Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tomaž Ahlin

RAZVOJ LASTNEGA SISTEMA ZA UPRAVLJANJE Z VSEBINAMI

diplomsko delo
na univerzitetnem študiju

Mentor: izr. prof. dr. Viljan Mahnič

Ljubljana, 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

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

Št. naloge: 01934/2013
Datum: 03.09.2013

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

Kandidat: **TOMAŽ AHLIN**

Naslov: **RAZVOJ LASTNEGA SISTEMA ZA UPRAVLJANJE Z VSEBINAMI**
DEVELOPING OWN CONTENT MANAGEMENT SYSTEM

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Opišite vlogo sistemov za upravljanje z vsebinami in analizirajte značilnosti nekaterih prosto dostopnih rešitev (Wordpress, Joomla). Na podlagi ugotovljenih pomanjkljivosti realizirajte lasten sistem za upravljanje z vsebinami, opišite njegovo funkcionalnost in predstavite najpomembnejše tehnične rešitve. Na koncu primerjajte svojo rešitev z že obstoječimi.

Mentor:
izr. prof. dr.Viljan Mahnič

Dekan:
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Tomaž Ahlin, z vpisno številko **63070017**, sem avtor diplomskega dela z naslovom:

Razvoj lastnega sistema za upravljanje z vsebinami

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Viljana Mahniča,
- 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 24. september 2013 Podpis avtorja:

Zahvaljujem se mentorju izr. prof. dr. Viljanu Mahniču za dosegljivost, nasvete in usmerjanje pri izdelavi diplomske naloge. Prav tako gre velika zahvala sodelavcem pri podjetju VSISI, spletni marketing, d.o.o. V enaki meri pa se zahvaljujem navsezadnje tudi družini za vso potrebno podporo v času študija.

Kazalo

Povzetek

V diplomski nalogi je opisana realizacija lastnega spletnega sistema za urejanje vsebin.

V prvem delu naloge se osredotočimo na naloge tovrstnih spletnih sistemov. Sledi tudi opis in analiza že obstoječih sistemov, kot sta Wordpress ter Joomla. Predvsem nas zanimajo prednosti in slabosti omenjenih sistemov, da dobimo občutek za možne izboljšave.

V osrednjem delu naloge si podrobnejši opis ogledamo skozi uporabniški in kasneje tudi tehnični vidik. Z uporabniškim vidikom se osredotočimo na opis glavnih funkcionalnosti sistema, da razumemo, kaj približno je mogoče s sistemom realizirati in kako poteka delo z njim. Tehnični vidik pa najprej predstavi potrebno programsko opremo ter uporabljene tehnologije. Vsebuje tudi razlago nekaterih zanimivejših komponent, s katerimi je zagotovljena konkurenčna prednost sistema pred že obstoječimi. Razlaga je podkrepljena tudi s primeri programske kode, zato je naloga primerna tudi za bralce z znanjem objektnega programiranja.

Na koncu naloge si na kratko ogledamo še primerjavo med že obstoječimi sistemi ter razvito lastno rešitvijo. Primerjava je prikazana v obliki tabele, kjer pa je potrebno, je podan tudi bolj podroben opis primerjave.

Ključne besede: sistem za upravljanje z vsebinami, spletna stran, spletna optimizacija, objektno programiranje

Abstract

This thesis describes the realization of our own built content management system.

In the first part we focus on the functionalities of such systems, where we also make an analysis of currently existing systems, such as Wordpress and Joomla. We are mainly interested in their advantages and disadvantages to get some ideas for possible improvements.

In the second part we look at a more detailed description of our own built system through the user perspective and later on, the technical perspective. With the user perspective we look at the main functionalities of the system and their abilities. While technical perspective first introduces the necessary software to set up our own developed system and it's used technologies. It also contains an explanation of the most interesting functionalities which ensure a competitive advantage. The explanation is sometimes supported with samples of the source code, which makes the thesis a lot more attractive for readers with programming experience.

At the end of the thesis we make a brief comparison between the existing systems and our own developed system. The comparison is first shown with a simple table and later on, where required, supported with a more detailed description.

Keywords: content management system, website, search engine optimization, object-oriented programming

Poglavje 1

Uvod

V današnjih časih število spletnih strani hitro narašča, s tem pa tudi potrebe v podjetjih, da lahko na trgu ponudijo ugodnejšo izdelavo kakovostne spletne strani kot konkurenca. Da se zmanjšajo stroški izdelave spletne strani ter kasneje tudi vzdrževanje, se podjetja vse pogosteje odločajo za uporabo sistemov za upravljanje z vsebinami (angl. Content Management System - CMS) [1]. Primarni namen CMS sistemov [2] je delo z raznimi vsebinami, kar je razvidno iz same kratice. Vendar pa se je potrebno zavedati, da poleg omenjenega pogosto omogočajo še veliko več.

Na trgu je že več pripravljenih CMS sistemov, med njimi so daleč najbolj uporabljeni seveda zastonjski, kateri so ponavadi tudi odprtokodni. Zelo poznana primera sta recimo Wordpress [3] ter Joomla [4]. Vsak izmed njiju pa ima tako prednosti, kot tudi slabosti, zato je včasih delo z njima nekoliko neprijetno. Več o njih si bomo pogledali v kasnejšem, za to namenjenem poglavju. Zaenkrat pa omenimo le to, da za bolj ali manj vse že obstoječe sisteme na trgu velja naslednje pravilo: več funkcionalnosti kot sistem podpira, zahtevnejše postaja delo z njim, pa naj bo to le urejanje vsebin, ali programiranje dodatnih funkcionalnosti.

Namen izdelave lastnega CMS je bil ponuditi podjetjem tehnično kakovostne spletne strani, predvsem iz vidika spletne optimizacije (angl. Search Engine Optimization - SEO) [5]. Spletna optimizacija je postopek, kjer se z dobro strukturiranim načinom urejanja spletnih vsebin pripomore k iz-

boljšavi pozicije spletišča med rezultati spletnih iskalnikov, kot je naprimer Google [6]. K dobrim pozicijam pripomorejo skrbno pripravljene vsebine, obogatene z raznimi naslovi in meta podatki. Med osnovne meta podatke [7] spletne strani spadata opis ter ključne besede.

Ker smo CMS sistem izdelali v podjetju, katerega glavna dejavnost je optimizacija spletnih strani, so razlogi večinoma naslednji:

1. Prvotni razlog, zakaj smo se odločili za izdelavo, je bilo dejstvo, da imajo že obstoječi CMS sistemi slabo podporo za večjezičnost. Vse več uspešnih podjetij želi prodreti tudi na tuje trge, zato je seveda potreba po večjezičnem spletišču skoraj nujnost. Wordpress omogoča postavitev takšne vrste spletišč z uporabo zunanjega vtičnika. Vendar se pojavi težava, ker prevedenim vsebinam v tujih jezikih ne moremo napisati tudi prevedenih meta podatkov. Manjše težave pa so tudi pri uporabi sistema Joomla.

2. Veliko oblikovnih predlog (angl. template), ki so že pripravljene za obstoječe sisteme, je zelo slabo optimiziranih. Uporabljajo odvečne elemente in s tem naredijo kodo v jeziku HTML (HyperText Markup Language) daljšo, kot je potrebno. Če je koda že v osnovi daljša, to hkrati pomeni tudi, da jo je kasneje težje spreminjati.

3. Pri že obstoječih sistemih je pogosto tako, da so nekoliko nerodni, ko spreminjamo enolične povezave (angl. Uniform Resource Locator - URL) vsebinskih strani. To pomeni, če URL prvotne vsebinske strani vključimo v vsebino drugih vsebinskih strani ter kasneje URL prvotne strani spremenimo, bodo v vsebinah ostalih vsebinskih strani ostale stare, sedaj nepravilne povezave. Če imamo malo vsebinskih strani, problem še ni tako izrazit, pri večjih spletiščih pa lahko hitro pride do večje zmede, ki negativno vpliva na obiskovalce strani ter spletne pajke (angl. crawler), kadar naletijo na nepravilne povezave.

Pred začetkom izdelave sistema CMS smo si dobro ogledali delovanje sistema Wordpress, da smo dobili dober pregled nad funkcionalnostmi, ki jih ponuja. Še posebej pozoren sem bil na pomanjkljivosti ter katere komponente bi bilo dobro dodati. Šele potem sem si lahko okvirno pripravil načrt in seznam stvari, ki jih bo potrebno realizirati – razne programske razrede ter funkcije. V mislih sem imel ves čas tudi to, da mora biti oblikovna predloga ločena od programske kode, saj je to nujno potrebno, da je postavitev spletišča preprosto in prijetno opravilo.

CMS sistem je bil po nekaj mesecih razvoja uspešno uporabljen za postavitev prvega spletišča. Prav tako je dokaj enostaven za programerja, hkrati pa lastniku ponuja preprosto upravljanje s spletiščem. Z nadaljnjimi izboljšavami in popravki je sistem postajal vse bolj uporaben ter vizualno prilagodljiv. V nadaljevanju naloge si bomo za lažje razumevanje, zakaj je lasten CMS sistem uspešen, najprej pogledali naloge spletnih CMS sistemov, nato pa opis sistemov Wordpress in Joomla. Pri slednjih bodimo še posebej pozorni na njune slabosti, saj so le te odpravljene v lastnem CMS sistemu.

Poglavje 2

Kratek opis spletnih CMS sistemov

Spletni CMS sistemi so sistemi, ki so dostopni preko spleta z uporabo spletnega brskalnika. Njihova najpomembnejša lastnost je, da vsebujejo administrativni vmesnik, ki je dostopen le s predhodno prijavo lastnika v sistem.

Glavne naloge spletnih CMS sistemov bi lahko okvirno razdelili v dve skupini:

1. Programerju in kasneje tudi lastniku spletišča (množica vseh spletnih strani, ki skupaj tvorijo celotno spletno mesto) omogočajo lažje delo s spletnimi vsebinami. Ker ima sistem ponavadi uporabniku prijazen spletni vmesnik, uporabnikom ni potrebno imeti širšega tehničnega znanja, kot ga ima recimo oseba, ki se že dlje časa ukvarja z izdelavo spletišč. Med drugim sistem tudi, če je le mogoče, uporabniku preprečuje, da bi na kakršenkoli način naredil večje sintaktične napake v izvorni kodi in s tem onemogočil pravilno prikazovanje določene spletne strani v brskalniku. Če ima spletišče več razdelkov, recimo novice, forum ter katalog izdelkov, je včasih možno urejanje vsakega razdelka dodeliti ločenemu uporabniku, glede na njegovo vlogo.

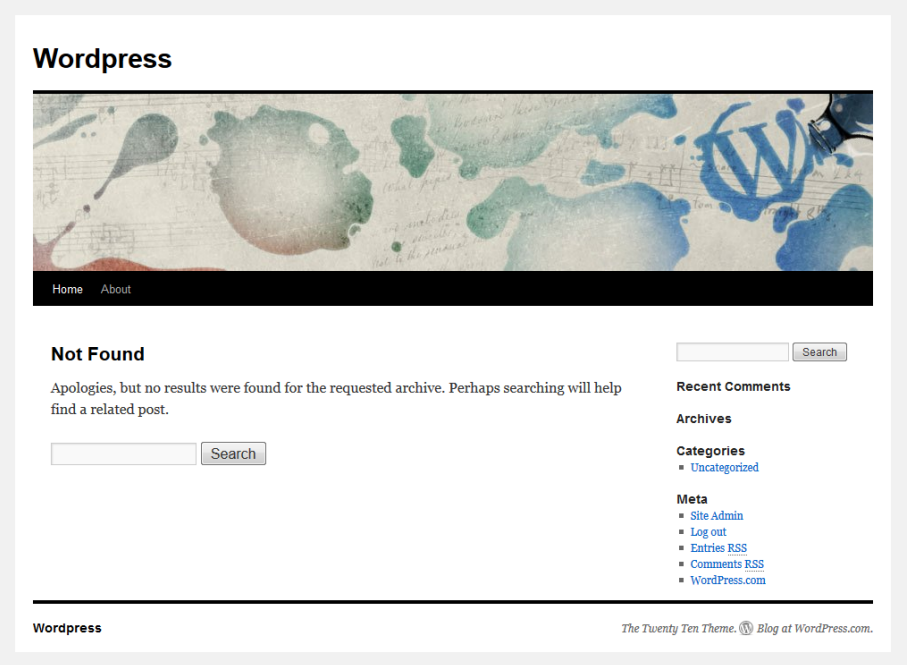
2. Lastniku spletišča omogočajo prikaz povratnih informacij. Ker CMS seveda ni nič drugega kot sodobna programska oprema, so podatki shranjeni v podatkovni bazi in ne več v navadnih datotekah. To omogoča lastniku prikaz zgodovine iskalnih poizvedb obiskovalcev strani, izris statistike obiska posameznih strani, seznam prejetih kontaktnih sporočil strank in še veliko več.

