosti spleta druge generacije vključil še uporabo API-jev. Natančneje sem predstavil, kako si lahko z vgraditvijo API-jev prihranimo veliko dela oziroma kako nam Google Maps API to omogoča. Za to demonstracijo sem v spletni aplikaciji izdelal poseben prikaz, ki na URL naslovu http://www.deanpodgornik.si/api-demonstracija tako rekoč služi lociranju več krajev na zemljevidu, kjer je administrator aplikacije fizično dostopen, ter tudi iskanju najoptimalnejše poti med želeno uporabnikovo lokacijo in navedenimi lokacijami administratorja. Predstavlja nekakšen naprednejši prikaz »kje smo«, ki se pogosto pojavi na raznih novodobnih spletnih mestih.

Na tem prikazu spletne aplikacije sem z vključitvijo raznih JavaScript datotek, dostopnih na strežnikih podjetja Google, na enostaven in hiter način prikazal osnoven zemljevid z eno izpostavljeno lokacijo. Za ta korak ni bilo potrebno nobeno posebno programersko predznanje in bi ga lahko brez težav izvedel tudi laik s področja spletnega razvoja. Prava moč tega API-ja pa se pokaže šele pri uporabi njegovih funkcionalnosti upravljanja z lokacijami, ki sem jih v nadaljevanju seveda tudi predstavil. Odločil sem se, da bom na zemljevidu vizualiziral več lokacij, kjer je administrator aplikacije fizično dostopen. V ta namen sem izdelal XML datoteko, za katero sem tudi predvidel svojo strukturo in vanjo navedel vse lokacije, ki naj se na zemljevidu prikažejo. Za vsako lokacijo posebej sem znotraj XML dokumenta ustvaril svoj XML element, kateremu sem nato v atributih navedel zemljepisno širino, zemljepisno dolžino, ime lokacije in naslov lokacije. To XML datoteko sem ob vsakem zahtevku za osvežitev prikaza ponovno prebral, iz nje razčlenil potrebne podatke lokacij ter jih na predpisan način posredoval Google Maps API-ju, ki jih je nato na zemljevidu sam vizualiziral. Poleg tega sem še omogočil, da se ob kliku na posamezno lokacijo nad njo pojavi oblaček z navedenim naslovom lokacije, ki se prebere iz XML datoteke. Za omogočanje te funkcionalnosti pojavljanja oblačka sem seveda uporabil zmogljivosti Google Maps API-ja, natančneje zmožnost upravljanja z dogodki ter zmožnost generiranja oblačkov znotraj zemljevida, pri katerih mi je bila v pomoč velika količina razpoložljive dokumentacije Google Maps API-ja.

Kot sem že omenil, sem za še bolj poglobljeno predstavitev tega API-ja vključil tudi funkcionalnost iskanja najoptimalnejše poti od uporabnikove želene lokacije do ene izmed lokacij, navedenih v XML datoteki, kjer je administrator aplikacije fizično dostopen. Vključil sem namreč vnosno polje, v katerega uporabnik navede svojo lokacijo, na primer »Nova

Gorica«, ter seznam razpoložljivih lokacij, iz katerega uporabnik izbere želeno lokacijo, kjer je administrator fizično dostopen. Ko uporabnik navede oziroma izbere zahtevane parametre ter seveda to akcijo tudi potrdi, se izvede JavaScript programska koda, ki Google Maps API-ju preko definiranega načina posreduje izbrano začetno in končno lokacijo. Google Maps API nato sam poskrbi za izračun najoptimalnejše poti ter tudi za vizualizacijo le-te. Vse te operacije potekajo s pomočjo tehnologije AJAX, tako da je celotna interakcija zelo tekoča in predvsem uporabniku prijazna. JavaScript programska koda, ki omogoča upravljanje z Google Maps API-jem oziroma vizualizacijo lokacij na Google Maps zemljevidu ter iskanje najoptimalnejše poti med dvema lokacijama, je priložena v prilogi F.

Menim, da sem z vključitvijo Google Maps API-ja dosegel zastavljeni cilj. Predstavil sem namreč, da si z uporabo API-jev lahko na enostaven in hiter način olajšamo veliko zahtevnega dela. Če bi moral sam implementirati vse uporabljene funkcionalnosti na zemljevidu, ki mi jih je ponujal Google Maps API, bi za to implementacijo porabil ogromno časa, z uporabo Google maps API-ja pa je celoten razvoj tega prikaza zahteval le tri ure dela. Izpostavil bi še primerjavo s kompleksnejšimi spletnimi storitvami, natančneje s tehnologijo SOAP. Google Maps API sem namreč vzpostavil veliko enostavneje kot pa odjemalca oziroma uporabnika storitve za spletno storitev vključevanja bloga, saj za vzpostavitev API-ja ni bilo potrebno uporabljati kompleksnih orodij za zagotavljanje komunikacije med ponudnikom storitve in uporabnikom storitve, ampak le osnovno HTML in JavaScript kodo.

Kljub temu, da omenjeni prikaz spletne aplikacije ponuja precej kompleksno storitev, pa se ta ne more klasificirat kot mashup. Za pridobitev tega naziva bi bila potrebna vključitev vsaj še enega zunanjega API-ja, saj je trenutno vključen le Google Maps API, ki uporablja podatke, shranjene na strežniku aplikacije. V primeru, da bi vključil še API za pridobivanje prometnih informacij, ki bi v kombinaciji z Google Maps API-jem vizualiziral, kje na ponujeni poti lahko uporabnik pričakuje prometne zastoje, bi se posledično ta prikaz aplikacije oziroma storitev klasificirala kot mashup.

4 SKLEPNE UGOTOVITVE

Splet druge generacije je danes, v letu 2011, že močno uveljavljen oziroma je v pravem razcvetu in njegove kritike so skoraj že potonile v pozabo. Kljub njegovi visoki dovršenosti pa se še naprej neprestano razvija in širi. Pri tej širitvi sodelujejo tudi njegovi etični, ustvarjalni in kritični uporabniki. Meja z njegovim predhodnikom, tako imenovanim spletom prve generacije, je vedno bolj jasna, tako da ni več dvoma, da si splet novo oznako vsekakor zasluži.

V diplomski nalogi sem poleg spleta druge generacije predstavil tudi, da je moč z uporabo sodobnih spletnih tehnologij ter trenutnih konceptov oziroma značilnosti spleta druge generacije na enostaven in hiter način zgraditi manjšo spletno aplikacijo, ki ji upravičeno lahko rečemo spletna aplikacija spleta druge generacije. Z uporabo orodij, kot so CMS sistem Myportal, JavaScript ogrodje Mootools ter skupek PHP razredov pod imenom NuSOAP, sem si olajšal večino kompleksnega in zamudnega kodiranja ter se tako lahko bolj posvetil implementaciji raznih konceptov oziroma značilnosti spleta druge generacije. Posamezne značilnosti spleta druge generacije sem tudi implementiral na precej enostaven način z uporabo na spletu razpoložljivih API-jev, ki kot storitev ponujajo vgraditev različnih značilnosti spleta druge generacije. Tak primer je uporabljen API Facebook Connect, ki omogoča prijavo uporabnika na katerokoli spletno mesto z uporabniškim računom socialnega omrežja Facebook.

Na podlagi izdelane manjše spletne aplikacije menim, da je za uspešno izgradnjo spletne aplikacije spleta druge generacije potrebno vanjo vključiti koncepte vseh treh glavnih elementov spleta druge generacije: RIA, SOA in socialni splet. Z vpeljavo konceptov RIA uporabniku aplikacije zagotovimo bogato uporabniško izkušnjo oziroma visoko stopnjo interaktivnosti ter posledično dosežemo, da postane spletna aplikacija uporabniku veliko bolj prijazna. Z uporabo konceptov SOA se v kompleksnejših aplikacijah poveča interoperabilnost, olajša se tudi ponovna uporaba storitve ali spreminjanje le-te. S tem pristopom se lahko v aplikacijo vključi podatke in funkcionalnosti drugih aplikacij oziroma storitev ter se tako zniža stroške razvoja aplikacije. Lahko pa se tudi aplikacijo odpre navzven in se z vključitvijo raznih funkcionalnosti in podatkov aplikacije na druga spletna mesta poveča prepoznavnost same aplikacije. Pri uporabi konceptov socialnega spleta pa se končne uporabnike povabi k ustvarjanju uporabniško generirane vsebine, s čimer se spodbudi občutek pripadnosti skupnosti in občutek prepoznavnosti, kar se pozitivno odraža tudi na sami aplikaciji, saj se na ta način širi krog uporabnikov, ki hkrati prispevajo k vedno bogatejši vsebini aplikacije. Poleg teh glavnih elementov oziroma njihovih konceptov pa menim, da je za uspešno spletno aplikacijo potrebna uporaba tudi ostalih značilnosti spleta druge generacije, ki vključujejo uporabo značk, omogočanje spletnega vira ter iskanje po vsebini. Delovanje oziroma uporaba teh značilnosti je pri večini uporabnikov postala že intuitivna, kar se seveda lahko uporabi v prid zagotavljanja enostavne interakcije uporabnika z aplikacijo.