Za boljšo predstavo o CMS sistemih si v nadaljevanju oglejmo sistema Wordpress in Joomla ter na kratko nekaj zanimivosti o lastni razviti rešitvi.

2.1 Wordpress

Projekt Wordpress se je pričel že leta 2001, po dveh letih razvoja pa je bila uporabnikom ponujena prva dostopna različica. Wordpress je napisan v programskem jeziku PHP [30] in za shranjevanje informacij uporablja podatkovno bazo MySQL [31].

Wordpress je odprtokodni projekt, kar pomeni, da na njem sočasno dela veliko ljudi po celem svetu. Uporablja ga preko 60 milijonov spletišč, kar predstavlja približno 20% spleta. Začetni namen je bil ustvariti sistem z dobro arhitekturo, katerega komponente in funkcionalnosti bi bilo možno preprosto nadgrajevati. V začetni fazi je bil namenjen predvsem uporabnikom, ki se ukvarjajo z vsakodnevnim pisanjem vsebin, torej bloganjem. Temu primerno je oblikovana tudi uvodna stran spletišča, ki jo lahko vidimo na sliki 2.1.



Slika 2.1: Izgled uvodne spletne strani sistema Wordpress takoj po namestitvi.

Z nadaljnjim razvojem se je Wordpress vse do danes uveljavil kot vsestransko namenjen CMS, s katerim lahko uporabniki pišejo članke ter novice, katere lahko obiskovalci komentirajo. Tisti bolj tehnični pa lahko dodajo tudi različne vtičnike (angl. plugin). Do danes so uporabniki objavili na tisoče vtičnikov, gradnikov in tem. Ker so seveda napisani neodvisno, se pogosto zgodi, da med seboj razna dela niso združljiva, kar lahko postane težava, kadar jih uporabljamo v kombinaciji. Za boljše razumevanje, si poglejmo njegove prednosti in slabosti [11].

Najpomembnejše prednosti sistema Wordpress so sledeče:

1. Začetna namestitev je preprosta in ne zahteva predhodnega znanja, prav tako je preprosto tudi kasnejše upravljanje s sistemom [12].

2. Ponuja preprosto upravljanje z vsebinskimi stranmi. Vsaki ustvarjeni vsebinski strani lahko določimo URL, glavni naslov, kategorijo, povzetek ter niz, namenjen za iskalnik. Vsebino lahko z interaktivnim urejevalnikom dopolnimo z raznimi stili pisave, odlomki in slikami.

3. Omogočeno je dodajanje raznih vtičnikov, ki jih napišejo različni razvijalci. Večinoma so vtičniki zastonjski ter odprtokodni, najdejo pa se tudi boljši, ki pa so plačljivi. Nekateri vtičniki so namenjeni za izboljšavo uporabniške izkušnje, drugi pa so namenjeni tehnični izpopolnjenosti spletišča.

4. Wordpress uporabnikom ponuja ogromno oblikovnih predlog oziroma tem. Ker pa se na spletu nahaja množica različnih tem, ki jih seveda pripravijo različni ustvarjalci, se pogosto zgodi, da nekatere niso dovolj optimizirane s tehničnega vidika. Pogosto so boljše teme plačljive, hkrati pa so dobro pripravljene in zanimive za uporabnika.

Glavne slabosti sistema Wordpress pa so naslednje:

1. Osnovna različica ne omogoča vnosa meta oznak vsebinskim stranem, ki so ključnega pomena za spletne iskalnike. To slabost lahko preprosto odpravimo z uvedbo zunanjega vtičnika.

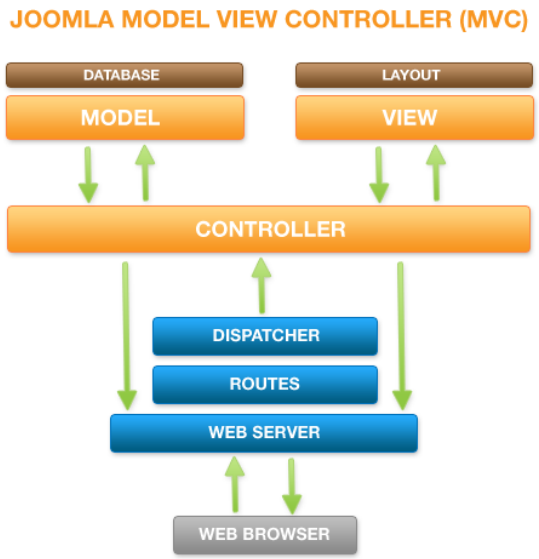
2. Težko je postaviti kakovostna večjezična spletišča. Obstajajo sicer vtičniki, ki nam omogočajo postavitev takšne vrste spletišč, med njimi je najbolj poznan vtičnik WPML (Wordpress Multilingual Plugin) [13]. Vendar se moramo spomniti, da se zunanji vtičniki ne zavedajo en drugega. Prav tako je uporabnik še vedno omejen glede urejanja vsebinskih strani po uvedbi omenjenega vtičnika.

3. Ob uporabi raznih vtičnikov se velikokrat zgodi, da se v dokumentu HTML, ki ga prejme brskalnik, pojavi tudi nezaželena dolga koda JavaScript, prav tako pa morda tudi koda CSS, če ni shranjena v ločenih datotekah.

4. V vsebinah se shranjujejo polne enolične povezave (URL) do ostalih strani, torej v takšni obliki, kot jo vidi tudi obiskovalec, ko se z miškinim kazalcem pomakne čez njih. Če lastnik izbriše določeno vsebinsko stran ali le spremeni njen URL, bo v ostalih vsebinah še vedno ostal star, sedaj nepravilen URL. To ni dobro za obiskovalce spletišča, saj ga bo klik na takšno povezavo ponavadi pripeljal do napake strežnika ali pa bo Wordpress opozoril, da stran s takšnim URL ne obstaja. Po drugi strani pa to negativno vpliva na spletno optimizacijo, kadar želimo imeti spletno stran prikazano čim višje med rezultati iskanja pri spletnih iskalnikih, recimo Googlu.

2.2 Joomla

Joomla je zastonjski CMS, ki je na voljo pod licenco GPL [14]. Napisan je v programskem jeziku PHP, za shranjevanje podatkov pa uporablja podatkovno bazo MySQL. Projekt Joomla sega nazaj do leta 2000, prve uporabnikom dostopne različice pa so bile na voljo po letu 2005. V tem času je na celotnem projektu sodelovalo ogromno delovnih skupin. Veliko vlogo ima tudi lastno delovno ogrodje (angl. framework), ki omogoča programerjem lažje in hitrejše delo. Programerji morajo biti zaradi delovnega ogrodja dobro seznanjeni z objektnim programiranjem, veliko pomoči pa lahko najdejo v dokumentaciji iz uradne spletne strani ali pa za pomoč povprašajo na za to namenjenem uradnem forumu. Razširitve (angl. extensions) so ponujene v obliki modulov, ki so integrirani v sistem. Omogočeno je tudi dodajanje razširitev, ki so na voljo na spletu za prenos in se lahko preprosto delijo med razvijalci. Sistem je zgrajen na sodobni arhitekturi MVC (Model View Controller), katero prikazuje slika 2.2. Izgled uvodne spletne strani pa je podoben, kot smo ga videli že pri sistemu Wordpress.



Slika 2.2: Ponazoritev elementov arhitekture MVC, ki jo uporablja Joomla.

Zaradi lastnega delovnega ogrodja je Joomla namenjena tudi naprednejšim spletnim portalom, kjer je pogosto potrebno dodati raznolike funkcionalnosti [15]. Takšni primeri bi lahko bila orodja za izdelavo poročil, katalogi izdelkov, integrirana e-trgovina, komunikacijska orodja ali rezervacijski sistemi.

Torej, poglejmo si najprej glavne prednosti sistema Joomla:

1. Uporaba lastnega delovnega ogrodja omogoča hitro in učinkovito nadgradnjo sistema.

2. Poleg vsebine se članku vpiše tudi opis in ključne besede. Zanimivo je, da je povezava, preko katere se naredi dostop do članka, imenovana pvsedonim (angl. alias). Opcijsko lahko uredimo tudi pravice dostopa obiskovalcem strani ter katera skupina prijavljenih uporabnikov ga lahko ureja. Poleg omenjenega lahko vnesemo še mnogo drugih opcijskih meta podatkov. Vnese se lahko tudi avtor članka ter čas veljavnosti.

3. Zanimivost sistema so tudi kategorije, ki so obvezen del sistema, saj se vanje razvrstijo članki. Na ta način dobimo sistematično urejen pregled nad članki, saj se jih da skoraj vedno razvrstiti po skupinah.

4. Dobra lastnost sistema Joomla je še, da je možno prijavljene uporabnike v sistemu ločiti po skupinah. Vsaki skupini se določi pravice, ki jih imajo njeni uporabniki pri urejanju spletišča.

Čeprav je sistem Joomla možno nadgrajevati, ima tudi nekaj slabosti:

1. Prej omenjeni članki morajo biti vključeni v eno izmed kategorij, kar pripomore k temu, da je URL posameznih člankov primerno daljši. Poleg tega so v URL pogosto vidne tudi identifikacijske številke strani (ID). Z zunanjim vtičnikom se lahko ti slabosti obideta, vendar lahko to pripelje do drugih težav.

2. Joomla v osnovi podpira postavitev večjezičnih spletišč, vendar je delo včasih zahtevno, saj je potrebno naložiti dodatne vtičnike. Zopet pa

lahko pride do podobnih težav kot pri sistemu Wordpress, da se nekaterih delov spletišča ne da prevesti. Večjezično podporo imajo samo novejše različice sistema Joomla.

3. Prav tako kot pri sistemu Wordpress se tudi pri sistemu Joomla povezave med vsebinskimi stranmi ob njihovem spreminjanju ne osvežujejo samodejno, kar lahko vodi do nekonsistentnih notranjih povezav.

2.3 Lasten CMS sistem

Sistem je, tako kot tudi predhodnika, napisan v programskem jeziku PHP, za shranjevanje podatkov pa uporablja podatkovno bazo MySQL. Razvoj je trajal približno 1 leto, v prihodnosti pa so planirane še dodatne razširitve.

Ker sem pred začetkom izdelave že imel izkušnje na področju programiranja, sem vsekakor izkoristil prednosti objektnega pristopa, kar se je kasneje med razvojem izkazalo za še posebej koristno. Moč je bilo ustvarjati ponovno uporabljive komponente in si s tem privarčevati ogromno programerskega časa. Še posebej sem si vnaprej pripravil samo strukturo aplikacije s tem, da sem programsko kodo razdelil v več različnih datotek PHP, pri čemer ima vsaka datoteka svoj pomen. Med samo izdelavo sem uporabil tudi nekaj prosto dostopnih razredov in knjižnic, ki sem jih vključil v sistem, da sem dopolnil še nekatere funkcionalnosti. Takšen primer sta knjižnica za ustvarjanje Excel datotek, imenovana PHPExcel [8] ter knjižnica za vizualizacijo podatkov, imenovana ArborJS [9]. Zadnjo sem moral še dopolniti in prilagoditi za ustrezno delovanje z obstoječimi podatki. Med izdelavo pa je bilo ves čas potrebno paziti še na to, da sistem med svojim izvajanjem ne izpisuje neveljavnih elementov v jeziku HTML, oziroma da je končna koda v jeziku HTML izpisana čim boljše glede na spletne standarde, s katerimi se ukvarja konzorcij W3C [10].

Ker je lastna rešitev obsežna in je na sistem možno gledati skozi oči navadnega uporabnika ali pa programerja, je opis razdeljen na uporabniški ter tehnični vidik, ki sledita v nadaljevanju naloge.

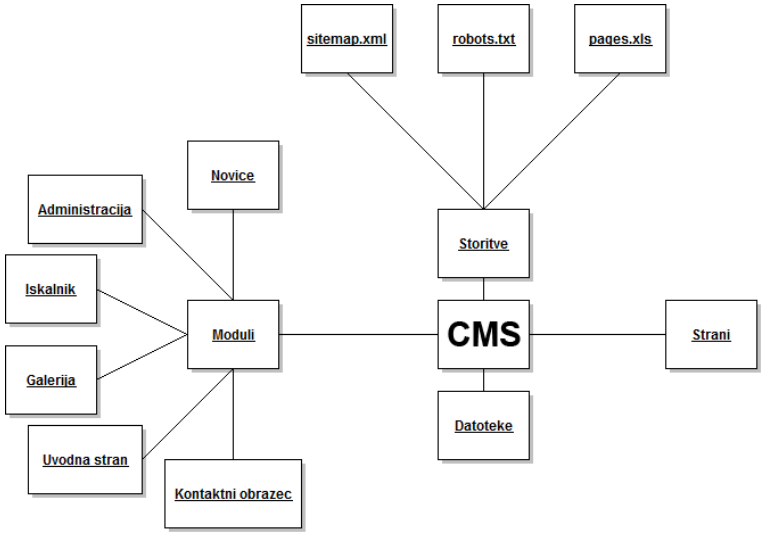
Poglavje 3

Uporabniški vidik

CMS je primarno namenjen izdelavi spletišč za podjetja, ki se ukvarjajo z raznolikimi dejavnostmi. Lahko pa se ga brez težav uporabi tudi za izdelavo osebnih ali drugačnih preprostejših spletišč. V okviru uporabniškega vidika si bomo najprej ogledali funkcionalnosti uporabniškega vmesnika (angl. user interface), ki so ponujene obiskovalcem, v kasnejšem poglavju pa možnosti urejanja spletišča, ki so lastniku spletišča ponujene preko administrativnega vmesnika.

3.1 Funkcionalnosti uporabniškega vmesnika

Za lažjo predstavo si na sliki 3.1 najprej oglejmo kaj vse lahko ponudimo potencialnim obiskovalcem, katerim je spletišče sploh namenjeno.

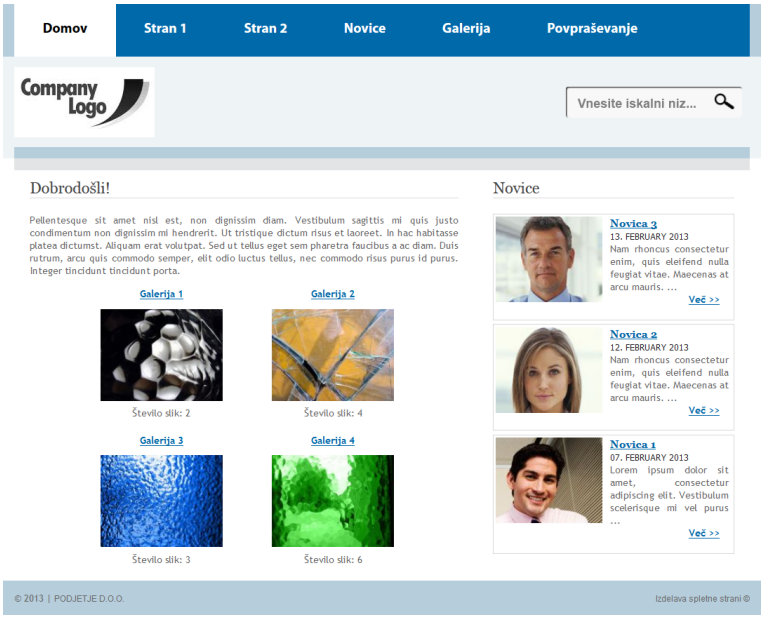


Slika 3.1: Prikaz funkcionalnosti, zanimivih za obiskovalce strani.

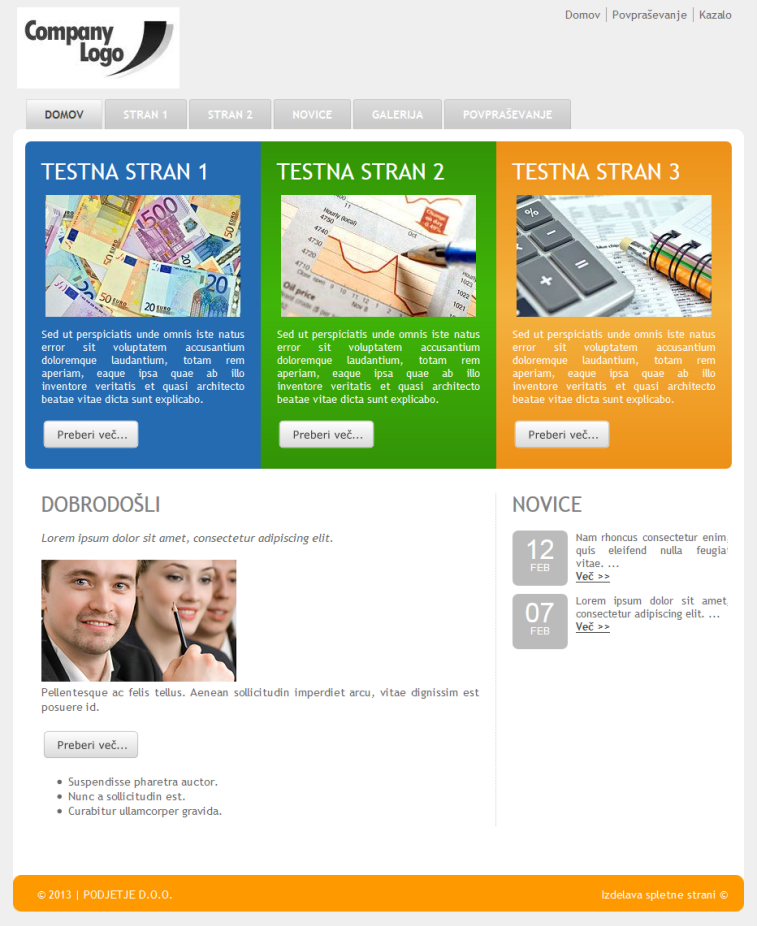
3.1.1 Moduli

Uvodna stran

Uvodna stran je ponavadi prva stran, ki jo stranka oziroma obiskovalec spletne strani vidi, zato je pomembno, da je skrbno oblikovana in nudi dober pregled nad tem, kaj spletišče ponuja. Iz tega razloga ima uvodna stran poleg lastnega naslova, opisa ter ključnih besed tudi povsem svojo oblikovno predlogo, katero je možno urediti po potrebah podjetja. Na uvodno stran lahko vključimo povezave na vsebinske strani, prikažemo zadnje novice, dodamo slike iz galerije, drsnik elementov ali kakšno drugačno, za uporabnika zanimivo interaktivno vsebino. Za hiter dostop do informacij je dobro na uvodno stran vključiti tudi meni, iskalnik ali morda celo kontaktni obrazec. Za boljšo predstavo si oglejmo primera uvodnih strani na slikah 3.2 in 3.3.



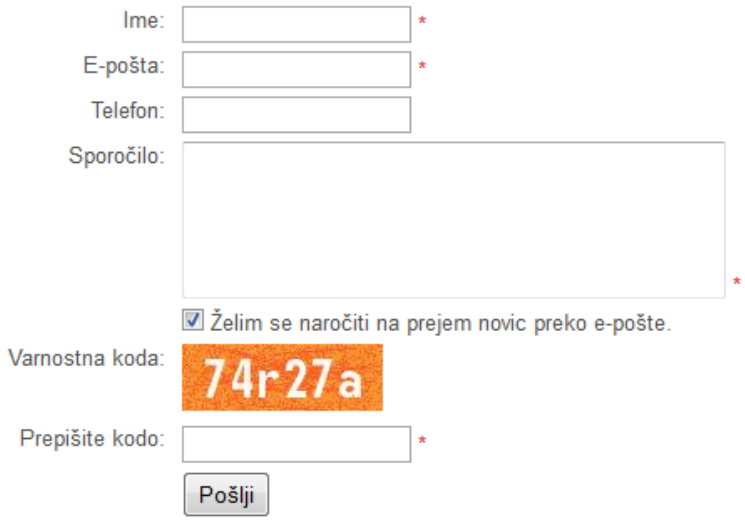
Slika 3.2: Prvi primer uvodne strani, ki v glavi oblikovne predloge vsebuje glavni meni, logotip podjetja ter iskalnik. Osrednji del je razdeljen na levi in desni stolpec. V levem stolpcu je predstavitveno besedilo z galerijami, v desnem pa zadnje vpisane novice s pripadajočimi sličicami. V nogi oblikovne predloge se nahajajo podatki o lastništvu ter izdelavi spletne strani.



Slika 3.3: Drugi primer uvodne strani, ki v glavi oblikovne predloge vsebuje pomožni meni, logotip podjetja ter glavni meni. V osrednjem delu se najprej nahajajo opisi glavnih dejavnosti podjetja, potem pa sledita levi in desni stolpec. V levemu stolpcu je predstavitveno besedilo s sliko ter seznamom, v desnem pa zadnje vpisane novice. V nogi oblikovne predloge se nahajajo podatki o lastništvu ter izdelavi spletne strani.

Kontaktni obrazec

Obiskovalci spletišča oziroma stranke lahko pošljejo povpraševanje preko spletnega obrazca, kot ga prikazuje slika 3.4. Obiskovalec mora obvezno vpisati svoje ime in elektronski naslov, kamor lahko podjetje odgovori. V primeru, da vpiše svojo telefonsko številko, pa se mu lahko odgovori tudi na ta način. Obiskovalec mora pred pošiljanjem sporočila iz majhne slike prepisati še varnostno kodo, ki deluje kot zaščita proti spletnim robotom. Po uspešno poslanem sporočilu se izpiše sporočilo o uspehu, lastnik pa na e-poštni naslov prejme sporočilo, ki je sestavljeno tako, da se da nanj direktno odgovoriti, odgovor pa prejme obiskovalec na svoj vpisani e-poštni naslov. Če iz kakršnega koli razloga lastnik spletišča ne prejme elektronske pošte, so sporočila še vedno shranjena v podatkovni bazi in se jih lahko pregleduje preko vmesnika v administraciji. S prejetim sporočilom lahko lastnik vidi na katerem spletnem naslovu znotraj spletišča se je obiskovalec nahajal pred pisanjem sporočila, vendar to velja le, če je kliknil na posebno povezavo, ki vključuje poleg povezave kontaktnega obrazca tudi referenco.



Slika 3.4: Kontaktni obrazec pred izpolnjevanjem.

Galerija

Kot vsak sodoben CMS sistem, tudi razvita rešitev podpira modul za urejanje galerij. Vsaka galerija lahko vsebuje poljubno število slik. Po začetnem ustvarjenju galerije se slike dodaja preko nalagalnika, ki je napisan v tehnologiji Flash [17], lahko pa tudi preko odjemalca FTP. Ko so slike naložene na strežnik, pa je možno vsaki sliki dopisati še naslov in opis. Nekaj zadnjih, točno določenih ali naključnih galerij se lahko vključi tudi na uvodni ali katerikoli drugi strani. Sicer pa je možen pregled nad vsemi galerijami. Ko se obiskovalec strani nahaja v določeni galeriji, lahko hitro in interaktivno pregleduje slike, saj so opremljene z vtičnikom LightBox [19], napisanem v tehnologiji JavaScript [25].

Novice

Novice imajo danes velik pomen, saj se z njimi stranke obvešča o ugodnostih ali novostih v ponudbi, lahko pa so drugačne informativne narave, s katero uporabnik dobi občutek, da je spletišče redno posodobljeno. Ob objavi novice lahko pisec označi možnost, če želi novico poslati tudi naročnikom. Zadnjih nekaj novic se lahko z uporabo gradnika izpiše na uvodni strani, kar smo opazili na slikah 3.2 in 3.3, medtem ko so vse novice na voljo v arhivu novic, ki ga prikazuje spodnja slika 3.5.

Slika 3.5: Arhiv novic, ki izpiše 2 novici na stran, ob strani pa je v oblikovni predlogi na voljo prijava na novice preko e-pošte.

Iskalnik

V samo spletišče je možno vključiti iskalnik. Z vnosom iskalnega niza v okence iskalnika, ki je viden na sliki 3.2, lahko obiskovalec hitro najde potrebne informacije. Aplikacija po prejemu iskalnega niza le tega ustrezno filtrira (odstrani nezaželene znake) in uporabnika preusmeri na stran z rezultati. Rezultati iskanja obsegajo vse dostopne vsebinske strani, iskalnik pa

išče tudi po novicah. Iskanje ne poteka le po principu, če neka vsebina vsebuje iskalni niz ali ne, temveč so rezultati urejeni po relevantnosti glede na iskalni niz. Preprosta formula, ki se uporabi pri izračunavanju relevantnosti zadetkov, se da prilagoditi. Seznam zadetkov iskanja prikazuje slika 3.6.

Slika 3.6: Prikaz rezultatov iskanja, pri čemer so rezultati urejeni po relevantnosti, ujemajoče besede pa so podčrtane.

Administracija

Poseben primer modula je navsezadnje tudi administracija, ki omogoča urejanje vseh vsebin ter modulov, kar je opisano v nadaljevanju naloge.