Pri grajenju spletne aplikacije sem naletel tudi na nekaj manjših težav, in sicer mi je preglavice povzročalo neenako izvajanje Javascript programske kode v različnih spletnih brskalnikih. Posamezni spletni brskalniki namreč določene JavaScript ukaze različno interpretirajo ter tako tudi povzročijo razne anomalije. Za zagotavljanje konsistentnosti izvajanja JavaScript programske kode v vseh trenutno aktualnih spletnih brskalnikih sem moral izvajanje spletne aplikacije testirati v vseh teh spletnih brskalnikih ter po potrebi tudi izbrati druge pristope v JavaScript programski kodi.

Kot večino novo izdelanih aplikacij je mogoče tudi mojo manjšo spletno aplikacijo še izboljšati, kljub temu da sem že sam vpeljal veliko izboljšav po osnovnem dokončanem razvoju aplikacije. Tako bi na primer lahko v moji aplikaciji zagotovil večjo podporo uporabnikom, ki imajo v spletnem brskalniku onemogočeno tehnologijo JavaScript ter posledično tudi tehnologijo AJAX in posledično ne morejo izvajati vseh operacij kot ostali uporabniki. Menim, da je za prvo različico trenutna rešitev zadovoljiva, kljub temu pa je potrebno upoštevati, da sodobne spletne aplikacije zahtevajo stalen razvoj, odpravljanje pomanjkljivosti in nadgrajevanje v skladu z aktualnimi trendi spleta, saj sicer sčasoma postanejo zastarele in začne število njihovih uporabnikov hitro upadati.

PRILOGE

Priloga A: JavaScript programska koda za upravljanje z API-jem Facebook Connect

```
//spodnja programska koda se zažene ob vsaki zgraditvi DOM-a
window.addEvent('domready',function(){
  //ID Facebook API-ja za domeno www.deanpodgornik.si
  var fb_api_id='171556686230276';
  //inicializacija API-ja
  FB.init({
    appId:fb_api_id, cookie:true, status:true, xfbml:true
  });
  FB.api('/me', function(user) {
      //Dogodek prijave uporabnika
    FB.Event.subscribe('auth.login', function(){
      var fb_data="";
      //Inicializacija storitve Facebook Connect
      FB.init({
        //podati je potrebno identifikacijsko številko aplikacije,
        //ki se ustvari na servisnih straneh socialnega omrežja Facebook
        appId:fb_api_id,
        cookie:false, status:true, xfbml:true
      });

      //pridobitev podatkov o uporabniku (ime, priimek, email)
      FB.api('/me', function(user) {
        if(user != null) {
           fb_data={
            id: user.id, //ID uporabnika
            first_name: user.first_name, //ime uporabnika
            last_name: user.last_name, //priimek uporabnika
            email: user.email //email uporabnika
           }
        }
      });

      //funkcija za nastavljanje seje oziroma prijavo uporabnika
      var nastavim_sejo=function(){
        if(fb_data){
          //ID uporabnika Facebook-a zapišem v session, s pomočjo
          //tehnologije AJAX
          var req = new Request.HTML({
            evalScripts: true,
            url: MP_path+'/sys/session',
            data: {
                fb_data: fb_data,
                action: "set"
            },
            onComplete: function(){
              //Ob uspešno nastavljeni seji je potrebna osvežitev strani
              //zaradi posodobitve podatkov na strani
              window.location.reload();
            }
          });
          req.post();
        }
      //V primeru, da Facebook še ni vrnil podatkov, se funkcija periodično
```

```
      //ponavlja dokler teh podatkov ne pridobi
      }.periodical(200);
    });

    //Dogodek odjave uporabnika
    FB.Event.subscribe('auth.logout', function() {
      //pobriše se seja, s pomočjo tehnologije AJAX
      var req = new Request.HTML({
        evalScripts: true,
        url: MP_path+'/sys/session',
        data: {
            action: "reset"
        },
        onComplete: function(){
          //Ob uspešno izbrisani seji je potrebna osvežitev strani
          //zaradi posodobitve podatkov na strani
          window.location.reload();
        }
      });
      req.post();
    });
  });
});
```

Priloga B: WSDL dokument

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<definitions xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="urn:blog_service_WSDL"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="urn:blog_service_WSDL">
  <types>
    <xsd:schema targetNamespace="urn:blog_service_WSDL">
       <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
       <xsd:import namespace="http://schemas.xmlsoap.org/wsdl/" />
       <!-- definirana podatkovna struktura vhodnih parametrov -->
       <xsd:complexType name="struktura_iskalnih_parametrov">
        <xsd:sequence>
         <xsd:element name="iskani_niz" type="xsd:string"/>
         <xsd:element name="tip_iskanja" type="xsd:string"/>
        </xsd:sequence>
       </xsd:complexType>
       <!-- definiran podatkovni tip vhodnih parametrov
       (struktura_iskanih_parametrov) -->
       <xsd:complexType name="izvedi_iskanjeRequestType">
        <xsd:all>
         <xsd:element name="iskalni_parametri"
            type="tns:struktura_iskalnih_parametrov" form="unqualified"/>
        </xsd:all>
       </xsd:complexType>
       <!-- definiran podatkovni tip izhoda (tabela) -->
       <xsd:complexType name="izvedi_iskanjeResponseType">
        <xsd:all>
         <xsd:element name="return" type="xsd:Array" form="unqualified"/>
```