3.1.2 Storitve

Spletna storitev je metoda komuniciranja med napravami, ki se nahajajo v spletu. Lasten CMS sistem podpira naslednje spletne storitve [18]:

1. Storitev, ki vrne vsebino datoteke sitemap.xml. Vsebina se ustvari ob vsakem dostopu, zato nepotrebno osveževanje potrebne datoteke odpade. XML kazalo za obiskovalce strani ni posebej zanimivo, velik pomen pa ima pri obisku spletnih pajkov, saj lahko na enostaven način vnaprej dobijo vse spletne naslove spletišča.

2. Storitev, ki vrne vsebino datoteke robots.txt. Vsebina te datoteke je namenjena spletnim robotom, saj pove katerih spletnih naslovov naj spletni roboti nikakor ne indeksirajo. Ker je datoteka javno dostopna in berljiva, to seveda ni način za skrivanje pomembnih delov spletišča, recimo povezave do administracijskega vmesnika.

3. Storitev, ki vrne Excelovo datoteko z glavnimi podatki vsebinskih strani. Nad podatki v razpredelnici imamo hiter in učinkovit pregled.

3.1.3 Vsebinske strani

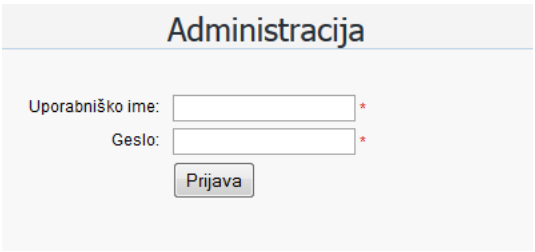
Vsebinske strani so ponavadi tiste, ki največ pripomorejo pri pozicijah spletišča med rezultati pri spletnih iskalnikih, zato morajo biti vsebinsko najbolje pripravljene za spletno optimizacijo. V ta namen so bile ločene od modulov, saj imajo pripravljenih veliko več vnosnih polj ter funkcionalnosti, ki omogočajo lažje delo in pripravo takšnih strani.

3.1.4 Datoteke

Na spletišče je možno naložiti neomejeno število datotek. Sem spadajo slike, ki so lahko vključene tudi v galerije, dokumenti ter glasbene in video datoteke. Datoteke so lahko ponujene v pomoč lastniku spletišča za podporo pri raznih sestankih, lahko pa služijo obiskovalcem spletišča, kadar je njihovo vsebino težje ali celo nemogoče vstaviti med vsebino.

3.2 Možnosti urejanja spletišča

Lastnik spletne strani se mora, če želi urediti določeno vsebino, najprej prijaviti v sistem. Prijavo izvede z vnosom uporabniškega imena in gesla v za to namenjen spletni obrazec, katerega prikazuje slika 3.7.



Slika 3.7: Izgled prijavnega spletnega obrazca

Po prijavi v sistem je lastniku v administracijskem vmesniku na voljo veliko možnosti, s katerimi lahko vpliva na spletišče, razvrstimo pa jih lahko v naslednje 3 skupine:

1. Urejanje osnovnih nastavitev spletišča
2. Urejanje vsebin
3. Delo z datotekami

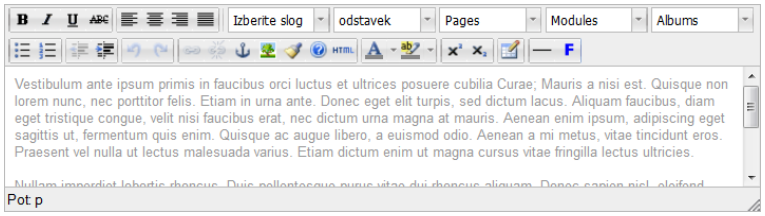
3.2.1 Urejanje osnovnih nastavitev spletišča

Sem spada urejanje podatkov podjetja, kot so naziv podjetja, opis ter ključne besede dejavnosti, slogan, obdobje poslovanja, pripadajoči elektronski poštni naslov za povpraševanja ter spletni naslovi najznamenitejših družabnih omrežij.

Po drugi strani pa se ureja tudi nekoliko bolj tehnične lastnosti spletišča, recimo maksimalno število izpisanih objektov pri uporabi paginacije, dimenzija prenesenih slik in oblika e-poštnih sporočil. V primeru, da želi podjetje spremljati statistiko obiskov preko storitve Google Analytics [16], je možen vnos identifikacijske številke sledenja. Vso potrebno programsko kodo Javascript pa nam, v primeru, da smo vpisali veljavno številko, v glavi HTML dokumenta ustvari CMS sam.

3.2.2 Urejanje vsebin

Lastnik spletišča lahko podrobno ureja vsebinske strani. Ker je takih strani ponavadi veliko, je na voljo tudi seznam, urejen po abecednem vrstnem redu naslovov strani. Vsebinskim stranem vnesemo naslov, opis in ključne besede. Poleg tega vsaki vsebinski strani določimo poljubno, še neuporabljeno enolično spletno povezavo (URL) preko katere bo dostopna. V namen spletne optimizacije je možno določiti ali se posamezna vsebinska stran prikaže v kazalih HTML in XML (Extensible Markup Language). Ker pa je poleg spletne optimizacije potrebno poskrbeti za lagodje obiskovalcev, so za različne vsebinske strani na voljo različne oblikovne predloge, ki se jih lahko po potrebi tudi preprosto dodaja. Predloga je lahko narejena tako, da vključi tudi razne gradnike, kateri se določijo preko spletnega obrazca. Če se želi, se lahko eno izmed vsebinskih strani nastavi kot uvodno stran. Vsebinska stran se poljubno uredi z urejevalnikom, ki podpira vnos različnih stilov pisave (odebeljen, poševen, podčrtan), poravnavo besedila, barvanje besedila, vnos slik ter spletnih povezav, kar je razvidno iz slike 3.8.



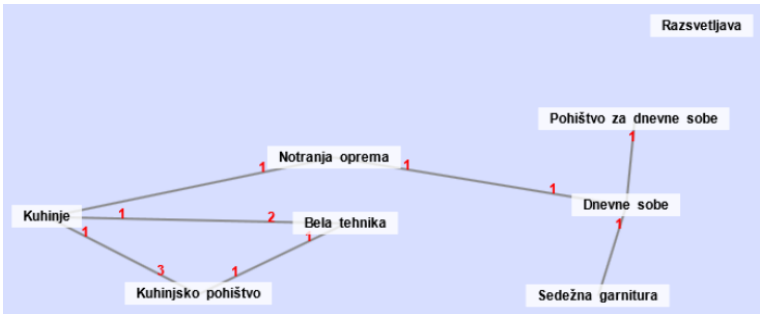
Slika 3.8: Prikaz funkcionalnosti urejevalnika vsebin.

V primeru da vnesemo spletno povezavo, ki kaže izven spletišča, se k povezavi doda še atribut target, katerega vrednost postane _blank, kar pomeni, da bo klik na takšno povezavo povzročil, da se bo odprla v novem zavihku brskalnika in obiskovalec strani ne bo zapustil spletišča. V kolikor pa vnesemo spletno povezavo, ki kaže znotraj spletišča, bo sistem to prepoznal in jo ustrezno zašifrirano shranil v podatkovni bazi, da povezava morebiti ne bo postala zlomljena ob kasnejšem spreminjanju ali brisanju vsebine na ciljnem naslovu. Da spletnih povezav ne vstavljamo na roke, so zato poleg urejevalnika dodani tudi padajoči meniji, ki vsebujejo spletne povezave strani ter modulov, da lahko le s klikom vstavimo želeno povezavo.

Vsaka vsebinska stran ima na voljo tudi dodatna polja, v katera lahko lastnik oz. pisec vsebin vpiše cilje, ki jih ima stran glede optimizacijskega vidika, oziroma katere spremembe glede optimizacije so bile narejene na določen datum. Vsebinskim stranem lahko pripišemo še opcijsko dodatno vsebino, kadar želimo, da je nek del vsebine prikazan v ločenem delu njene predloge. Prav tako lahko vsebinski strani določimo pomanjšano sličico (angl. thumbnail) ter pasico, če je prikaz slednje vključen v predlogi.

Ker sistem v vsebinah prepozna spletne povezave, ki kažejo znotraj spletišča, je na voljo tudi pregled povezav med vsebinskimi stranmi v obliki seznama in tudi v obliki grafa. Zelo je namreč pomembno in uporabno, da so sorodne vsebinske strani povezane med seboj in s pomočjo grafa postane tovrstno opravilo veliko lažje. Poleg grafa, ki je prikazan na sliki 3.9, je možen tudi izvoz povezav, naslovov, opisov in ključnih besed vsebinskih strani v zunanjo XLS datoteko, ki smo jo že omenili. Ponuja namreč dober pregled v Micro-

softovem programu Excel, kjer lahko vidimo osnovne podatke vseh vsebinskih strani naenkrat.



Slika 3.9: Grafični prikaz povezav med notranjimi stranmi. Številke ob straneh pomenijo koliko povezav vsebuje stran do strani, do katere vodi povezava. Stran, ki ne vsebuje povezav pa je prikazana samostojno.

Na zelo podoben način, kot je opisan za urejanje vsebinskih strani, se ureja tudi novice, le da za njihove polne spletne povezave velja, da vsebujejo še predpono modula.

Poleg samega urejanja vsebin je lastniku v administraciji mogoče preko spletnega vmesnika pošiljati e-pošto uporabnikom, ki so se predhodno prijavili na e-novice. Pošti je potrebno vpisati predmet (angl. subject), naslov in vsebino. Ker vsi odjemalci e-pošte nimajo podpore za prikaz sporočil, oblikovanih z jezikom HTML, je na voljo tudi opcija čistih tekstovnih sporočil. Pred pošiljanjem se vsekakor ustvari predogled, s katerim lastnik vidi, kako približno bo stranka videla sporočilo. Po potrditvi se vsa e-poštna sporočila ne odpošljejo takoj, ampak se vnesejo v podatkovno bazo in se odpošiljajo po predhodno definiranih časovnih intervalih. Če je od predhodnega pošiljanja e-pošte v podatkovni bazi še vedno nekaj čakajočih sporočil, nas sistem o tem predhodno obvesti.

Lastnik spletne strani lahko ureja še gradnike ter menije. Gre za elemente s kratko vsebino HTML, ki jih je možno prikazati v oblikovni predlogi ene ali več strani oziroma modulov. Gradnike ter menije se preprosto uredi z uporabo za to specializiranih spletnih obrazcev.

3.2.3 Delo z datotekami

Lastnik spletne strani lahko preko administrativnega vmesnika na strežnik prenese želene datoteke. Na voljo sta dva različna nalagalnika datotek. Prvi je vgrajen v urejevalnike vsebin. Preko nalagalnika je možno na enostaven način iz računalnika prenesti slike ali druge datoteke in jih neposredno vstaviti v vsebino, katero se ureja. Drugi način je uporaba ločenega nalagalnika, ki je napisan v tehnologiji Flash [17]. Podpira sočasen prenos več datotek hkrati in je zato primernejši, kadar želimo recimo naložiti množico slik v določeno galerijo ali prenesti veliko drugih datotek. V primeru večjih datotek pa je še vedno bolj priporočena uporaba odjemalca FTP (File Transfer Protocol), ki se lahko uporabi tudi sicer. Zamenjata se lahko tudi logotip in pasica (angl. banner), ki se morata pred zamenjavo preko vgrajenega nalagalnika prenesti na strežnik v ustrezno mapo. Pasico je možno prilagoditi za vsako vsebinsko stran posebej, saj je včasih dobro, da se uporabijo različne.

Poglavje 4

Tehnični vidik

V okviru uporabniškega vidika smo si ogledali opise funkcionalnosti ter upravljanje s sistemom, v okviru tehničnega vidika pa si bomo ogledali programsko opremo, ki je potrebna za izgradnjo ter namestitev spletišča na strežnik, kasneje pa tudi podrobnejši opis delovanja sistema.

4.1 Potrebna programska oprema

4.1.1 Spletni strežnik Apache2

Ker gre za spletno aplikacijo, je potrebno predhodno namestiti strežnik Apache2 [20], ki bo skrbel za prejem zahtev ter obiskovalcem spletišča odgovarjal z dokumenti HTML. Apache2 je po priljubljenosti uporabnikov s celega sveta na prvem mestu, saj gre za odprto-kodno programsko opremo, za katero ni potrebno plačati. Poleg tega je na voljo za UNIX in Windows NT sisteme.

4.1.2 Prevajalnik PHP

Z namestitvijo PHP dodamo spletnemu strežniku modul, ki ga poveže s programsko opremo PHP. Modul bo poskrbel, da spletni strežnik ne bo odgovoril obiskovalcu direktno z dostopno datoteko PHP, temveč jo bo prej posredoval prevajalniku PHP, ki jo bo ustrezno prevedel in posredoval nazaj strežniku v obliki HTML, kar bo kasneje prejel tudi obiskovalec spletišča.

4.1.3 Strežnik MySQL

Za shranjevanje podatkov CMS sistem uporablja podatkovno bazo MySQL, zato je potrebno namestiti tudi omenjeni strežnik. Dostop do podatkovne baze opravijo funkcije, ki so že vgrajene v jedro PHP. Priporočena je namestitev čim novejše različice strežnika. Iz varnostnih razlogov pa je dobro, da za CMS sistem ustvarimo ločenega uporabnika, ki bo imel dostop le do podatkovne baze aplikacije. Za uvoz in izvoz podatkovne baze je priporočljivo namestiti tudi spletno orodje PhpMyAdmin [21]. To je aplikacija, ki predstavlja spletni vmesnik za delo s podatkovnimi bazami. Ponuja dober pregled nad vsemi podatki, ki jih uporabljajo aplikacije, omogoča pa tudi napredno iskanje in sortiranje podatkov.

4.1.4 Cron Daemon

Če bomo uporabljali pošiljanje e-pošte in avtomatsko predlaganje kazala strani spletnim iskalnikom, moramo imeti na strežniku nameščen UNIX sistem, na primer Linux, ki podpira proces »cron daemon« [22]. To je proces, ki ob vnaprej določenih časovnih intervalih zaganja uporabniške skripte oziroma programe. Za omenjeni proces ni nujno, da je nameščen na delovnem okolju, kjer programer oblikuje spletišče, je pa priporočljivo, da je na strežniku, ki bo gostil spletišče, ko bo le to objavljeno. Skripti za pošiljanje e-pošte in obveščanje spletnih iskalnikov sta ločeni v datotekah mail.php ter ping.php. V obeh je potrebno programerju nastaviti skrivno geslo za dostop. V spodnji kodi 4.1 spremenimo parameter skrivnost na nekaj naključnega, kar bo predstavljalo geslo za zagon skripte.

```
define("CRON_SECRET", "skrivnost");
```

Izsek kode 4.1: Varnost datotek, do katerih dostopa Cron Daemon.

Priporočeno je, da za obe datoteki nastavimo isto vrednost, ni pa nujno. Pomembno je le, da je geslo dovolj naključno, da ga uporabniki ne morejo

preprosto uganiti. V datoteki mail.php nastavimo še koliko e-poštnih sporočil se največ odpošlje v programski zanki. Priporočena vrednost je 50 e-poštnih sporočil naenkrat, če predpostavimo zagon skripte na vsake pol ure. Če nam iz kakršnega koli razloga ni omogočeno upravljanje z Cron opravili, lahko skripti izvršimo tudi ročno. Primer prikazuje spodnji spletni naslov:

http://www.test.si/cron/mail.php?secret=skrivnost

V kolikor pa lahko sami vnašamo opravila Cron, je pogosto potrebno vnesti ukaz, ki ga prikazuje spodnja vrstica, katera je včasih nekoliko odvisna od ponudnika gostovanja:

php -f /home/username/public_html/cron/mail.php skrivnost

Zgornji skripti z enim zagonom opravita svoje delo na vseh jezikovnih različicah spletišča, zato ju ni potrebno navesti večkrat, kadar je spletišče na voljo v več jezikih.

4.1.5 Urejevalnik tekstovnih datotek

Za urejanje stilskih datotek in nekaterih nastavitev spletišča je potrebno na razvijalnem okolju imeti pregleden urejevalnik, po možnosti s preverjanjem sintaktične veljavnosti in možnostjo barvanja programske kode.

4.2 Uporabljene tehnologije

4.2.1 HTML

HTML (HyperText Markup Language) [23] je oblikovni jezik, ki je namenjen za izgradnjo spletnih strani. Vsebina oziroma struktura spletne strani je zapisana v dokumentu, kjer morajo biti pravilno uporabljene in ugnezdene značke HTML, da lahko spletni brskalniki nedvoumno sestavijo izgled spletne strani 4.2. HTML je standardni jezik za izdelavo spletnih strani že od leta 1990 naprej.

```
<html>
<head>
  <title>Naslov spletne strani</title>
</head>
<body>
  <div id="content" class="text">Vsebina spletne strani</div>
<div class="text php"></div>
</body>
</html>
```

Izsek kode 4.2: Enostaven primer spletne strani.

Vsaka spletna stran sestoji iz glave (angl. head) in telesa (angl. body). V glavi dokumenta je najbolj pomemben element `title`, ki predstavlja naslov spletne strani. Vsebino znotraj elementa `body` pa v današnjih časih razdelimo z elementi `div`, v preteklosti pa se je to delalo s tabelami, ki se danes uporabljajo izključno za razpredelnice s podatki.

4.2.2 CSS

Izgled spletne strani lahko izboljšamo že z dopisovanjem atributov v značke jezika HTML, vendar je bolj priročna uporaba zunanje datoteke CSS (Cascade Style Sheet) [24]. V njej točno deklariramo oblikovna pravila za posamezne skupine elementov 4.3, hkrati pa se lahko na preprost način izognemo večkratnemu definiranju istih oblikovnih pravil na različnih elementih. Po drugi strani pa ohranimo HTML kodo preglednejšo in lažjo za urejanje.

```
* { margin:0; padding:0; }
body {
  font-family:Arial, Helvetica, sans-serif;
  font-size: 12px;
  text-align:justify;
}
div#content, div.php {
  margin: auto;
  clear: both;
  width: 800px;
}
div.text {
  color:#505050;
}
div.php {
  color: blue !important;
}
```

Izsek kode 4.3: Primer oblikovnih pravil datoteke CSS.

Zvezdica pomeni, da naj oblikovno pravilo velja za vse elemente, kadar pa želimo pravilo napisati za določen tip elementov, vpišemo ustrezno ime elementa, na primer `body`. Pravila za izrecno določen element navedemo tako, da imenu elementa dodamo znak `#` in pripišemo njegovo enolično oznako. Na skupine elementov oz. razrede se v datoteki CSS sklicujemo z uporabo pike. Kadar pa želimo da določena pravila veljajo za več skupin, jih naštejemo ter

ločimo z vejico. Sama oblikovna pravila se napišejo znotraj zavitih oklepajev, njihova sintaksa pa je preprosta. Lastnost `!important` v zadnjem razredu pomeni, da če pride do ponavljanja deklaracije, da naj se uporabi sledeča.

4.2.3 PHP

PHP (PHP HyperText Preprocessor) je najbolj razširjen programski jezik, ki se uporablja pri gradnji dinamičnih spletnih strani. Datoteke PHP se prevedejo na strežniku, odjemalec pa dobi končni rezultat v čisti obliki HTML. PHP je zelo razširjen tudi zato, ker je inštalacijska programska oprema na voljo brezplačno, hkrati pa je jezik dokaj preprost 4.4 in s tem primeren tudi za začetnike. Sledi primer kode PHP:

```
function randomDigit() {
    return rand(0,9);
}

$a = randomDigit();
echo $a;
```

Izsek kode 4.4: Preprost primer programske kode PHP.

Pravkar napisana koda najprej deklarira funkcijo, ki vrne naključno številko. Nato spremenljivki `a` priredimo eno izmed naključnih števk, na koncu pa to vrednost še izpišemo. Kodo PHP je možno vstaviti recimo znotraj elementa `div` z razredom `php` iz prejšnjega primera kode HTML.

Za pravilno delovanje sistema je potrebno namestiti PHP različice 5.3 ali novejše. Potrebne nastavitve pa so prikazane na spodnjem odseku kode 4.5:

```
file_uploads = On
upload_max_filesize = 20M
post_max_size = 20M
memory_limit = 64
magic_quotes_gpc = Off
```

Izsek kode 4.5: Potrebne nastavitve modula PHP.

V dokumentih PHP je vedno uporabljena krajša različica začetne značke, zato je potrebno omogočiti še `short open tag`. Zaradi večjezične podpore je potrebno vključiti še modul `php_gettext`. Kadar želimo programski opremi PHP omogočiti pošiljanje e-pošte, moramo namestiti tudi poštni strežnik. Kateri poštni strežnik namestimo, je vseeno, pomembno pa je, da določimo pravilna vrata (angl. port) za povezavo do strežnika.

4.2.4 SQL

Pri delu s podatkovno bazo MySQL se uporablja jezik SQL (Structured Query Language). Omogoča raznoliko delo s podatki, osnovni ukazi pa omogočajo vnos, urejanje, brisanje podatkov. Z nekoliko zahtevnejšimi poizvedbami pa se da realizirati tudi razna sortiranja, štetja ter iskanja. Prej omenjeni programski jezik PHP ima vgrajene funkcije, s katerimi se povežemo na strežnik MySQL, kasneje pa dostopamo do podatkov 4.6:

```
SELECT * FROM page WHERE Deleted = '0' ORDER BY ID DESC LIMIT 10;
```

Izsek kode 4.6: Enostavna poizvedba SQL.

Zgornji stavek izbere največ 10 vrstic iz relacije `page`, katerih atribut `Deleted` ima vrednost `0`, na koncu pa jih uredi po padajočem primarnem ključu `ID`.

4.2.5 JavaScript

JavaScript [25] je programski jezik, katerega se, podobno kot PHP, vstavi med elemente HTML. Pomembna razlika je, da se le-ta izvaja v brskalniku, ki teče na odjemalčevi strani. Na ta način je možno strežnik do določene meje razbremeniti sortiranja, štetja ali kakšnega drugega preprostega dela. JavaScript se je začel obširno uporabljati s prihodom knjižnice jQuery [26], ki programerjem bistveno olajša programiranje interaktivnih aplikacij. Z dodano podporo za animacije pa lahko spletno stran naredi veliko bolj zanimivo za obiskovalce. Z uporabo prototipov lahko napišemo funkcije, ki delujejo nad določenimi elementi 4.7.

```
$.fn.randomDigit = function() {
  var digits = "0123456789";
  return Math.floor(Math.random() * digits.length);
}

var a = $.fn.randomDigit();
alert(a);
```

Izsek kode 4.7: Primer preproste funkcije, ki je dodana med obstoječe jQuery prototipe. Vloga funkcije pa je podobna, kot pri prej napisani funkciji PHP.

4.2.6 JSON

JSON (JavaScript Object Notation) je tekstovni standard, namenjen izmenjavi podatkov. CMS sistem uporablja JSON za obveščanje odjemalca, da je strežnik opravil določeno delo in za izmenjavo nekaterih osnovnih informacij. JSON je nekoliko težje človeku berljiv zapis 4.8, vendar je njegova prednost v tem, da ima sporočilo minimalno količino sintaktičnih podatkov, ki so nujno potrebni za sintaktično pravilnost in nedvoumnost vsebine. Uporaben je predvsem takrat, kadar želimo zmanjšati količino prenosa nekompresiranih podatkov.

```
{"Status":1,"Message":"Zahvaljujemo se za vase sporocilo."}
```

Izsek kode 4.8: Primer sporočila tipa JSON, ki ga strežnik posreduje odjemalcu po prejemu kontaktnega sporočila.

Ustrezno daljše sporočilo XML pa je prikazano z izsekom kode 4.9:

```
<Sporocilo>
  <Status>1</Status>
  <Message>Zahvaljujemo se za vase sporocilo.</Message>
</Sporocilo>
```

Izsek kode 4.9: Primer sporočila XML, ki ustreza sporočilu JSON.