```
        </xsd:all>
      </xsd:complexType>
      <xsd:element name="izvedi_iskanje"
           type="tns:izvedi_iskanjeRequestType"/>
      <xsd:element name="izvedi_iskanjeResponse"
           type="tns:izvedi_iskanjeResponseType"/>
    </xsd:schema>
  </types>
  <message name="izvedi_iskanjeRequest">
    <part name="parameters" element="tns:izvedi_iskanje" />
  </message>
  <message name="izvedi_iskanjeResponse">
    <part name="parameters" element="tns:izvedi_iskanjeResponse" />
  </message>
  <portType name="blog_service_WSDLPortType">
    <operation name="izvedi_iskanje">
      <!-- dokumentacija -->
      <documentation>Metoda izvedi_iskanje kot vhodne parametre sprejem
tabelo dveh nizov. Prvi niz je namenjen iskanemu nizu, drugi pa tipu
iskanja, katerega vrednosti so lahko: tag, post, ali pa prazen niz.
Vrednost tag oznacuje, da se iskani niz nanasa na iskanje po znackah,
vrednost post, da se iskani niz nanasa tocno na dolocen URL naslov objave,
prazna vrednost pa, da se iskani niz nanasa na katerikoli atribut objave.
Kot rezultat pa metoda vrne dvo-dimenzionalno tabelo s podatki o posamezni
objavi</documentation>
      <input message="tns:izvedi_iskanjeRequest"/>
      <output message="tns:izvedi_iskanjeResponse"/>
    </operation>
  </portType>
  <binding name="blog_service_WSDLBinding"
      type="tns:blog_service_WSDLPortType">
    <soap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http"/>
    <!-- ime registrirane metode -->
    <operation name="izvedi_iskanje">
      <soap:operation soapAction="urn:blog_service_WSDL#izvedi_iskanje"
           style="document"/>
      <input><soap:body use="literal"
           namespace="urn:blog_service_WSDL"/></input>
      <output><soap:body use="literal"
           namespace="urn:blog_service_WSDL"/></output>
    </operation>
  </binding>
  <service name="blog_service_WSDL">
    <port name="blog_service_WSDLPort"
      binding="tns:blog_service_WSDLBinding">
      <!-- naslov streznika oziroma ponudnika storitev -->
      <soap:address
        location="http://www.deanpodgornik.si//sys/soap_server/"/>
    </port>
  </service>
</definitions>
```

Priloga C: PHP programska koda ponudnika storitve

```
<?php
require_once('lib/nusoap.php');

// Kreiranje strežnika
```

```
$server = new soap_server;

// Vklop WSDL podpore
$server->configureWSDL('blog_service_WSDL', 'urn:blog_service_WSDL',
'http://www.deanpodgornik.si/sys/soap_server/');

// Namespace
$server->wsdl->schemaTargetNamespace = 'urn:blog_service_WSDL';

// Registracija podatkovne strukture, ki jo uporablja storitev(lahko se jih
// definira več)
$server->wsdl->addComplexType(
    'struktura_iskalnih_parametrov', //ime podatkovne strukture
    'complexType',
    'struct',
    'sequence',
    '',
    array(
        'iskani_niz' => array('name' => 'iskani_niz',
            'type' => 'xsd:string'),
        'tip_iskanja' => array('name' => 'tip_iskanja',
            'type' => 'xsd:string'),
    )
);

//registracija metode, ki se obnaša kot storitev
$server->register('izvedi_iskanje', // Ime metode
    array('iskalni_parametri' => 'tns:struktura_iskalnih_parametrov'),
      // Vhodni parametri
    array('return' => 'xsd:Array'),           // Izhodni parametri
    'urn:blog_service_WSDL',                  // Namespace
    'urn:blog_service_WSDL#izvedi_poizvedbo', // Funkcija
    'document',                               // Način
    'literal',                                // Uporaba
    'Metoda izvedi_poizvedbo kot vhodne parametre sprejem tabelo dveh
nizov. Prvi niz je namenjen iskanemu nizu, drugi pa tipu iskanja, katerega
vrednosti so lahko: "tag", "post", ali pa prazen niz. Vrednost "tag"
oznacuje, da se iskani niz nanasa na iskanje po znackah, vrednost "post",
da se iskani niz nanasa tocno na dolocen URL naslov objave, prazna vrednost
pa, da se iskani niz nanasa na katerikoli atribut objave. Kot rezultat pa
metoda vrne dvo-dimenzionalno tabelo s podatki o posamezni objavi'
                                          // Dokumentacija
);

// glavna PHP metoda te spletne storitve
function izvedi_iskanje($iskalni_parametri){
  global $DB;
  //preverjanje po katerem polju naj se izvede iskanje
  $SQL_poizvedba="";
  switch ($iskalni_parametri['tip_iskanja']) {
    case "tag":
      $SQL_poizvedba="articles.tags LIKE '%" .
            $iskalni_parametri['iskani_niz']."%'";
    break;
    case "post":
      SQL_poizvedba="articles.furl='".$iskalni_parametri['iskani_niz']."'";
    break;
    default:
      //po vseh poljih
      $SQL_poizvedba="
        articles.name LIKE '%".$iskalni_parametri['iskani_niz']."%'
```

```
        OR articles.description LIKE
            '%".$iskalni_parametri['iskani_niz']."%'
        OR articles.text LIKE '%".$iskalni_parametri['iskani_niz']."%'
        OR articles.tags LIKE '%".$iskalni_parametri['iskani_niz']."%'";
    break;
  }
  //Poizvedba v podatkovno bazo
  $rezultat=$DB->get_results("
    SELECT articles.*,users.name AS first_name,users.surname AS last_name
    FROM articles
    LEFT JOIN users ON articles.uid=users.id
    WHERE (".$SQL_poizvedba.") AND articles.cid=1 AND articles.published=1
    ORDER BY articles.datefield DESC");

  return array('return' => $rezultat);
}

// Zgoraj definiran zahtevek uporabi za vzpostavitev storitve
$HTTP_RAW_POST_DATA = isset($HTTP_RAW_POST_DATA) ? $HTTP_RAW_POST_DATA :
      '';
$server->service($HTTP_RAW_POST_DATA);
?>
```

Priloga D: PHP programska koda uporabnika storitve

```
<?php
require_once('nusoap.php');

  class nu_soap_client_helper {
    private $client;

    function __construct($proxyhost='',
      $proxyport='',$proxyusername='',$proxypassword='',$usecurl='') {
      global $DB;

      //deklaracija parametrov za spletno storitev
      $proxyhost = isset($proxyhost) ? $proxyhost : '';
      $proxyport = isset($proxyport) ? $proxyport : '';
      $proxyusername = isset($proxyusername) ? $proxyusername :
      '';
      $proxypassword = isset($proxypassword) ? $proxypassword :
      '';
      $useCURL = isset($usecurl) ? $usecurl : '0';

      $this->client = new
            nusoap_client("http://www.deanpodgornik.si/sys/soap_server?wsdl
            ", 'wsdl',      $proxyhost, $proxyport, $proxyusername,
            $proxypassword);
      $err = $this->client->getError();
      if ($err) {
        echo '<h2>Napaka konstruktorja</h2><pre>'.$err.'</pre>';
        echo '<h2>Debug</h2><pre>'.htmlspecialchars(
            $this->client->getDebug(), ENT_QUOTES). '</pre>';
        exit();
      }
      $this->client->setUseCurl($useCURL);

    }
```

```
public function get_results($tip_iskanja='',$get_2='',$iskani_niz=''){
  //priprava parametrov za klic metode znotraj spletne storitve na
  //podlagi trenutnega URL naslova
  switch ($tip_iskanja) {
    case "tag":
      //iskanje po značkah
      $params = array(
        'iskani_niz' => $get_2,
        'tip_iskanja' => 'tag'
      );
    break;
    case "post":
      //iskanje samo določenega članka (izpis določenega članka)
      $params = array(
        'iskani_niz' => $get_2,
        'tip_iskanja' => 'post'
      );
    break;
    default:
      //iskanje po kateremkoli polju
      $params = array(
        'iskani_niz' => $iskani_niz,
        'tip_iskanja' => ''
      );
    break;
  }

  //klic metode izvedi_iskanje z želenimi parametri
  $tmp_results = $this->client->call('izvedi_iskanje',
        array('iskalni_parametri' => $params));
  if ($this->client->fault) {
    echo '<h2>Napaka (Zahtevek vsebuje neveljavno SOAP
        telo)</h2><pre>';
    print_r($tmp_results); echo '</pre>';
  } else {
    $err = $this->client->getError();
    if ($err) {
      echo '<h2>Napaka</h2><pre>' . $err . '</pre>';
    } else {
      //NI NAPAK, KLIC SE JE IZVEDEL, potrebno je le
      //še pripraviti pravilno tabelo za izpis
      $tmp_results=reset($tmp_results);
      if($tmp_results){
        if(($tmp_results[item][0])){
          $tmp_results=reset($tmp_results);
        }
      }
      return $tmp_results;
    }
  }