4.2.7 Datoteka .htaccess

Ker so današnje aplikacije dinamične, je izpis njihovih vsebin pogosto določen na podlagi URL naslova, v katerem se pogosto nahajajo tudi spremenljivke. Takšni zapleteni URL naslovi niso prijazni, ne za uporabnika, ne za iskalnike, niti za programerja. Namen datoteke `.htaccess` je, da sistemu omogoči podporo uporabniku prijaznih URL naslovov. Primer URL naslova z uporabo spremenljivk:

http://www.test.si/index.php?page=product&product_id=10

Lepši, ustrezno preslikan naslov z uporabo datoteke `.htaccess` 4.10 pa bi izgledal takole, lahko pa bi se odločili tudi za drugačno obliko preslikovanja:

http://www.test.si/product/10

```
Options +FollowSymlinks
RewriteEngine On

RewriteRule ^en/(.*)$ $1 [NC]
RewriteRule sitemap.xml sitemapxml [NC,L]
RewriteRule pages.xls pagesxls [NC,L]
RewriteRule robots.txt robotstxt [NC,L]
RewriteRule ^([a-zA-Z0-9-\/]+)(/)?$ ?%{QUERY_STRING} [NC,L]

Redirect permanent /star-url.htm /nov-url
```

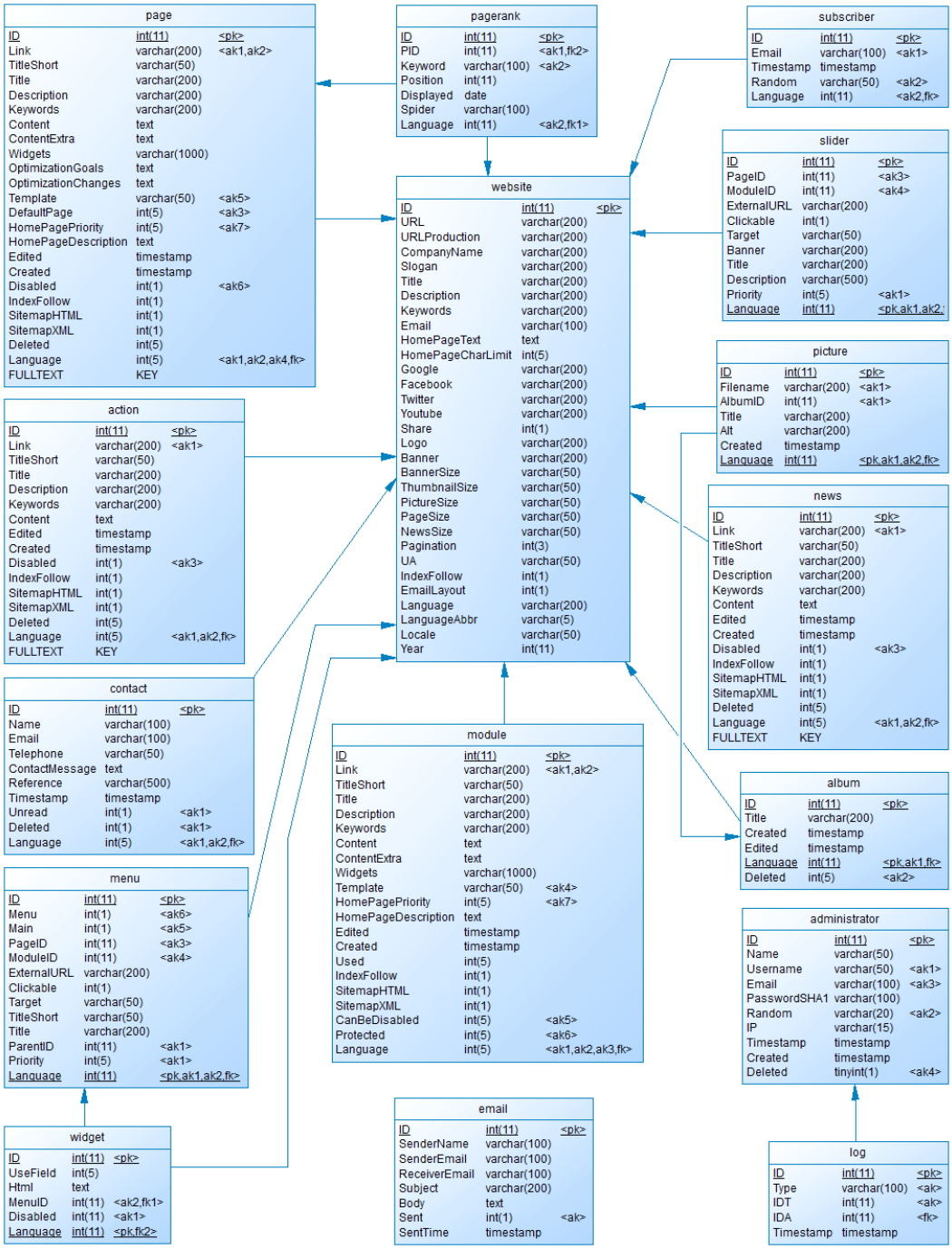
Izsek kode 4.10: Vsebina datoteke .htaccess.

Včasih strežnik Apache2 zahteva, da vnesemo prvo napisano vrstico, sicer lahko ustvari napako. Pove, da naj strežnik sledi simboličnim povezavam, kadar dostopamo do neke datoteke na strežniku. Druga vrstica pove, da naj strežnik začne z upoštevanjem pravil, napisanih pod njo. Napisana pravila, ki sledijo, imajo dva parametra. Prvi je URL naslov, katerega vidi uporabnik, drugi parameter pa pove v kakšen URL naslov naj se preslika prvotni. Torej

kakšen URL naslov bosta v resnici videla strežnik ter aplikacija, napisana v jeziku PHP. Prvi parameter se ponavadi podaja z uporabo regularnega izraza. Če želimo, da strežnik upošteva tudi preusmeritve, jih po koncu zgoraj naštetih pravil navedemo v primerni obliki. Princip preusmeritev je uporaben predvsem takrat, kadar se stranki staro spletišče nadomesti z novim, pri čemer preusmerimo URL naslove starih vsebin na nove.

4.3 Načrt podatkovne baze

Na naslednji strani si lahko na sliki 4.1 ogledamo načrt podatkovne baze. Naštete so vse uporabljene relacije, ki jih CMS sistem uporablja. Za lažje razumevanje so dodane tudi povezave med posameznimi relacijami.



Slika 4.1: Model podatkovne baze lastnega CMS sistema.

Iz priloženega načrta ni težko ugotoviti, da je osrednja relacija `website` namenjena za shranjevanje informacij o spletišču. Prav tako vsak vnos v tej relaciji pomeni svojo jezikovno različico. Primaren ključ `ID` iz relacije `website` v večini ostalih relacij predstavlja tuj ključ, kjer je poimenovan `Language`. Kot vidimo, se lahko za vsako jezikovno različico vnesejo povsem poljubne vsebinske strani (relacija `page`) ter moduli (relacija `module`). To pomeni, da nima vsaka vsebinska stran nujno svojega ustreznega prevoda, niti ni nujno, da so na vseh jezikovnih različicah vklopljeni enaki moduli. Za vsako vsebinsko stran lahko vnesemo tudi podatke o pozicijah na spletnih iskalnikih, ki se shranjujejo v relaciji `pagerank`. V primeru vklopa modula za urejanje novic se novice shranjujejo v relaciji `news`, v primeru vklopa modula za urejanje vsebin z raznimi prodajnimi akcijami ter ugodnostmi pa se vsebine shranjujejo v relaciji `action`. Obiskovalci spletišča se imajo možnost prijaviti na e-novice, zato se njihovi podatki zapišejo v relaciji `subscriber`. Vsaki jezikovni različici spletišča lahko uredimo na uvodni strani tudi drsnik s slikami ter kratkimi besedili, za njihovo shranjevanje pa je namenjena relacija `slider`. Podobno ima vsaka jezikovna različica tudi svoje menije. Izgled menija je povsem določen z oblikovnimi pravili, iz tehničnega vidika pa ločimo glavni meni, ki ga običajno vključimo v zgornji del spletne strani ter pomožne menije, ki jih lahko vključimo kjerkoli na spletišču. Za shranjevanje elementov menijev je namenjena relacija `menu`. Za razne namene lahko ustvarimo poljubne gradnike z vsebino, ki se shranjujejo v relaciji `widget`. Ker lahko v gradnik vključimo tudi pomožni meni, je zato potreben tuj ključ s šifro menija v relaciji `widget`. V primeru vklopa modula s kontaktnim obrazcem se povpraševanja obiskovalcev shranjujejo v relaciji `contact`. V primeru vklopljenega modula za gradnjo galerij, se albumi shranijo v relaciji `album`, pripadajoče slike pa v relaciji `picture`. Podatki o administratorjih spletišča se shranjujejo v relaciji `administrator` in so skupni za vse jezikovne različice zaradi enostavnejše uporabe. Vse aktivnosti administratorjev spletišča se beležijo v relaciji `log`. Kot smo omenili, se elektronska sporočila pred odpošiljanjem shranjujejo v bazi, za njih pa je namenjena relacija `email`.

4.4 Opis delovanja

Poglejmo si potek delovanja od prejema obiskovalčevega zahtevka do prikaza spletne strani:

1. Obiskovalec strani obišče spletišče, pri čemer brskalnik od strežnika zahteva vsebino iz določenega URL naslova.

2. Spletni strežnik bo zaradi pravil, napisanih v datoteki `.htaccess`, vedno posredoval datoteko `index.php` modulu PHP.

3. Datoteka `index.php` poskrbi za nalaganje vseh potrebnih komponent sistema:

   (a) Najprej se naložijo prosto dostopne knjižnice, ki so na voljo na spletu. Takšen primer so recimo knjižnica SimpleImage [27] za delo s slikami, knjižnica SimpleHTMLDomParser [28] za napredno delo z jezikom HTML, knjižnica PHPMailer [29] za enostavno pošiljanje e-pošte ter knjižnica za ustvarjanje datotek XLS, imenovana PHPExcel [8].

   (b) Sledi uvoz datoteke `functions.php`, katera vsebuje funkcije, ki smo jih napisali sami. Večinoma gre za funkcije, ki preverjanje vrednosti spremenljivk, funkcije za delo z mapami na strežniku in funkcije za delo s piškotki (angl. cookies) ter preusmeritvami.

   (c) Naslednja datoteka, ki se uvozi, je datoteka `class.php`. Vsebuje razrede, ki se uporabljajo za učinkovito delo z vsebinskimi stranmi (`class Page`), moduli (`class Module`), uporabniki (`class User`) in še bi lahko naštevali.

   (d) Ko so vse potrebne funkcije in razredi naloženi, se lahko z vključitvijo datoteke `database.php` ustvari povezava do podatkovne baze. V primeru neuspešne povezave, bo sistem na to opozoril z ustreznim opisom napake.

(e) Datoteka `globals.php` iz zahtevanega URL naslova ugotovi jezikovno različico spletišča, datoteka `localization.php` pa določi jezik, v katerega bo modul `php_gettext` prevedel nize iz izvorne kode.

(f) V datoteki `security_login.php` se preveri, ali je obiskovalec prijavljen v sistem kot administrator, ter pregleda prejet strežniški zahtevek za obstoj morebitnih piškotkov.

(g) Iz prejetega URL naslova se v datoteki `page.php` ugotovi, kaj naslov predstavlja – vsebinsko stran, modul ali datoteko. Če se ugotovi zadetek, se poleg kreiranja raznih objektov ustrezno nastavi oblikovna predloga strani. V primeru zgrešitve, se naredi preusmeritev na domačo stran, po potrebi pa bi lahko strežnik odgovoril z napako.

(h) Ker lahko oblikovne predloge vsebujejo razne spletne obrazce, se le ti naložijo v datoteki `forms.php`.

4. Naloži se oblikovna predloga, ki jo potrebuje stran iz zahtevanega URL naslova. Sistem uporablja dve vrsti oblikovnih predlog. Prva vrsta so zunanje oblikovne predloge, ki določajo skelet spletišča, druga vrsta pa so notranje oblikovne predloge, ki pogojujejo postavitev vsebin in so vključene v eno izmed zunanjih.

5. Modul PHP po opravljenem delu spletnemu strežniku vrne dokument HTML z ustreznimi atributi v glavi dokumenta ter ustrezno oblikovano spletno stranjo.

6. Spletni strežnik pošlje obiskovalcu spletišča dokument HTML.

7. Obiskovalčev brskalnik iz značk HTML in pravil iz datoteke CSS izriše ustrezno oblikovano spletno stran.

Podrobnejši opis delovanja nekaterih funkcionalnosti si bomo ogledali v nadaljevanju ob primerih programske kode.