  //NAPAKA - če se do tu metoda še ni končala, pomeni da je prišlo do
  //napake, zato tudi metoda tu vrne false
  return false;
}

//Metoda, ki omogoča razhroščevanje v primeru napak
public function debug(){
  echo '<h2>Request</h2><pre>' . htmlspecialchars(
        $this->client->request, ENT_QUOTES) . '</pre>';
  echo '<h2>Response</h2><pre>' . htmlspecialchars(
```

```
            $this->client->response, ENT_QUOTES) . '</pre>';
        echo '<h2>Debug</h2><pre>' . htmlspecialchars(
            $this->client->getDebug(), ENT_QUOTES) . '</pre>';
    }
  }

  $nu_soap = new nu_soap_client_helper();
  //rezultat, kot dvo-dimenzionalna tabela objav bloga
  $result=$nu_soap->get_results($GET[1],$GET[2],
      $_GET['iskana-beseda']);
?>
```

Priloga E: JavaScript programska koda za upravljanje z AJAX-om

```
//deklaracija gumba za potrditev oddaje komentarja
commnet_form_submit=comment_form.getFirst('div.submit_cnt').getFirst('input
.submit');
//deklaracija dogodka oddaje komentarja
comment_form.addEvent('submit',function(e){
  e.stop();
  if(!comment_form.getFirst('textarea').value){
    //V primeru, da komentar ni naveden, se izpiše napaka
    comment_form.getFirst('div.submit_cnt').getFirst('span.result').set(
      'text',"Napaka");
  }else{
  //začasno onemogočim gumb za ponovno oddajo komentarja
  commnet_form_submit.toggleClass('dn');
  //deklaracija AJAX-a za oddajo komentarja
  var my_ajax = new Request.HTML({
    //URL naslov PHP skripte za shranjevanje komentarja
    url: comment_form.getProperty('action'),
    //metoda pošiljanja parametrov, ki je natančneje "post"
    method: comment_form.getProperty('method'),
    //podatki, ki se pošljejo PHP skripti (ID uporabnika, komentar)
    data: comment_form,
    //referenca na HTML element, v katerega se shrani rezultat AJAX-a
    update: comment_form.getFirst('div.submit_cnt').getFirst('
      span.result'),
    //navedba akcij, ki se zgodijo ob uspešni zaključitvi AJAX-a
    onComplete: function(){
    //ponovna omogočitev gumba za oddajo komentarja
    commnet_form_submit.toggleClass('dn');
    //izbris teksta v polju za oddajo komentarja
    comment_form.getFirst('textarea').innerHTML="";
    //izbris rezultata akcije čez 2 sekundi
    var pocakaj=function(){
      comment_form.getFirst('div.submit_cnt').getFirst('
          span.result').setProperty('text','');
    }.delay(2000);
    //kreiranje elementa v katerega se bo vstavil nov komentar
    var my_new_li  = new Element('li', {'class': 'comment h_0'});
    my_new_li.inject($('ul_comments'),'bottom');

    //IZVEDBA AJAXA-A ZA PRIKAZ ZADNJEGA KOMENTARJA
    var my_ajax_2 = new Request.HTML({
      url: MP_path+'/sys/comment_get_last',
      method: 'get',
      data: {
      cid: comment_form.getFirst('input.cid').value
```

```
      },
      update: $('ul_comments').getLast('li'),
      //navedba akcij, ki se zgodijo ob uspešni zaključitvi AJAX-a
      onComplete: function(){
      //izračun višine za HTML element novega komentarja
      preracunana_visina=$('ul_comments').getLast('li').getFirst('
        div').getStyle('height').toInt();
      visina_slike=$('ul_comments').getLast('li').getFirst('
        img').getStyle('height').toInt();
      if(preracunana_visina<visina_slike)
        preracunana_visina=visina_slike;
      var efekt = new Fx.Morph($('ul_comments').getLast('li'),{
            duration: 500, transition: Fx.Transitions.Sine.easeOut});
      //prikaz novega komentarja oziroma nastavitev višine njegovega bloka
      efekt.start({
        'height': preracunana_visina,
        'padding-top': '10px',
        'padding-bottom': '10px'
      });
      }
    });
    //Zagon AJAX-a za prikaz zadnjega komentarja
    my_ajax_2.send();
    }
  });
  //Zagon AJAX-a za dodajanje novega komentarja
  my_ajax.send();
  }
});
```

Priloga F: JavaScript programska koda za upravljanje z Google Maps API-jem

```
var side_bar_html = "";
var gmarkers = [];

//funkcija za kreiranje oblačka z besedilom
function createMarker(point,html,name)
{
  var marker=new GMarker(point);
  GEvent.addListener(marker, "click", function() {
    marker.openInfoWindowHtml(html);
  });
  gmarkers.push(marker);
  //dodajanje gumbov zunaj zemljevida
  side_bar_html += '<li><a class="locations" href="javascript:myclick(' +
(gmarkers.length-1) + ')">' + name + '<\/a><\/li>';
  return marker;
}

//funkcija, ki se izvede ob kliku na določeno lokacijo na zemljevidu
function myclick(i) {
  GEvent.trigger(gmarkers[i], "click");
  //označevanje končne točke za vizualizacijo najoptimalnejše poti
  var lat=gmarkers[i].Ca.y;
  var lng=gmarkers[i].Ca.x;
  //pridobitev imena lokacije iz seznama lokacij
  var locations_link=$$('a.locations')[i].getProperty('text');
  $('google_api_to_value').value=lat+","+lng;
  $('google_maps_api_to').innerHTML=locations_link;
```

```
  $('google_maps_api_to').removeClass('fc2');
}

//KO SE ZGRADI DOM, SE POKLIČEJO FUNKCIJE ZA PRIKAZ ZEMLJEVIDA TER
//INICIALIZACIJO LE TEGA
window.addEvent('domready', function() {
  //določanje prostora, kje naj se zemljevid prikaže
  var map = new GMap2(document.getElementById("map_canvas"));
  var markers;

  //razčlenjevanje XML dokumenta, ki vsebuje vse lokacije
  GDownloadUrl(MP_path+"/sys/map", function(doc){
    var xmlDoc = GXml.parse(doc);
    markers = xmlDoc.documentElement.getElementsByTagName("marker");
    for (var i = 0; i < markers.length; i++) {
      //pridobitev zemljepisne širine in zemljepisne dolžine
      var lat = parseFloat(markers[i].getAttribute("lat"));
      var lng = parseFloat(markers[i].getAttribute("lng"));
      var point = new GLatLng(lat,lng);
      //pridobitev imena in naslova lokacije
      var html = markers[i].getAttribute("html");
      var label = markers[i].getAttribute("label");
      var marker = createMarker(point,html,label);
      map.addOverlay(marker);
      //določitev center zemljevida, na podlagi prve lokacije V XML
      //dokumentu
      if(i==0){
        map.setCenter(new GLatLng(lat,lng), 8);
        map.setUIToDefault();
      }
    }
    //izpis lokacij oziroma povezav zunaj zemljevida
    document.getElementById("side_bar").innerHTML = side_bar_html;
    //inicializacija iskanja najoptimalnejše poti
    gdir = new GDirections(map, document.getElementById("directions"));
    //upravljanje z napakami, pri iskanju najoptimalnejše poti
    GEvent.addListener(gdir, "error", handleErrors);
  });

  //FUNKCIJE ZA IZRIS NAJOPTIMALNEJŠE POTI MED DVEMA TOČKAMA - začetek
    //pridobitev podatkov za izris najoptimalnejše poti
    $('google_api_submit').addEvent('click',function(){
      //preverim, če je uporabnik nastavil vse potrebne parametre
      $all_ok=true;
      if($('google_api_to_value').value==0){
        //parameter "do" ni določen
        $('google_maps_api_to').addClass('fc2');
        $all_ok=false;
      }
      if($('google_api_from_text').value==""){
        //parameter "od" ni določen
        $all_ok=false;
        $('google_api_from_text').setStyle('border','1px solid #0099FF');
      }else{
        $('google_api_from_text').setStyle('border','1px solid #DDDDDD')
      }

      //izvedem iskanje najoptimalnejše poti
      if($all_ok){
        setDirections($('google_api_from_text').value,
          $('google_api_to_value').value, "en_US");
```

```
      }
    });

    //funkcija, ki posreduje podatke Google Maps API-ju za izris
    //najoptimalnejše poti
    function setDirections(fromAddress, toAddress, locale) {
      gdir.load("from: " + fromAddress + " to: " + toAddress,
        { "locale": locale });
    }