4.5 Primeri programske kode z razlago

4.5.1 Oblikovne predloge spletišča

Ker smo v prejšnjih poglavjih omenili, da je možno vsebinskim stranem ter modulom enostavno urejati oblikovne predloge, si na tem mestu ogledamo zglede na odsekih kode 4.11, 4.12 ter 4.13.

```
<div class="content-left content-lightbox">
  <?= $page->GetShareButtons(); ?>
  <?= $page->H1(); ?>
  <?= $page->GetContent(); ?>
  <?= $page->GetEditButton(); ?>
</div>
<div class="sidebar-right">
  <?= $page->GetWidgetsHtml(); ?>
</div>
```

Izsek kode 4.11: Ena izmed oblikovnih predlog vsebinskih strani. Razdeljena je na levi stolpec, kjer je izpisana vsebina, v desnem stolpcu pa so izrisani gradniki, katere za posamezno vsebinsko stran izberemo v administraciji.

```
<div class="content">
  <div class="home-text">
  <?= widget('Home-Text', 'homepage-text'); ?>
  <?= widget('Gallery', 'gallery', true, array('normal', 4)); ?>
  </div>
  <div class="home-news">
  <?= widget('Home-News', 'home-news', true, array(4, 100)); ?>
  </div>
</div>
```

Izsek kode 4.12: Oblikovna predloga uvodne strani. V prvem delu se izpiše predstavitveno besedilo, pod njim pa predogled zadnjih nekaj galerij. Drugi del pa vsebuje gradnik, v katerem se izpišejo zadnje vnesene novice.

```
<html>
<?
define('CMS_CSS', 'style.css');
include("header.php");
?>
<body id="page">
  <div id="header">
    <div id="menu-wrap">
      <div id="menu">
        <?= $website->Menu->GetHtml(); ?>
      </div>
    </div>
    <div id="header-center">
      <div id="logo">
        <?= $website->GetHtmlLogo(); ?>
      </div>
      <div id="search-form">
        <?= widget('Search', 'search-wrapper', true); ?>
      </div>
    </div>
  </div>
  <div id="container">
    <div id="container-line"></div>
    <div id="main">
    <?
    include("include/templates/".$page->GetTemplateLink()."php");
    ?>
    </div>
  </div>
  <? include("footer.php"); ?>
  <script type="text/javascript" src="js/tinymce-init.js" />
  <script type="text/javascript" src="js/html-processing.js" />
</body>
</html>
```

Izsek kode 4.13: Primer zunanje oblikovne predloge, ki določa skelet spletišča.

Oblikovna predloga iz izseka kode 4.13 najprej definira ime datoteke CSS, ki vsebuje oblikovna pravila. Vključi tudi datoteko PHP, ki ustvari glavo dokumenta HTML. Kot vidimo se v oblikovni predlogi nahajajo glavni meni, logotip podjetja ter iskalno polje. V kasnejšem razdelku pa se vključi notranja oblikovna predloga vsebinske strani ali modula. Nato se vključi tudi noga (angl. footer), ki najpogosteje vsebuje podatke o lastniku ter izdelovalcu spletne strani, lahko pa vanjo vključimo tudi druge elemente. Na koncu pa se vključita še zunanji datoteki JavaScript, ki poskrbita za ustrezno oblikovanje polj za vnos besedila ter lepše oblikovanje nekaterih elementov HTML.

4.5.2 Ugotavljanje jezikovne različice spletišča

```
function urlToWebID($url) {

  $cols = array("URL", "URLProduction");
  $type = 0;
  $error = true;
  $web_id = 0;
  $server_vars = "";

  for($i = 0; $i < count($cols); $i++) {
    $q = mysql_query("SELECT * FROM website");
    $shortest = 0;
    while($w = mysql_fetch_array($q)) {
      $found = false;
      if(strlen($w[$cols[$i]]) > $shortest) {
        $found = isSubURL($url, $w[$cols[$i]]);
      }
      if($found) {
        $shortest = strlen($w[$cols[$i]]);
        $server_vars = trim(substr($url,$shortest), "/");
        $error = false;
        $web_id = $w['ID'];
        $type = $i;
      }
    }
  }
  $a = array($error, $web_id, $type, $cols, $server_vars);
  return $a;
}
```

Izsek kode 4.14: Funkcija `urlToWebID()`, ki na podlagi prejetega URL naslova ugotovi pripadajočo jezikovno različico spletišča.

Deklaracija funkcije, prikazane na odseku kode 4.14, se nahaja v datoteki `functions.php`, za njen klic pa poskrbi datoteka `globals.php`. Funk-

cija sprejme parameter url, ki predstavlja trenuten prejeti URL naslov na strežniku. Primer vrednosti parametra, ki ga lahko sprejme funkcija izgleda takole:

www.test.si/testna-stran-1

Vsaka jezikovna različica spletišča je lahko istočasno dostopna iz dveh različnih URL naslovov. Eden je končni URL naslov, ko je spletišče objavljeno, drugi pa je produkcijski URL naslov, ko se spletišče še ureja. Funkcija najprej nastavi določene lokalne spremenljivke, kasneje pa v programski zanki za oba tipa URL naslovov pregleda, če se prejeti URL naslov ujema s katerim izmed naštetih v podatkovni bazi. V primeru, da je najden zadetek, algoritem ustrezno nastavi spremenljivke, vsekakor pa jih vrne ob koncu preverjanja. Sledi pomen spremenljivk:

1. `error` – zavzame vrednost `true` ali `false` in s tem pove, ali je prišlo med ugotavljanjem različice spletišča do napake ali ne.
2. `web_id` - v primeru uspešno ugotovljene različice vsebuje identifikacijsko številko spletišča.
3. `type` – zavzame vrednost `0` ali `1` in pove ali je šlo za končni URL naslov ali za produkcijski.
4. `cols` – seznam, ki vsebuje še ustrezna imena stolpcev z URL naslovi spletišč v podatkovni bazi.
5. `server_vars` – vsebuje le povezavo brez osnovnega URL naslova spletišča. Iz zgornjega primera bi bila to vrednost `testna-stran-1`.

4.5.3 Graf povezav med vsebinskimi stranmi

Ker je grajenje kakovostnih povezav med vsebinskimi stranmi ena izmed pomembnejših prednosti spletišča, je zato primerno, da si ogledamo realizacijo te funkcionalnosti. Preden se lotimo analize zgornjih razredov, je potrebno omeniti, da se povezave med vsebinskimi stranmi v podatkovni bazi ne shranjujejo v polni obliki, temveč so shranjene zašifrirane, kar ponazarjata izseka kode 4.15 in 4.16.

```
<p>
  <a href="http://www.test.si/testna-stran-1">Povezava 1</a>
  <a href="http://www.test.si/druga-testna-stran">Povezava 2</a>
</p>
```

Izsek kode 4.15: Primer vsebine, ki jo lastnik spletišča vnese v urejevalnik.

```
<p>
  <a href="#P_1_1">Povezava 1</a>
  <a href="#P_1_2">Povezava 2</a>
</p>
```

Izsek kode 4.16: Primer prejšnje vsebine, shranjene v podatkovni bazi.

Povezave v vsebini so torej shranjene v obliki `#P_X_Y`, pri čemer `P` pove, da gre za povezavo do vsebinske strani, število `X` pove identifikacijsko številko jezikovne različice spletišča, število `Y` pa identifikacijsko številko vsebinske strani. Začetni znak `#` je uporabljen zato, da se ob izbrisu določene vsebinske strani v vsebinah izpisujejo povezave v obliki sidra (angl. anchor links). To so povezave, ki kažejo znotraj spletne strani, na kateri se nahajajo, klik na njih pa nikoli ne povzroči napake.

```
class PageLinks {

  public $In = array(); public $Out = array(); public $Pages;

  function __construct($pages) {
    $l = $_SESSION['Language'];
    $this->Pages = $pages;
    for($i=0; $i<count($pages); $i++) {
      preg_match_all('/\#P\_'.$l.'\_(\d+)/', $pages[$i]->Raw, $f);
      $f = $f[1];
      for($j=0; $j<count($found); $j++) {
        if(!array_key_exists($pages[$i]->ID, $this->Out)){
          $this->Out[$pages[$i]->ID] = array();
        }
        if(!array_key_exists($f[$j], $this->In)){
          $this->In[$f[$j]] = array();
        }
        if(array_key_exists($f[$j], $this->Out[$pages[$i]->ID])){
          $this->Out[$pages[$i]->ID][$f[$j]]++;
        }
        else {
          $this->Out[$pages[$i]->ID][$f[$j]] = 1;
        }
        if(array_key_exists($pages[$i]->ID, $this->In[$f[$j]])){
          $this->In[$f[$j]][$pages[$i]->ID]++;
        }
        else {
          $this->In[$f[$j]][$pages[$i]->ID] = 1;
        }
      }
    }
  }
}
```

Izsek kode 4.17: Razred `PageLinks`, ki sestavi seznam vhodnih in izhodnih povezav za vse vsebinske strani.

Konstruktor razreda `PageLinks` iz izseka kode 4.17 kot parameter sprejme seznam vseh vsebinskih strani spletišča, skozi katere se sprehodi v programski zanki. Vsebino vsake vsebinske strani pregleda za obstoj šifriranih povezav in si ujemajoče se identifikacijske številke shrani v seznam `found`. V naslednjih korakih se trenutno obravnavana identifikacijska številka shrani na ustrezno mesto v seznamu `Out` 4.18. Identifikacijska številka se zapiše tudi v seznam `In` 4.19, ki vsebuje vhodne povezave, za razliko od seznama `Out`, ki vsebuje izhodne povezave. Načeloma bi bil dovolj samo en tovrsten seznam. Dva seznama sta uporabljena zaradi razumljivejšega nadaljnjega programiranja.

```
Array (
  [1] => Array ( [2] => 2 )
  [2] => Array ( [1] => 1 )
  [3] => Array ( [1] => 1 [2] => 1)
)
```

Izsek kode 4.18: Seznam `Out`.

```
Array (
  [1] => Array ( [2] => 1 [3] => 1 )
  [2] => Array ( [1] => 2 [3] => 1 )
)
```

Izsek kode 4.19: Seznam `In`.

Iz seznama `Out` lahko razberemo, da vsebinska stran z identifikacijsko številko 1 vsebuje dve povezavi na vsebinsko stran s številko 2. Vsebinska stran s številko 2 vsebuje eno povezavo na vsebinsko stran s številko 1. Vsebinska stran s številko 3 pa vsebuje eno povezavo na vsebinsko stran s številko 1, eno povezavo pa na vsebinsko stran s številko 2. Podobno lahko preberemo tudi seznam `In`. Povezave na vsebinsko stran z identifikacijsko številko 1 vsebuje vsebinska stran s številko 2 in sicer eno takšno povezavo. Poleg tega pa eno takšno povezavo vsebuje še vsebinska stran s številko 3. Povezave na vsebinsko stran z identifikacijsko številko 2 vsebuje vsebinska stran s številko 1 in sicer dve takšni povezavi. Poleg tega pa eno takšno povezavo vsebuje še vsebinska stran s številko 3.

```
$pl = new PagesList();

$nodes = array();
$edges = array();

foreach($pl->PageLinks->Out as $key => $value) {

  foreach($pl->PageLinks->Pages as $page) {
    $nodes["P" . $page->ID] = array(
                    "type" => "Page",
                    "label" => $page->TitleShort
                    );
  }

  $edges["P" . $key] = array();

  foreach($value as $key_sub => $value_sub) {
    $edges["P" . $key]["P" . $key_sub] = array(
                    "count" => $value_sub,
                    "length" => 7
                    );
  }
}

$data = array();
$data["nodes"] = $nodes;
$data["edges"] = $edges;

echo json_encode($data);
```

Izsek kode 4.20: Preoblikovanje seznama `Out` v ustrezno obliko za gradnjo grafičnega prikaza povezav.

Spremenljivki `pl` v izseku kode 4.20 priredimo objekt razreda `PagesList`, ki vsebuje poleg seznama strani tudi objekt prej omenjenega razreda `PageLinks`. V programski zanki smo seznam `Out` preoblikovali v dva seznama `nodes` in `edges`, iz katerih se kasneje na grafu ustvarijo vozlišča in povezave. Omenjena seznama združimo v nov seznam `data`, ki ga PHP izpiše v formatu JSON, ki je opisan v prejšnjem poglavju.

```
$.ajax({
  type: "GET",
  url: "../../source/get/arbor-pages.php",
  success: function(data) {
    data = $.parseJSON(data);
    nodes = data["nodes"];
    edges = data["edges"];
    sys.merge({nodes:nodes, edges:edges})
    sys.parameters(_maps[map_id].p)
    $("#dataset").html(_maps[map_id].source)
  }
});
```

Izsek kode 4.21: Funkcija `ajax()`, napisana v jeziku JavaScript.

Zgornja funkcija 4.21 preko metode `GET` iz ustreznega spletnega naslova pridobi podatke za izris vozlišč in povezav, predstavljene v formatu JSON. Z nadaljnjimi algoritmi se na risalno površino izriše graf, ki smo ga lahko videli že na sliki 3.9.

4.5.4 Pošiljanje e-poštnih sporočil

```
class SimpleMail {

  ...

  public function Send() {

    if($this->Validate()) {
      $mailer = new FreakMailer();
      $mailer->SetSenderEmail($this->SenderEmail);
      $mailer->SetSenderName($this->SenderName);
      $mailer->Subject = $this->Subject;
      $mailer->ContentType = "text/html";
      $mailer->Body = $this->Body;
      $mailer->AddAddress($this->ReceiverEmail);

      $mailer->Send();

    }

    $this->Delete();

  }

  ...

}
```

Izsek kode 4.22: Razred `SimpleMail`, v katerem je zaradi obširnosti prikazana le metoda `Send()`, ki poskrbi za pošiljanje sporočila.