    //funkcija za izpis napake pri iskanju najoptimalnejše poti
    function handleErrors(){
       if (gdir.getStatus().code == G_GEO_UNKNOWN_ADDRESS)
       alert("Napaka! Neznana lokacija.");
       else alert("Prišlo je do napake!");
    }
  //FUNKCIJE ZA IZRIS NAJOPTIMALNEJŠE POTI MED DVEMA TOČKAMA - konec
});
```

KAZALO SLIK

LITERATURA IN VIRI

[1] (2008) 7 Key Attributes of Social Web Applications. Dostopno na: http://connollyshaun.blogspot.com/2008/05/7-key-attributes-of-social-web.html

[2] (2006) A survey of new media. Dostopno na: http://www.economist.com/node/6794172?story_id=6794172

[3] (2011) AJAX Introduction. Dostopno na: http://www.w3schools.com/ajax/ajax_intro.asp

[4] (2002) Amazon Web Services API. Dostopno na: http://www.oreillynet.com/pub/wlg/1707?wlg=yes

[5] (2007) Enterprise Mashups. Dostopno na: http://msdn.microsoft.com/en-us/architecture/bb906060.aspx

[6] (2005) Folksonomy - New York Times. Dostopno na: http://www.nytimes.com/2005/12/11/magazine/11ideas1-21.html?_r=3

[7] (2006) DeveloperWorks Interviews: Tim Berners-Lee. Dostopno na: http://www.ibm.com/developerworks/podcast/dwi/cm-int082206txt.html

[8] (2011) JavaScript Introduction. Dostopno na: http://www.w3schools.com/js/js_intro.asp

[9] (2009) Millionth English word' declared. Dostopno na: http://news.bbc.co.uk/2/hi/middle_east/9380534.stm

[10] (2011) Usage Statistics and Market Share of Server-side Programming Languages for Websites. Dostopno na: http://w3techs.com/technologies/overview/programming_language/all

[11] (2004) Programming with NuSOAP Using WSDL. Dostopno na: http://www.scottnichol.com/nusoapprogwsdl.htm

[12] (2011) RSS Introduction. Dostopno na: http://www.w3schools.com/rss/rss_intro.asp

[13] (2009) Running your SOA like a Web startup. Dostopno na: http://www.zdnet.com/blog/hinchcliffe/running-your-soa-like-a-web-startup/525

[14] (2010) Semantični in sinaptični splet. Dostopno na: http://www.mojmikro.si/geekfest/moram_imeti/semanticni_in_sinapticni_splet

[15] (2005) SOA and Web Services. Dostopno na: http://www.oracle.com/technetwork/articles/javase/soa-142870.html

[16] (2009) SOA principes. Dostopno na: http://soaprinciples.com/

[17] (2011) Social Plugins - Facebook Developers. Dostopno na: http://developers.facebook.com/docs/plugins/

[18] (2007) Socialni zaznamki - Social Bookmarking - razlaga. Dostopno na: http://www.genspot.com/video-21106/socialni-zaznamki-social-bookmarking-razlaga.aspx

[19] (2010) Spletna socialna omrežja. Dostopno na: http://www.matejzagar.com/2010/01/socialna-omrezja-facebook-twitter/

[20] (2007) Thinking is so over - Times Online. Dostopno na: http://technology.timesonline.co.uk/tol/news/tech_and_web/personal_tech/article1874668.ece

[21] (2006) Time's Person of the Year: You - TIME. Dostopno na: http://www.time.com/time/magazine/article/0,9171,1569514,00.html

[22] (2007) Web 1.0 vs Web 2.0. Dostopno na: http://www.sizlopedia.com/2007/08/18/web-10-vs-web-20-the-visual-difference/

[23] (2006) Web 2.0 Compact Definition: Trying Again - O'Reilly Radar. Dostopno na: http://radar.oreilly.com/2006/12/web-20-compact-definition-tryi.html

[24] (2004) Web 2.0 Conference. Dostopno na: http://conferences.oreillynet.com/pub/w/32/presentations.html

[25] (2008) Web 2.0 Definition. Dostopno na: http://www.techterms.com/definition/web20

[26] J. Governor, D. Nickull, D. Hinchcliffe, *Web 2.0 Architectures*, O'Reilly Media, 2009.

[27] (2003) Web Services in PHP. Dostopno na: http://talks.php.net/show/oscon-webservices/

[28] (2011) Web Services Tutorial. Dostopno na: http://www.w3schools.com/webservices/default.asp

[29] (2005) What is Rich Internet Application (RIA). Dostopno na: http://searchsoa.techtarget.com/definition/Rich-Internet-Application-RIA

[30] (2005) What Is Web 2.0. Dostopno na: http://oreilly.com/web2/archive/what-is-web-20.html

[31] (2010) Which server-side technology should I choose. Dostopno na: http://www.adobe.com/devnet/dreamweaver/articles/which_serverside_technology.html

[32] (2011) Wiki. Dostopno na: http://en.wikipedia.org/wiki/Wiki