Kljub obširnosti pa je razred `SimpleMail` 4.22 napisan z namenom poenostavljenega dela z razredom `FreakMailer`, ki je še obširnejši in zahtevnejši za uporabo.

Metoda `Send()` najprej opravi validacijo, s katero preveri veljavnost atributov elektronskega sporočila. V primeru uspešne validacije metoda spremenljivki `mailer` priredi objekt razreda `FreakMailer` z ustreznimi spremenljivkami, nato pa ga posreduje poštnemu strežniku s svojo lastno metodo `Send()`.

```
class EmailsList {

  public $Emails = array(); public $Limit = 50;
  public $Table = "email";

  function __construct($n = 50) {
    $n = intval($n);
    $q = mysql_query("SELECT ID FROM " . $this->Table
              . " WHERE Sent = '0' ORDER BY ID ASC LIMIT " . $n);
    while($data = mysql_fetch_array($q)) {
      $e = new SimpleMail();
      $e->ConstructByID($data['ID']);
      array_push($this->Emails, $e);
    }
  }
  ...
  public function SendAll() {
    for($i = 0; $i<count($this->Emails); $i++) {
      $this->Emails[$i]->Send();
    }
  }
}

$e = new EmailsList();
$e->SendAll();
```

Izsek kode 4.23: Razred `EmailsList`, ki poskrbi za odpravo e-poštnih sporočil iz podatkovne baze. Zaradi obširnosti sta prikazana le konstruktor ter metoda `SendAll()`.

V izseku kode 4.23 spremenljivki `e` priredimo objekt razreda `EmailsList`. Ker ne podamo parametra, bo konstruktor razreda uporabil privzeto vrednost, ki predstavlja zgornjo mejo elektronskih sporočil, ki jih je možno odposlati v enem intervalu. Konstruktor iz podatkovne baze pridobi seznam najstarejših še neodposlanih elektronskih sporočil in iz njih ustvari objekte razreda `SimpleMail`, katere doda v svoj seznam `Emails`. S klicem metode `SendAll()` poskrbimo, da se v programski zanki odpošljejo vsa elektronska sporočila iz seznama `Emails`.

4.5.5 Gradnja kazala strani v obliki XML

```
function generateSitemap() {
  header("Content-type: text/xml; charset=utf-8");

  $xml =  "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n";

  $sl = new SitemapList("sitemap-xml");

  for($i = 0; $i < count($sl->Items); $i++) {
    $xml .= "<url>\n" .
    "\t<loc>" . $sl->Items[$i]->Loc . "</loc>\n" .
    "\t<lastmod>" . $sl->Items[$i]->LastMod . "</lastmod>\n" .
    "\t<changefreq>" . $sl->Items[$i]->Freq . "</changefreq>\n" .
    "\t<priority>" . $sl->Items[$i]->Priority . "</priority>\n" .
    "</url>\n";
  }

  $xml .= "</urlset>";
  echo $xml;
  exit;
}
```

Izsek kode 4.24: Funkcija `generateSitemap()`, ki poskrbi za gradnjo kazala strani v obliki XML.

Funkcija iz izseka kode 4.24 poskrbi, da bo strežnik ustvaril veljavno glavo odgovora. To je potrebno zato, da bo brskalnik ob prejetju odgovora strežnika natančno vedel tip datoteke in obiskovalcu ponudil ustrezno predstavitev informacije. Funkcija spremenljivki `sl` priredi objekt razreda `SitemapList`, ki v lastnem seznamu `Items` vsebuje vse potrebne informacije o vseh vsebinskih straneh ter modulih. Elementi iz seznama se kasneje ustrezno izpisujejo v programski zanki. Na koncu funkcije se nahaja še ukaz `exit`, ki prekine izvajanje skripte PHP, saj nadaljnje delo prevajalnika ni potrebno.

Poglavje 5

Primerjava spletnih CMS sistemov

Ker smo do tega trenutka spoznali vse omenjene CMS sisteme, si lahko sedaj ogledamo njihovo podrobnejšo primerjavo. Wordpress in Joomla sta na spletu prisotna že dlje časa, uporabljena pa sta pri milijonih spletišč. Prav tako je prosto dostopna tudi izvorna koda, kar nekoliko poveča nevarnost za spletne napade. Lasten CMS sistem uporablja trenutno zelo malo podjetij. Ker pa izvorna koda ni objavljena javno, je zato sistem tudi precej bolj varen.

Wordpress je zelo lahek za uporabo glede začetne postavitve spletišča, saj potrebujemo slediti le nekaj navodilom na spletnemu inštalacijskemu vmesniku, medtem ko Joomla in lasten CMS sistem potrebujeta nekaj programerskega znanja in je potrebno slediti obširnejšim navodilom. Zaradi zadnjega razloga so tudi stroški postavitve nekoliko višji. Prednost lastnega CMS sistema je v tem, da so kasnejši stroški vzdrževanja majhni, saj povezave med vsebinskimi stranmi ostanejo konsistentne ob spreminjanju njihovih URL naslovov. Poleg omenjenega pa lasten CMS sistem vsebuje vse potrebno za kakovostno izvedbo spletne optimizacije in njeno vzdrževanje.

V kasnejši fazi razvoja lastnega CMS sistema je dopuščena tudi izdelava spletnih trgovin, saj je sistem možno preprosto nadgraditi z uvedbo novih modulov. Velika prednost lastnega CMS sistema je tudi v odlični podpori za

večjezičnost. V primeru, da želi podjetje večjezično spletišče, se lahko jezikovne različice nahajajo na poljubnih URL naslovih – na različnih domenah, poddomenah ali mapah, s tem da se aplikacija na strežniku nahaja samo enkrat in se ne podvaja. To pomeni, da je tudi podatkovna baza skupna za vse jezikovne različice. Za razliko od naše rešitve pa je potrebno pri sistemu Wordpress za vsako jezikovno različico ustvariti ločeno instanco baze. Druga možnost pa je, da se zadovoljimo z uporabo vtičnika, ki ne podpira vnosa meta podatkov za ostale jezikovne različice. Joomla je nekoliko bolje pripravljena za večjezično podporo, saj je možno uporabiti le eno instanco podatkovne baze. Omejitev sistema Joomla je, da morajo biti vse jezikovne različice dostopne preko iste domene, zato se njihove URL naslove ponavadi loči z uvedbo map. Preprosta primerjava je predstavljena na naslednji strani s tabelo 5.1.

Tabela 5.1: Primerjava lastne rešitve s sistemoma Wordpress ter Joomla.

	WORDPRESS	**JOOMLA**	**Lasten CMS**
Leto izida	2003	2005	2012
Število spletišč	60+ milijonov	30+ milijonov	10-100
Izpostavljenost vdorom	Velika.	Velika.	Majhna.
Zahtevnost postavitve spletišča	Majhna.	Srednja.	Srednja.
Stroški postavitve in oblikovanja spletišča	100 €	200-300 €	200 €
Stroški vzdrževanja	Majhni, vendar hitreje naraščajo z številom vsebin.	Majhni, vendar hitreje naraščajo z številom vsebin.	Majhni.
Podpora za SEO	Slaba, lahko se izboljša z vtičniki.	Dobra.	Odlična.
Za kaj je namenjen?	Osebne strani, bloge, strani podjetij.	Osebne strani, bloge, strani podjetij, spletne trgovine.	Primarno za strani podjetij, lahko tudi za osebne strani.
Podpora za večjezičnost	Da, vendar z večjimi omejitvami.	Da, z manjšimi omejitvami.	Da, brez omejitev.
Konsistentnost notranjih povezav	Ne.	Ne.	Da, vključno z grafom povezav.
Možnost razširitev	Da.	Da.	Da.
Glavne uporabljene tehnologije	HTML, CSS, PHP, MySQL, JS.	HTML, CSS, PHP, MySQL, JS.	HTML, CSS, PHP, MySQL, JS.

Poglavje 6

Zaključek

Lasten CMS sistem vsekakor odlikuje boljše razmerje med stroški postavitve ter kakovostjo spletišča, kot pa ga imajo konkurenčni sistemi. Ker je odlično pripravljen za izvedbo spletne optimizacije, je najbolj uporaben pri postavitvi spletišč podjetij, saj jim sistem omogoča vse potrebno za doseg višjih pozicij na spletnih iskalnikih. To pomeni, da lahko podjetja, ki uporabijo lasten CMS sistem, pridobijo več potencialnih strank in s tem bistveno povečajo dobiček iz prodaje, tako na domačem, kot tudi na tujem trgu. Lahko pa CMS sistem uporabimo tudi za postavitev osebnih ali drugačnih spletišč.

Kot smo že omenili, je lasten CMS sistem možno dograjevati, zato je v prihodnosti planiranih kar nekaj izboljšav. Zaradi uvedbe zakona o uporabi spletnih piškotkov je veliko spletiščem prenehalo pravilno delovanje storitve Google Analytics [16]. Iz tega razloga je vsekakor v načrtu izdelava beleženja obiska in grafičen prikaz statistike obiskov. Ker pa se veliko podjetij danes na trgu predstavlja z raznimi izdelki in storitvami, je planirana tudi izdelava spletne trgovine. V začetni fazi razvoja bi lahko obiskovalec spletišča pregledoval in sortiral izdelke glede na njihove atribute, v kasnejši fazi pa opravil tudi varen nakup.

Literatura

[1] (2013) What is Content Management System (CMS). Dostopno na: http://www.comentum.com/what-is-cms-content-management-system.html

[2] (2013) What is Content Management System (CMS). Dostopno na: http://searchsoa.techtarget.com/definition/content-management-system

[3] (2013) WordPress.com - Get a Free Blog Here. Dostopno na: http://www.wordpress.com

[4] (2013) Joomla! The CMS Trusted By Millions for their Websites. Dostopno na: http://www.joomla.org

[5] (2013) Optimizacija spletnih strani. Dostopno na: http://sl.wikipedia.org/wiki/Optimizacija_spletnih_strani

[6] (2013) Google. Dostopno na: http://www.google.si

[7] (2013) Metadata on the Internet. Dostopno na: http://en.wikipedia.org/wiki/Metadata

[8] (2013) PHPExcel knjižnica. Dostopno na: http://phpexcel.codeplex.com

[9] (2013) A graph visualization library using web workers and jQuery. Dostopno na:
http://arborjs.org

[10] (2013) World Wide Web Consortium (W3C). Dostopno na:
http://www.w3.org

[11] (2013) The Pros and Cons of WordPress. Dostopno na:
http://prezi.com/1tgim6ngoxuc/the-pros-and-cons-of-wordpress-a-flag-tech-talk-lecture

[12] (2013) 5 Reasons Wordpress is great for SEO. Dostopno na:
http://www.notwillsmith.com/wordpress/wordpress-is-great-for-seo

[13] (2013) The WordPress Multilingual Plugin. Dostopno na:
http://wpml.org

[14] (2013) GNU General Public License. Dostopno na:
http://www.gnu.org/licenses/gpl.html

[15] (2013) Joomla version 2.5.6 Demo – Opensource CMS. Dostopno na:
http://www.opensourcecms.com/scripts/details.php?scriptid=39

[16] (2013) Uradna stran storitve Google Analytics. Dostopno na:
http://www.google.com/analytics

[17] (2013) Adobe Flash runtimes. Dostopno na:
http://www.adobe.com/products/flashruntimes.html

[18] (2013) Web service. Dostopno na:
http://en.wikipedia.org/wiki/Web_service

[19] (2013) jQuery lightBox plugin. Dostopno na:
http://leandrovieira.com/projects/jquery/lightbox

[20] (2013) The Apache HTTP Server Project. Dostopno na:
http://httpd.apache.org

[21] (2013) phpMyAdmin. Dostopno na:
http://www.phpmyadmin.net/home_page/index.php

[22] (2013) Cron. Dostopno na:
http://en.wikipedia.org/wiki/Cron

[23] (2013) HTML Tutorial. Dostopno na:
http://www.w3schools.com/html

[24] (2013) CSS. Dostopno na:
http://sl.wikipedia.org/wiki/CSS

[25] (2013) JavaScript. Dostopno na:
http://en.wikipedia.org/wiki/JavaScript

[26] (2013) jQuery – write less, do more. Dostopno na:
http://jquery.com

[27] (2013) Resizing images with PHP. Dostopno na:
http://www.white-hat-web-design.co.uk/articles/php-image-resizing.php

[28] (2013) PHP Simple HTML DOM Parser. Dostopno na:
http://simplehtmldom.sourceforge.net

[29] (2013) The classic email sending library for PHP. Dostopno na:
https://github.com/PHPMailer/PHPMailer

[30] (2013) PHP: Hypertext Preprocessor. Dostopno na:
http://php.net

[31] (2013) MySQL - The world's most popular open source database. Dostopno na:
http://www.mysql.com